# Efficient and Secure End-to-End Mobility Support in IPv6

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik
der Universität Fridericiana zu Karlsruhe (TH)

**genehmigte**

## Dissertation

von

**Christian Vogt**

aus Bonn

Tag der mündlichen Prüfung: 20. Juli 2007

Erste Gutachterin: Prof. Dr. Martina Zitterbart
Zweite Gutachterin: Prof. Dr. Carmelita Görg

*For my sister Kerstin —*
*for being encouragement and example*

# Preface

This thesis is the product of my work as a scientific associate at the Institute of Telematics at Universität Karlsruhe (TH), Germany. It would not have been possible without the teaching, help, patience, and inspiration of a great number of people, who all deserve more than simple acknowledgment.

Foremost, I wish to thank my doctoral advisor and thesis reviewer, Prof. Dr. Martina Zitterbart. She gave me the opportunity to pursue this thesis, and supported it with care and outstanding expertise up to its successful completion. She also admitted the funding I needed to present my work at national and international conferences and conventions, plus the time that was necessary to standardize major parts of the work. Additionally, the expensive testing environment that I could use to evaluate my ideas was by no means a matter of course.

For her review of my thesis and participation in the defense committee despite an extremely busy schedule, I want to express my gratitude to Prof. Dr. Carmelita Görg. Her suggestions as well as our technical discussions were highly valuable.

Many thanks are due to the Konrad-Adenauer-Stiftung for a graduate scholarship that included both financial aid and a distinguished seminar program. I am grateful in particular to Dr. Daniela Tandecki and Dr. Rita Thiele for their help and availability as trusted contact persons. Many thanks also to Dr. Gerd-Dieter Fischer for the cordial cooperation in organizing and carrying out two of the Konrad-Adenauer-Stiftung's scientific association's annual conventions.

I very especially wish to thank my colleagues and friends Mark Doll and Dr. Roland Bless for multifold excellent discussions and inspirations that shaped this thesis significantly. Thanks also for their readiness to listen to and reflect on countless ad-hoc ideas on making the Internet "yet a bit better".

It was a pleasure to work closely together with Tobias Küfner during the early phase of the dissertation, for supervising our first Master student and organizing the first student seminars. Thanks also to Marco Liebsch for the marvelous technical cooperation and friendship. I was delighted by the great collaboration with Oliver Stanze, Kilian Weniger, Erik-Oliver Blaß, Jidong Wu, Ingmar Baumgart, Christian Hübsch, and Christoph Wehrle on handling several student workshops and seminars. Gratitude is likewise due to Christoph Sorge, Lars Völker, Marc Torrent-Moreno, Peter Baumung, and my other colleagues for the cordial atmosphere at work.

Additional thanks go out to many more researchers for their much appreciated feedback and ideas: Jari Arkko, Dr. Pekka Nikander, Dr. James Kempf, Charles E. Perkins, Wassim Haddad, Rajeev Koodli, Gonzalo Camarillo, Dr. Lixia Zhang, Gabriel Montenegro, Gregory Daley, Dr. Lars Eggert, and Keiichi Shima.

Special thanks are due to Mrs. Astrid Natzberg, Mrs. Doris Weber, and Mrs. Dorothea Wagner from the institute's secretariat, and to Mrs. Ines Himpel from the Dean's office for the help on surmounting several administrative hurdles. Many thanks also to Mr. Gentiel Mussgnug, Mr. Detlev Maier, and Mr. Frank Winter for a superb technical support. I am furthermore indebted to my students Daniel Jungbluth, Constantin Schimmel, Ralf Beck, and Max Laier, all of whom added notably to the success of this thesis.

Last, but not least, my gratitude is due to my family and friends. Without their warmth and unwavering support, this thesis would never have been realized.

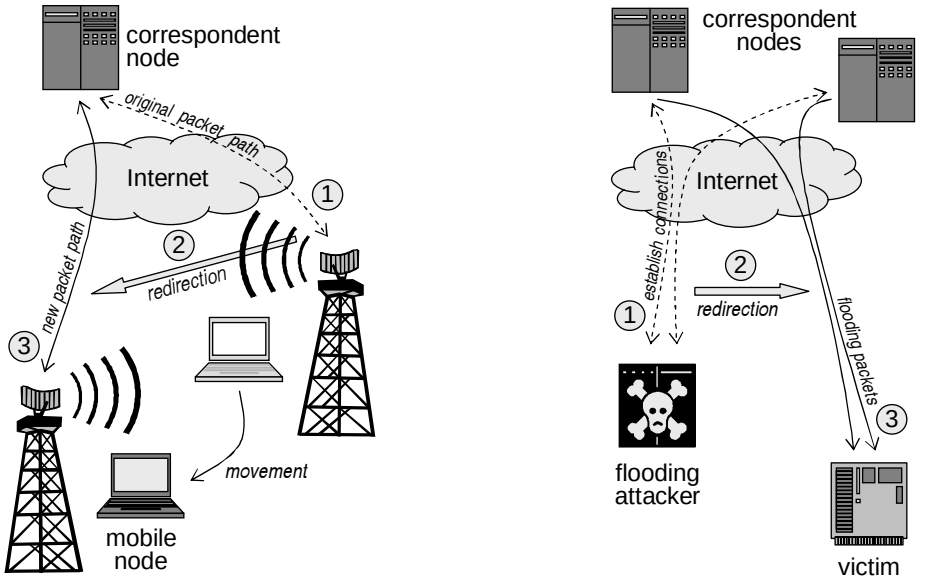
Thank you all.

# Contents

# 1. Introduction

Patterns of Internet use have been evolving rapidly in the recent years, shifting from delay- and bandwidth-tolerant applications like Web browsing, electronic mail, and file transfer towards more delay- and loss-sensitive soft-real-time applications, like audio and video streaming or broadcasts [46], and even hard-real-time services such as Internet telephony or videoconferencing [81, 96]. At the same time, people are increasingly expecting these services to be available anywhere, anytime, and through any access technology. These developments are the onset of a converged Internet, embracing wireless home and enterprise networks, hotspots in coffee shops and airport lounges, just as well as wide-area cellular networks.

The new heterogeneity in access technologies and the increasing mobility of Internet users calls for an efficient mechanism to hand active communication sessions over between different points of attachment to the Internet. Users may change their point of attachment during movements, for better service quality through a different access technology or provider, to save access costs, or for fail-over or load balancing. The base Internet Protocol, IP, does not provide such flexibility. It was mainly designed for stationary nodes with fixed IP addresses. Much effort has hence recently gone into mechanisms that transfer an active communication session from one IP address to another when a mobile node changes its point of attachment to the Internet. Figure 1.1(a) illustrates such a redirection based on a communication session between a mobile node and the correspondent node that it communicates with. The mobile node in the figure hands over from its current point of attachment to a new point of attachment. Along with this changes the IP address at which the correspondent node can reach the mobile node. The mobile node signals the new IP address to the correspondent node, and the correspondent node redirects the communication session to that new IP address. Part of the existing mobility mechanisms augment applications so as to survive IP address changes on either side of a connection. Others leave the applications untouched and add the necessary functionality into transport protocols that applications may use. A third approach is to build mobility support into IP directly. Changes in a mobile node's IP address are then invisible to transport protocols and applications. This thesis focuses on mobility support in IP.




(a) Packet redirection for a mobile node

(b) Redirection-based flooding

Figure 1.1: Legitimate versus malicious packet redirection

Any mechanism that enables a node to redirect packets from one IP address to another faces the threat of misuse [94]. Applications and transport protocols traditionally identify a node based on the node's IP address. Such IP-address-based identification is build on a trust on the routing infrastructure to deliver a packet for a particular IP address to exactly *that* IP address, or to drop the packet. Unless an attacker can compromise the routing infrastructure, it can eavesdrop on or tamper with packets exchanged between two nodes only if it is located on the communication path between these nodes. However, unless appropriate measures are provided, IP mobility support could give an attacker the ability to change the destination of packets that a remote node sends, transparently to applications and transport protocols, and without affecting the routing infrastructure. The attacker could so arrange to receive packets that should actually be delivered to a different node, regardless of where the attacker is located on the Internet. Similarly, the attacker could redirect packets that it is supposed to receive itself to a different node. This thesis addresses the latter threat, where the attacker causes a victim to receive unwanted packets.

Misuse of a mobility protocol to cause unwanted traffic is called a *redirection-based flooding attack*. A typical invocation of such an attack is shown in figure 1.1(b). The attacker first requests one or multiple correspondent nodes to transfer some large files. This involves a handshake between the attacker and a correspondent node, as indicated in the figure by the dashed arrows. Once a connection is set up, the attacker redirects the packet flow to the IP address of a victim. The attacker thereby clogs the victim's Internet access with packets that the attacker would actually be supposed to receive itself, impairing the victim's ability to pursue any reasonable communications. Redirection-based flooding attacks are an important threat because they enable the attacker to offload most of the effort it takes to wage the attack onto correspondent nodes. The correspondent nodes thereby unknowingly *amplify* the attacker's own effort: It is sufficient for the attacker to spoof acknowledgments for pretendedly received data in order to uphold the packet flows from the

correspondent nodes to the victim. The acknowledgments are smaller in both number and size than the packets that the correspondent nodes generate, so the damage caused to the victim can be much more severe than it could be if the attacker itself sent all packets to the victim directly. The only conventional type of flooding attack that provides a similar level of amplification is distributed denial-of-service attacks [79]. However, these imply a higher costs on the attacker side for programming and distributing viral software that subverts a suitable number of nodes into supporting the attack. Also conceivable is the use of a redirection-based flooding attack in combination with conventional flooding, for example as another layer of amplification in distributed denial of service. The threat of redirection-based flooding exists for any mechanism that enables a node to redirect packets from one IP address to another. This thesis focuses on IP mobility support, but its contributions are likewise applicable to mobility support in transport protocols and applications.

A widely accepted technique [94, 63, 88, 10, 45] to mitigate the threat of redirection-based flooding is for a correspondent node to check a mobile node's presence at a claimed new IP address by sending thereto an unpredictable token that the mobile node must echo back. The way that modern mobility protocols pusue such *reachability verification* is to drop packets destined to the new IP address until the verification completes. This has two fundamental disadvantages, however: First, the token exchange delays resumed bidirectional packet exchange by one extra round-trip time between the communicating nodes whenever the mobile node changes IP connectivity. This is additive to other handover-related delays and may therefore notably impair the quality of both hard- and soft-real-time applications. Second, the token exchange does not permit proactive mobility management, where the mobile node prepares a change in IP connectivity from its old point of attachment so that it can receive packets without delay once it arrives at the new point of attachment. While the performance of reactive mobility management is limited by a minimum IP layer handover delay of one round-trip time between the mobile node and the correspondent node, proactive mobility management permits handover delays to be reduced down to zero depending on the network topology.

The objectives of this thesis is hence to propose a secure mechanism that reduces, if not eliminates, the additional handover delay that modern mobility protocols cause during reachability verification. The mechanism should be end-to-end; it should avoid changes to existing network entities or the introduction of new entities, as both would be expensive to deploy in a single domain, and impossible to establish on a global scope in the short or medium term. The mechanism should also facilitate proactive mobility management. It should be generic enough to integrate smoothly into any mobility protocol without significant impacts on protocol design, and it should be easy to implement.

Related to mobility is multi-homing [37], which addresses the challenge of redirecting a packet flow between alternative IP addresses for the purpose of load balancing or fault tolerance. Mobility and multi-homing are in many aspects akin. In particular, multi-homing protocols face the same threats of illegitimate packet redirection that have been described above in the context of mobility. Some mobility protocols hence also support multi-homing [93, 124], which is why care was taken that the mechanism developed in this thesis also interoperates with multi-homing.

This thesis makes the following contributions:

1. An analysis of alternatives to the way modern mobility protocols pursue reachability verification, with respect to the objectives set forth above.

2. A mechanism for *concurrent* reachability verification, which permits bidirectional communications to resume in parallel with the token exchange. This meets the desired objectives and induces no handover delays on its own.

3. A set of standard-compliant optimizations to the Mobile IPv6 mobility protocol [63] that increase handover performance without compromising security, applicability, or deployability.

4. Modifications to Mobile IPv6 route optimization to facilitate concurrent reachability verification. These modifications are not standard-compliant.

5. Extensions to Mobile IPv6 route optimization that permit proactive mobility management.

6. Methods of integrating Mobile IPv6 with other optimized protocols on a mobile node to realize efficient reactive as well as proactive mobility management. These methods synchronize mobility-related activities across the mobile node's network protocol stack, and—in the case of proactive mobility management—provide the mobile node with the information necessary to generate a new IP address in advance.

7. Patches to the Kame-Shisa [119] Mobile IPv6 implementation for the FreeBSD operating system for all of the aforementioned optimizations, including support for concurrent reachability verification and proactive mobility management.

8. A thorough evaluation of the performance of concurrent reachability verification in Mobile IPv6 route optimization, demonstrating the benefits that can be obtained from it. Both optimized reactive and proactive mobility management are compared to standard Mobile IPv6 and Mobile IPv6 with standard-compliant optimizations. The measurements were obtained in an experimental testbed.

The primary reason for choosing Mobile IPv6 route optimization as the mobility protocol into which to integrate concurrent reachability verification was that the integration in this case requires modifications to the standard protocol signaling. Other mobility protocols typically interoperate with concurrent reachability verification without signaling changes.

The remainder of this thesis is organized as follows: Chapter 2 provides the background information that is necessary to understand the thesis. Chapter 3 starts off with a survey on related work in the area of optimized mobility support, and it proceeds with an analysis on how significant the threat of redirection-based flooding is compared to other flooding possibilities that traditionally exist in the Internet. It explains why the standard way of protecting against these attacks through reachability verification is insufficient. A number of alternative protection mechanisms are evaluated, and concurrent reachability verification is found to be the most promising among them. This provides the desired level of efficiency, but it requires an auxiliary mechanism to prevent misuse until reachability verification completes. The design of such a mechanism is the topic of chapter 4.

Chapter 5 demonstrates the integration of concurrent reachability verification into Mobile IPv6 route optimization. It is shown how the Mobile IPv6 optimizations can be used for efficient reactive as well as proactive mobility management. The optimizations have been implemented and evaluated in an experimental testbed. Chapter 6 explains measurements obtained for Internet telephony and file transfer applications. Again, reactive and proactive mobility management are both considered. The results evidence a strong performance advantage of optimized Mobile IPv6 compared to standard Mobile IPv6. Chapter 7 finally summarizes the contributions of this thesis and identifies opportunities for further research.

# 2. Fundamentals

This chapter creates an understanding of the fundamentals that are essential in understanding the rest of this thesis. Section 2.1 explains the Internet architecture and some early design decisions that today make IP mobility support very difficult. The difficulties are related to the way addressing works on the Internet, which is explained in section 2.2. Section 2.3 describes different ways of augmenting the Internet for mobility support a posteriori, as well as the security threats that this brings about. The threats limit the solution space for mobility protocol optimizations and are hence important to understand in view of the remainder of this thesis. The section also focuses more specifically on Mobile IPv6, today's standard mobility protocol for the next-generation Internet Protocol, version 6. It also introduces multi-homing. Section 2.4 gives an overview of other protocols that either play a fundamental role in IP mobility management, or are optional, but important to reach the desired level of efficiency. Many aspects of the performance evaluation at the end of this thesis require a firm understanding of TCP, which hence is provided in section 2.5.

## 2.1 An Internet for Stationary Nodes

The Internet as we know it originally descended from the "Arpanet", a network built as of the late 1960s to connect universities and government institutions for research purposes, enabling remote sites to quickly access information from each other and execute programs on distant computers. The Internet has since evolved into the most important communications medium besides the telephone network, serving more than a billion users today [122]. The Internet is a collection of *routers*, interconnected by *links*, that forward data from its source to the destination. Computers, or *nodes*, exchange data across the Internet by splitting the data into portions, so-called *packets*, that are each forwarded separately by routers.

To enable communications between nodes from different providers, nodes on the Internet conform to a set of standardized communication protocols. These protocols are arranged in layers, where each layer provides an increasingly sophisticated service based on the functionality of lower layers. Figure 2.1 shows the path of a packet

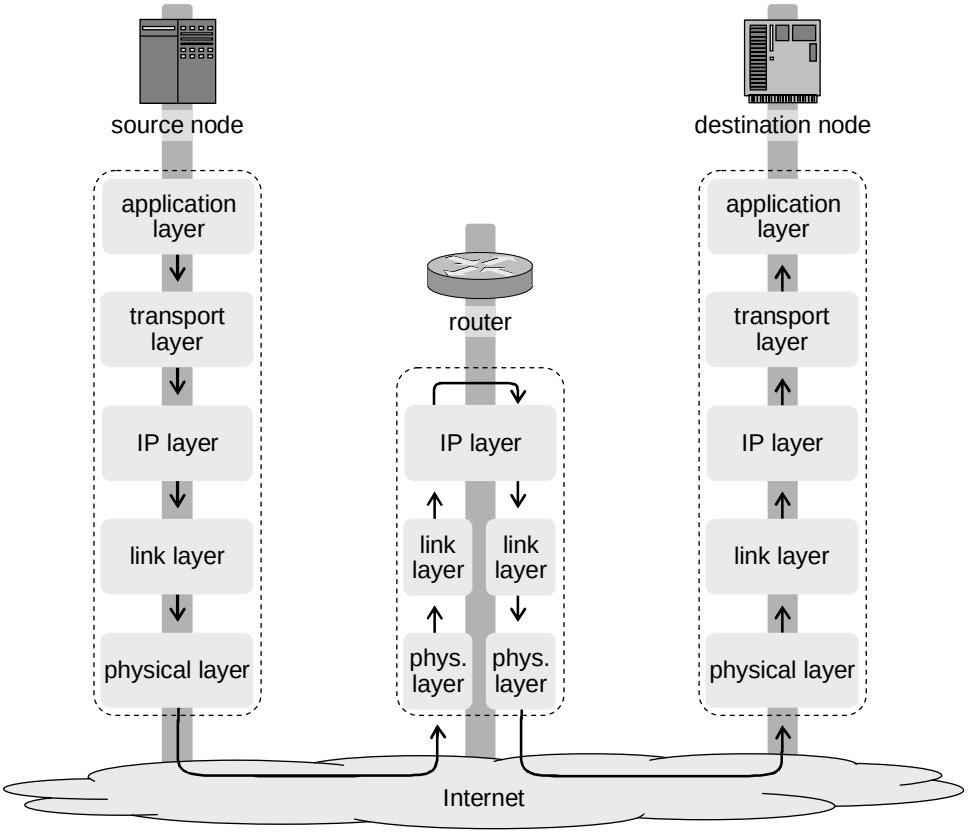



Figure 2.1: Protocol layering in the Internet

through this protocol architecture from its source node on the left to the destination node on the right. The *physical layer* defines the properties of the medium that constitutes a link. The actual transmission and reception of packets on a particular link is handled at the *link layer*. Link layer protocols are technology-specific, and without the help of upper layers permit communications only between nodes that attach to the same link. Communications beyond the local link, by help of routers, is provided by the *Internet Protocol, IP* [106], which resides at the *IP layer* above. IP harmonizes different technologies underneath and enables nodes in one link to send packets to nodes on a different link. Routers understand IP, and if they receive a packet that is destined to a node off the link on which it was sent, forward the packet onto a link closer to that node. The source and destination node of a packet furthermore pass the packet through a *transport layer*. This provides a certain level of abstraction from the peculiarities of Internet communications such as occasional packet loss, and possibly protects the Internet from becoming overloaded. One transport protocol, the *Transmission Control Protocol, TCP* [107], allows programs at the *application layer* to communicate bidirectionally without worrying about packetization, packet loss, or transmission rates tolerable by the network, but still high enough not to waste bandwidth. The *User Datagram Protocol, UDP* [105], was designed as an alternative transport protocol for those applications that do not require the services provided by TCP.

The fateful conclusion in the design of this protocol suite was that, since nodes were stationary, it would be feasible to identify them by their topological location in the Internet. This overloading of identity and location avoided the extra complexity that

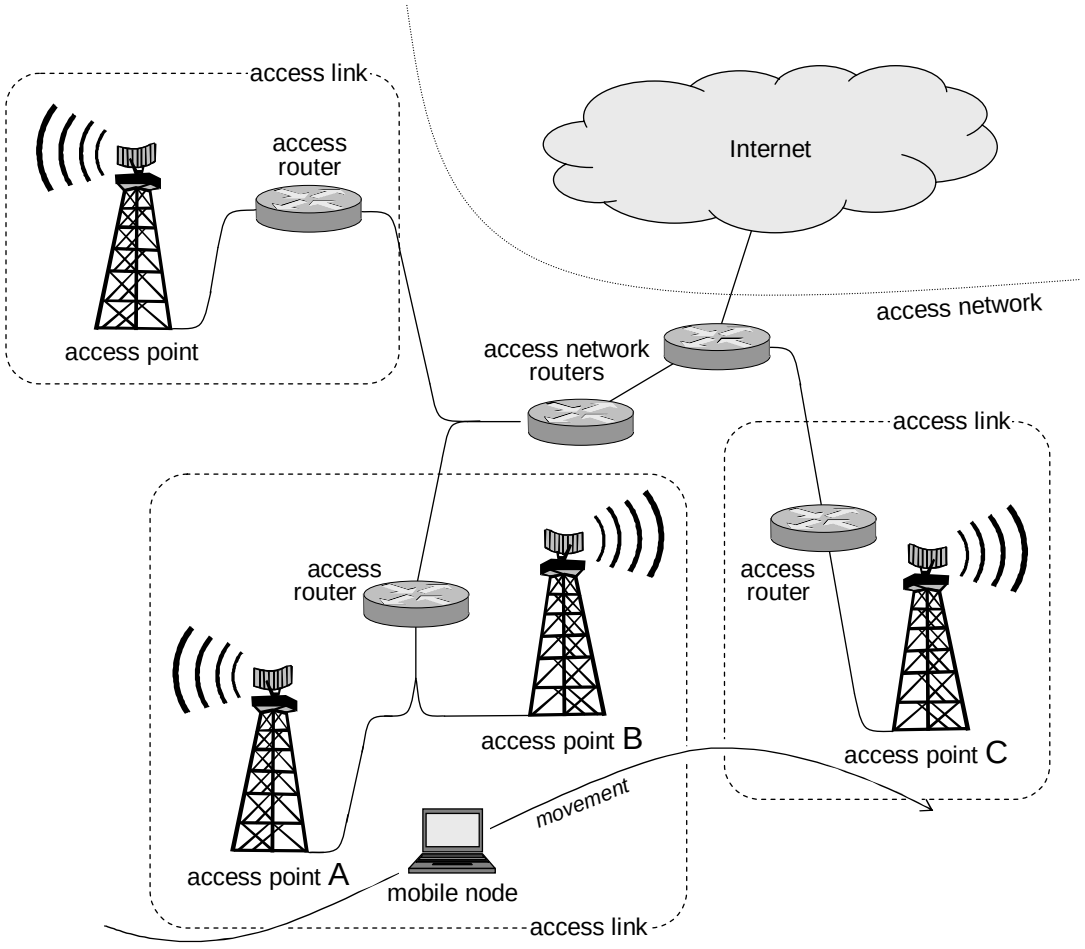



Figure 2.2: Evolved Internet architecture

an additional level of indirection would bring about, both implementation-wise and in terms of network administration. However, user behavior and expectations have changed over the years. While the early Internet was used mainly for electronic mail, file transfer, remote login, and later also Web browsing, today's users increasingly desire soft-real-time applications, like audio and video streaming or broadcasts [46], as well as hard-real-time services such as Internet telephony or videoconferencing [81, 96]. Such services are of particular benefit if available anytime, anywhere, and via any technology, and modern users increasingly access them on the go through wireless technologies. These developments call for an efficient mobility support, enabling a mobile node to continue active communication sessions while moving between different points of Internet attachment. The Internet architecture does not natively support this, however. Instead, a new location implies a new "identity" and moving there breaks any active communication sessions. This may be acceptable for short, transaction-based sessions, such as Web browsing or electronic mail, that could easily be retried in case of failure. But it defeats any meaningful use of longer sessions, like the aforementioned real-time services.

Figure 2.2 shows an example scenario of the evolved Internet architecture. Internet connectivity is provided by an *access network,* a collection of *access network routers* and interconnecting links. Some of the links are *access links* to which nodes can directly attach. The access link for a stationary node may be a single physical wire,

such as an Ethernet cable or a DSL line. But the node in figure 2.2 is mobile, so it associates with an *access point* via a wireless access technology, and the access point translates between the wireless medium and the fixed network in a manner invisible on the IP layer. An access link may include more than one access point for extended geographical coverage. The first access network router for packets that the node sends to destinations off-link is an *access router*. The trajectory of the mobile node shown in the figure passes three access points. Access points A and B belong to the same access link; they can directly communicate without an intermediate router. A switch between access points A and B is called a *link layer handover*. It occurs unnoticed by the IP layer and normally does not force active sessions to abort. On the other hand, access points B and C belong to different links. A switch between them causes the mobile node to change its location in the Internet topology and therefore forces active sessions to abort. This switch is called an *IP layer handover* or, for brevity, simply a *handover*. An IP layer handover includes a link layer handover.

The first IP version was number 4, *IPv4*; smaller version numbers were either reserved or unused. IPv4 has recently been accompanied by its prospective successor, the *Internet Protocol, version 6, IPv6* [38]. The major innovation of IPv6 is that it accommodates a much larger population of Internet nodes. Minor advancements include a revised, now modular packet structure and rebuilds of several auxiliary IP layer protocols, applying some of the lessons learned with IPv4. Yet the fundamental concepts of IPv4 still hold for IPv6, and IPv6 has the same deficiencies with respect to mobility support as IPv4. This thesis focuses on IPv6 due to its expected importance in future, so any mentioning of "IP" should be considered to mean "IPv6" unless otherwise noted.

## 2.2 Addressing and Routing

To allow two nodes on the Internet to communicate with each other, the nodes must be able to identify and address each other without ambiguity. The link layer and the IP layer use different addressing schemes for this purpose. These and their mapping are described in the following.

### 2.2.1 Network Attachment and Link Layer Addressing

The device that a node uses to attach to an access link is called a *network interface,* or simply *interface,* and the purpose of a *link layer address* is to uniquely identify an interface within the scope of the access link. Nodes that attach to the same link can directly communicate via their link layer addresses. Figure 2.3 illustrates a node with three built-in interfaces, two Ethernet interfaces and one wireless interface, and a node with a single Ethernet interface. The former node's interfaces have link layer addresses A, B, and C, respectively, and the latter node's interface has link layer address D. The nodes can directly communicate using link layer addresses C and D.

Link layer addressing is specific to the access technology that an interface supports, so two interfaces may use link layer addresses from different address families. Two prevalent types of link layer addresses are IEEE 802 MAC addresses and IEEE EUI-64 identifiers, 48-bit and 64-bit namespaces, respectively, that are administered by the *Institute of Electrical and Electronics Engineers, IEEE.* An *IEEE 802 MAC*

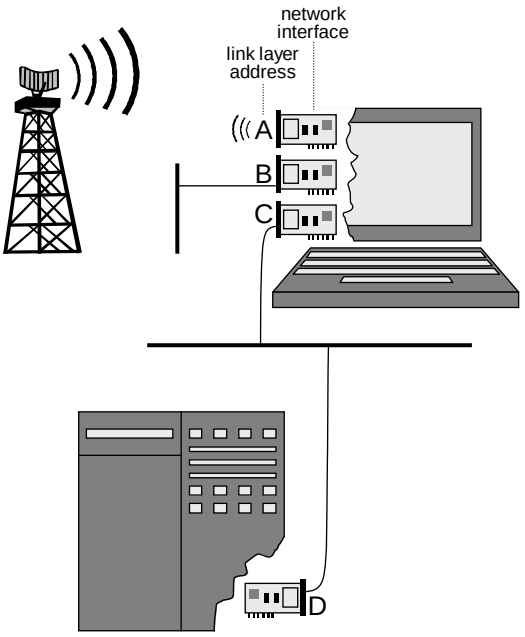
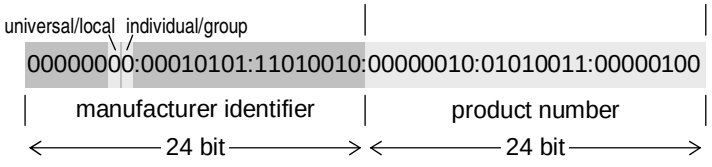


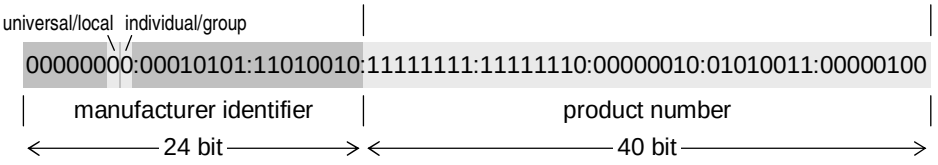

Figure 2.3: Network interfaces and link layer addresses

*address* is the concatenation of a 24-bit identifier of the interface manufacturer, and a 24-bit manufacturer-allocated product number that uniquely identifies the interface within the set of interfaces produced by the same manufacturer. Figure 2.4(a) depicts this composite structure. Part of the manufacturer identifier are two bits with special meaning: The *universal/local bit* marks a link layer address as globally unique if set to "0", or as locally unique within the scope of an access link if set to "1". The *individual/group bit* is set to "0" if the link layer address belongs to a single interface. It is set to "1" to form a *multicast link layer address*, which may be shared by multiple interfaces on the same link.

An *IEEE EUI-64 identifier* also begins with a 24-bit manufacturer identifier, but the product number is extended to 40 bits. The semantics of the universal/local and individual/group bits in an IEEE EUI-64 identifier are the same as those for IEEE

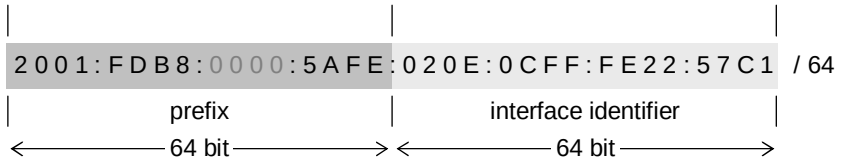



(a) IEEE 802 MAC address

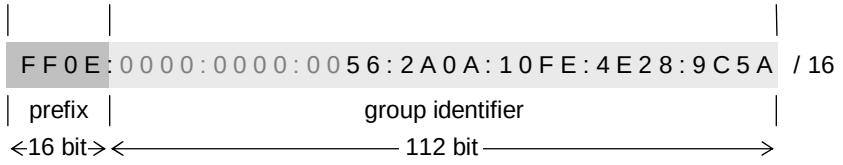



(b) IEEE EUI-64 identifier

Figure 2.4: Structure of IEEE 802 MAC address and EUI-64 identifier




(a) Unicast IP address




(b) Multicast IP address

Figure 2.5: Structure of unicast and multicast IP addresses

802 MAC addresses. An IEEE 802 MAC address can be transferred [57] into an IEEE EUI-64 identifier by inserting the bit string `FFFE`, given in hexadecimal form here, between the manufacturer identifier and the product number. Figure 2.4(b) shows the structure of an IEEE EUI-64 identifier that was derived from the IEEE 802 MAC address in figure 2.4(a).

### 2.2.2 IP Layer Addressing

IP addresses are 128-bit numbers that identify a single network interface or a group of interfaces. In this regard, IP addresses are very similar to link layer addresses. But since IP addresses are used for communications beyond one link, they need to be technology-independent and more sophisticated in structure so as to support a node in forwarding a packet towards a destination node. Figure 2.5(a) shows the basic IP address structure based on an example. The leading bits of the IP address, 64 bits in the example, form the *IP address prefix*. This specifies the type of IP address and, for some types, also encodes the location of the IP address owner in Internet topology. The remaining bits identify the interface or the group of interfaces to which the IP address belongs. For a given packet that is sent across the Internet, the IP addresses of the source and destination nodes are called *IP source address* and *IP destination address,* respectively.

Figure 2.5(a) also demonstrates the textual representation of an IP address. The 128 bits are given in hexadecimal form, with a colon between consecutive blocks of 16 bits. This representation can be simplified by replacing one sequence of zero bits by a double-colon ("`::`"). The length of the IP address prefix is denoted as a decimal number that is separated from the IP address by a slash. The prefix itself can then be referred to by setting all non-prefix bits to zero in this notation, that is, the prefix of the IP address in figure 2.5(a) is `2001:FDB8:0000:5AFE::/64`.

#### Unicast IP Addresses

Applications that communicate with a single remote node—such as Web browsing, electronic mail, or conventional Internet telephony—need to address exactly one network interface. This is the purpose of a unicast IP address. A *unicast IP address*

identifies the link to which it belongs, thereby localizing the IP address owner in Internet topology, and it specifies a particular interface attached to that link.

Link identification is the purpose of the IP address prefix, which in the case of a unicast IP address is 64 bits long. Each link on the Internet is assigned one or multiple *subnet prefixes*, globally unique numbers of 64 bits length, which nodes that attach to a particular link reuse as a prefix for their unicast IP addresses. The remaining 64 bits in a unicast IP address form an *interface identifier*, which is unique within the scope of a link and thus distinguishes the interface from others on the same link. The IP address shown in figure 2.5(a) is a unicast IP address.

For administrative purposes, the scope of a unicast IP address can be limited to the link to which the respective interface attaches. Such a *link-local IP address* can only be used for packet exchange between nodes on the same link. The subnet prefix of a link-local unicast IP address is set to the prefix `FE80:0000:0000:0000::/64`, and it does not bear any localization semantics. Link-local IP addresses are primarily used during auto-configuration when a node gains IP connectivity. To differentiate IP addresses with global scope from link-local IP addresses, the former are also referred to as *global IP addresses.*

As with link layer addresses, IP addresses include a universal/local bit and an individual/group bit, with the distinction that the meaning of the former is reversed to match the abovementioned subnet prefix for link-local unicast IP addresses. These special bits are not explicitly highlighted in figure 2.5, however, due to more simple, hexadecimal format used in the figure.

#### Multicast IP Addresses

A *multicast IP address* identifies a group of network interfaces of typically different nodes. Since the interfaces may attach to multiple links, there is no single subnet prefix that a multicast IP address could use. The interface group is instead solely identified by a 112-bit *group identifier*, which is appended to a 16-bit prefix to form a multicast IP address. The semantics of the universal/local and individual/group bits are the same for unicast and multicast IP addresses. Figure 2.5(b) displays an example multicast IP address.

Certain auto-configuration tasks require a node to send a packet to a neighbor of which it does not know the link layer address. The packet can in those cases be sent to a so-called *solicited-node multicast IP address*. This special multicast IP address is derived from the unicast IP address of interest, and it addresses all nodes on a link that use an IP address with a particular pattern in the last 24 bits. The packet is in this case sent to a multicast link layer address, so the sender does not require knowledge of the recipient's actual unicast link layer address. Given a unicast IP address, the corresponding solicited-node multicast IP address is formed by taking the lower 24 bits of the unicast IP address and prepending to this the prefix `FF02:0:0:0:0:1:FF00::/64`.

### 2.2.3 IP Headers and Extension Headers

For a packet to be forwarded across the Internet, routers must be able to determine the packet's destination node. Moreover, when the destination node eventually receives the packet, it should be able to identify the source node that originated the

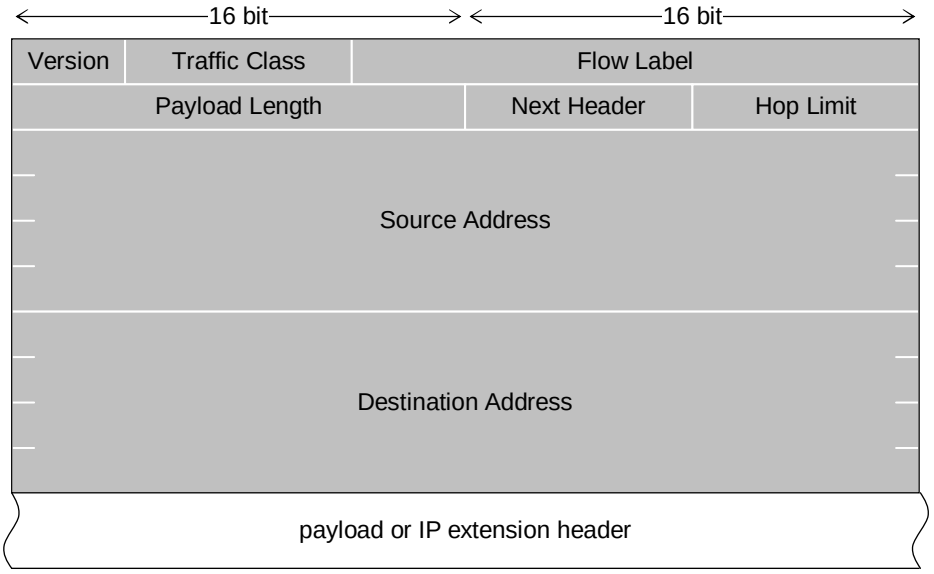



Figure 2.6: Format of IPv6 header

packet. Packets carry this and other related information in an *IP header*, a 40-byte data structure, depicted in figure 2.6, that is prepended to the packet's *payload* generated at the source node's transport layer. Routing is primarily based on the IP destination address in an IP header's Destination Address field, and a Source Address field holds the packet's IP source address.

The remaining fields in the IP header either define how routers should handle the packet, or they facilitate packet parsing. By means of the Hop Limit field, the source node of the packet can define how often the packet may be forwarded at most. A router decrements a non-zero value in the Hop Limit field before forwarding the packet, or drops the packet altogether if the value is already zero. The source node or a router may further use the Flow Label and Traffic Class fields, respectively, to classify the packet for special quality-of-service treatment. The value in the Next Header field specifies the type of payload that follows the header, and the Payload Length field indicates the length of that payload. The Version field contains a constant "6", indicating that the header format conforms to IPv6.

Some protocols, including mobility protocols, require packets to carry IP layer information or directives that go beyond what fits in an IP header. Such information can be transported in the following, optional *IP extension headers* [38] that are inserted between the IP header itself and the payload.

- Various auxiliary IP layer information can be included in a *Destination Options extension header*. This header carries one or more options to be processed by a destination node.
- A packet that is supposed to take a particular path to the destination node may include a *Routing extension header* to specify a sequence of intermediate IP destination addresses that the packet should traverse. These IP destination addresses may belong to different nodes, but as will be explained later on in this chapter, a mobility protocol may also use a Routing extension header to forward a packet virtually between different IP addresses of the same node.

- Packets can be authenticated, integrity protected, and encrypted at the IP layer through the *IP Security* protocol, *IPsec* [67]. Depending on the IPsec mode, a protected packet includes either an *Authentication extension header* or an *Encapsulating Security Payload extension header*.

- Fragmented packets include a *Fragmentation extension header* to aid reassembly at the destination node.

- A packet that requires special handling on each router along its path may carry a *Hop-by-Hop Options extension header* including one or more options that each router should process.

### 2.2.4 Routing

The process of forwarding a packet across the Internet from the source node to the destination node is called *routing*. Routing proceeds in a hop-by-hop manner, where each node on the packet's path from the source node to the destination node independently determines a neighbor closer to the destination node and hands the packet on to that neighbor. The routing process completes when that neighbor happens to be the destination node itself.

Routers maintain a *routing table* that maps a packet's IP destination address onto the IP address of a neighboring router closer to the destination node, or to determine that the destination node is itself a neighbor. A routing table may be manually preconfigured into a router, but in general, routers participate in a routing protocol to exchange the topological information they need to build a routing table automatically. Given that the subnet prefix in an IP destination address already locates the destination node, it is typically sufficient for a router to perform a routing table look-up based on only the subnet prefix. The interface identifier of the IP destination address is needed only when the packet is delivered to the destination node in the final forwarding step.

### 2.2.5 IP Address Resolution

A node that wishes to transmit a packet to another node on the same link needs to know the recipient's link layer address. This link layer address is in many cases not directly available, however: The packet's IP header does not bear any link layer information. And routers also typically retrieve an IP address when looking up the next-hop router for a to-be-forwarded packet in their routing table. A sending node must therefore be able to map a given IP address onto the corresponding link layer address. This process is called *IP address resolution*.

IP address resolution is accomplished by the *Neighbor Discovery* protocol with the message exchange illustrated in figure 2.7. A resolving node generates a Neighbor Solicitation message that asks the owner of a specific IP address to return its link layer address. This message is sent to the solicited-node multicast IP address that corresponds to the IP address of interest. The node that uses the IP address of interest returns its link layer address in a Neighbor Advertisement message. Resolved IP addresses are stored in a *neighbor cache* along with the respective link layer address to avoid repeated resolution of the same IP address.

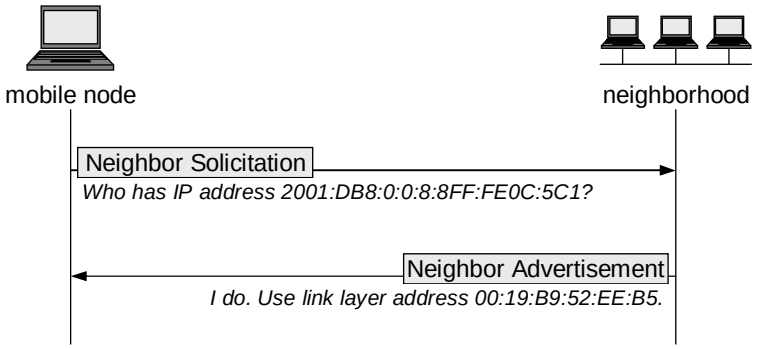



Figure 2.7: IP address resolution

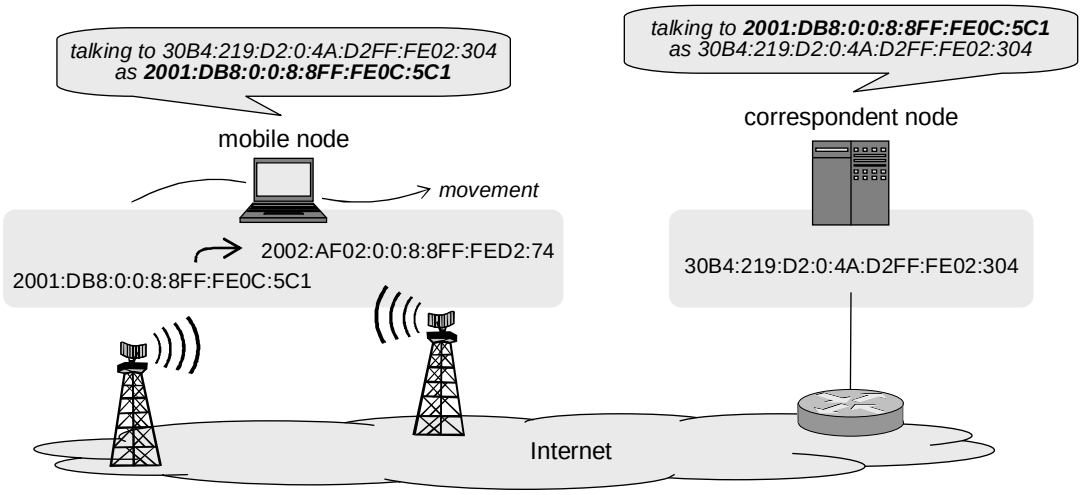



Figure 2.8: Impact of IP mobility on applications

## 2.3 Towards a Mobile Internet

As Internet users are becoming increasingly mobile, and delay- and loss-sensitive real-time applications are getting more and more popular, efficient mobility support is needed. The architecture of today's Internet fails to provide such support natively, however—both with IPv4 and with IPv6: When a mobile node moves, it configures a new IP address with a subnet prefix obtained of the new access link, but the new "identity" breaks active transport layer connections and applications. Figure 2.8 illustrates the adverse effect of mobility. It shows a mobile node that initiates a communication session with a correspondent node at the time it uses local IP address `2001:DB8:0:0:8:8FF:FE0C:5C1`. The application and transport protocol on both nodes thus uses this IP address to identify the mobile-node side of the session. At some point, the mobile node hands over to a different point of network attachment and replaces the existing IP address with `2002:AF02:0:0:8:8FF:FED2:74`. This causes the communication session to break because the application and transport protocol attempt to continue to use the mobile node's old IP address.

Different packet redirection techniques have been proposed to solve this addressing problem at the IP layer. Common to all of the mechanisms is that they mask changes in a mobile node's IP connectivity from protocols at layers above IP, as such called

*upper-layer protocols* in the following. To differentiate the packets generated by upper-layer protocols from signaling packets used by a mobility protocol, the former will henceforth be referred to as *payload packet.*

### 2.3.1 Packet Redirection Techniques

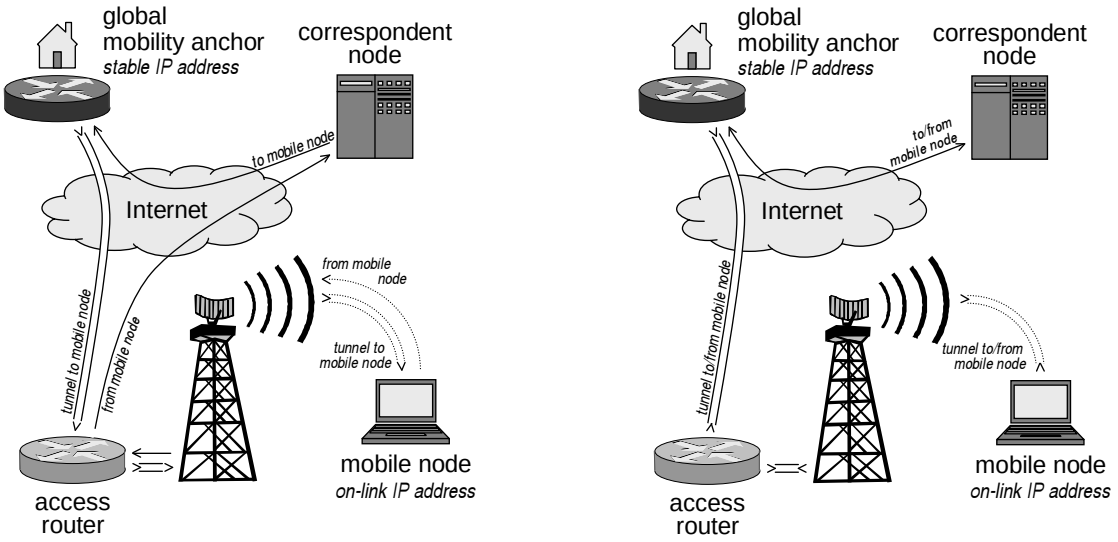
Mobility protocols differ in the way they change the routing of a mobile node's payload packets. Approaches can be classified into host routes, tunneling, and route optimization.

#### Host Routes

One approach towards supporting mobility is to use IP addresses only as node identifiers and to abandon the function of subnet prefixes as locators [110, 132]. A mobile node then has a *stable IP address* despite movements across different points of attachment. Since packets now can no longer be routed based on the subnet prefix alone, routers on the path from a correspondent node to the mobile node need to maintain one *host route* per mobile node in their routing table. An existing host route is replaced by a new one when the mobile node changes IP connectivity. Such a route update affects routers between the mobile node's new point of attachment and the correspondent node. It typically takes the form of a route update request message, which is generated either by the mobile node or by the mobile node's new access router, and which propagates in a hop-by-hop manner towards the correspondent node. Each router involved sets up a host route for the mobile node pointing towards the mobile node's new point of attachment. Where the old and the new host route merge in one router, the route update is complete because the part of the old host route between that router and the correspondent node is the same for the old and the new host route. That router then simply updates its existing host route to the new next-hop on the path towards the mobile node, and it does not propagate the route update request message further upstream. Existing state at routers that are not on the new host route times out eventually.

The participation of routers in mobility management renders host-route-based techniques little scalable and, hence, inappropriate for global use. Another issue with a global deployment of host routes is that it would require external help for correspondent nodes that wish to contact a mobile node when a host route to the mobile node is not yet in place. Host-route-based techniques were therefore proposed only for mobility management within one or a few access networks that would typically be administratively contiguous. The subnet prefix of a mobile node's stable IP address is selected such that it routes to an access network router that terminates all host routes to the mobile node. A packet destined to the stable IP address is then routed to that access network router based on its subnet prefix, and from there it gets forwarded along a host route towards the mobile node. Routers outside the access network can thus continue to rely on subnet prefixes for forwarding. Host routes within the access network are maintained proactively, such that mobile nodes are always reachable by correspondent nodes elsewhere on the Internet. Changes in a mobile node's point of attachment are invisible to correspondent nodes.

Due to the fundamental way in which host-route-based mobility approaches change classic routing as well as the scalability issues that go along with this, the techniques were never standardized. They eventually lost attention in the research community, mostly in favor of tunneling solutions.




(a) Triangular routing (b) Bidirectional tunneling

Figure 2.9: Mobility support through tunneling

### Tunneling

Tunneling provides an alternative to host routes that upholds the standard use of subnet prefixes for routing, while still providing mobile nodes with a stable IP address. The stable IP address in this case routes to a stationary proxy of the mobile node, its *global mobility anchor*. The global mobility anchor intercepts packets that correspondent nodes send to the stable IP address, encapsulates the packets, and forwards them through a tunnel to the IP address that the mobile node has temporarily configured while it stays at a particular access link. To differentiate the mobile node's configured temporary IP address from the stable IP address that locates the global mobility anchor, the former is also called the mobile node's current *on-link IP address.* The on-link IP address changes whenever the mobile node hands over to a different access link, and it is the mobile node's responsibility to inform its global mobility anchor about the new on-link IP address in such a case. The mapping between a mobile node and its current on-link IP address is called a *binding*; the process of establishing a new binding at a correspondent node or changing an existing one to a new on-link IP address is a *binding update*.

Early tunneling approaches were unidirectional from the global mobility anchor to the mobile node, and packets destined to a correspondent node were sent directly without tunneling. This is shown in figure 2.9(a), where the black router with the home symbol on top denotes the global mobility anchor at the mobile node's stable IP address. The asymmetric routing coined the term *triangular routing*. Triangular routing has substantial disadvantages [85], however. Most importantly, the technique requires a mobile node to use its stable IP address as an IP source address for the packets it sends. This is topologically incorrect because the subnet prefix of a stable IP address in general matches none of the subnet prefixes valid on a visited access link. Packets from the mobile node are hence at an increased risk of getting erased by IP source address filters [47], which many access network operators deploy to eliminate packets with IP source addresses that are—deliberately or

accidentally—invalid. Other problems with triangular routing are problems with IP multicast, as well as different hop counts on the forward and reverse paths that may cause packets from a mobile node to get dropped en route. In an effort to accommodate these difficulties, *bidirectional tunneling* has replaced triangular routing. This technique differs from triangular routing in that packets sourced at a mobile node are *reverse-tunneled* [85] to the global mobility anchor, which in turn decapsulates the packets and sends them, in topologically correct manner, from the mobile node's stable IP address to the respective correspondent node. Figure 2.9(b) illustrates this operation.

The advantage of tunneling in general is that it is compatible with the classic use of IP addresses in upper-layer protocols, so a correspondent node does not need to be mobility-aware. At the same time, the technique does not change the way in which packets are routed through the Internet. Tunneling hence scales well to global use across the Internet. Mobile IPv4 [99, 85] relies on triangular routing or bidirectional tunneling for this reason, and Mobile IPv6 uses bidirectional tunneling as a default. Bidirectional tunneling has further been adopted for localized mobility management [121, 50]. The disadvantage of tunneling is that it causes encapsulation overhead and increased packet propagation times. This is in particular an issue for hard-real-time applications such as Internet telephony or videoconferencing, which strongly depend on timely data delivery.

#### Route Optimization

Route optimization enables mobile and correspondent nodes to exchange packets along a direct path rather than to communicate via a global mobility anchor. This minimizes propagation latencies and packet overhead, and thus accommodates the growing popularity of real-time applications with strict delay requirements. Route optimization conceptually establishes a virtual tunnel directly between a pair of communicating mobile and correspondent nodes. Both end points maintain a binding that identifies the mobile node's current on-link IP address, and the mobile node is responsible for updating the binding at the correspondent node when its on-link IP address changes during a handover.

Upper-layer protocols may again identify the mobile node based on a stable IP address. This approach is followed by Mobile IPv6, where the stable IP address locates a roaming mobile node's global mobility anchor and can be used for both bidirectional tunneling or route optimization, depending on the availability of mobility support on the correspondent node. Route optimization per se does not require a global mobility anchor, so the mobile node may more generally be identified by a routable or non-routable *binding identifier*. The mobility extensions [93] to the *Host Identity Protocol* [88] are an example of a mobility protocol that uses pure route optimization. The binding identifier is in this case non-routable. It is cryptographically generated and bound to a public/private key pair of the mobile node, enabling the mobile node to authenticate securely to any correspondent node. The Host Identity Protocol requires cryptographic protection of all payload packets through IPsec.

Route optimization saves the additional overhead of packet encapsulation by rewriting the IP source and destination addresses in payload packets exchanged between mobile and correspondent nodes. Figure 2.10 illustrates this procedure for packets that the mobile node sends to the correspondent node. Upper-layer protocols on the

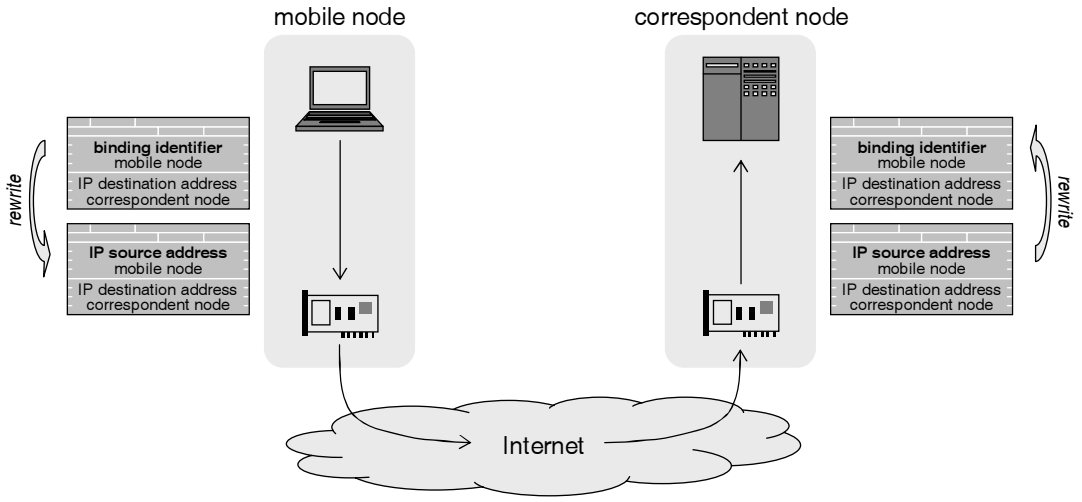



Figure 2.10: IP address rewriting in route optimization

mobile node use the binding identifier as an IP source address for outgoing packets, and the mobile node replaces this with its current on-link IP address when a packet reaches the IP layer. Packets received from the correspondent node include the current on-link IP address as the IP destination address. The mobile node overwrites this with its binding identifier before it hands the packet on to upper-layer protocols. On the correspondent node side, outgoing packets use the mobile node's binding identifier as an IP destination address when generated by upper-layer protocols, and the correspondent node overwrites this with the mobile node's current on-link IP address as the packets traverse its IP layer. The correspondent node further substitutes the mobile node's binding identifier for the on-link IP address when processing a packet received from the mobile node at the IP layer.

Mobility protocols with support for route optimization further include the binding identifier—or information sufficient to deduce the binding identifier—in route-optimized payload packets before sending the packets through the network. This enables both mobile and correspondent nodes to indicate whether an on-link IP address used in a payload packet is supposed to be replaced by the binding identifier. Payload packets that use the on-link IP address, but do not include the binding identifier or equivalent information, are exempt from mobility-specific processing at the IP layer. For example, route-optimized payload packets in Mobile IPv6 include the mobile node's stable IP address as a binding identifier within IPv6 extension headers. The mobility extensions for the Host Identity Protocol identify a binding based on a security parameter index of the mandatory IPsec security association between the end nodes. This, too, is contained in every payload packet exchanged.

Route optimization is a pure end-to-end mechanism and as such not suited for localized mobility management. Its appealing property of reduced propagation latencies, however, comes at the cost of more intractable security challenges. Mobile and correspondent nodes usually do not know each other a priori and do not pre-share credentials based on which mobility management could be secured. Classic means for authentication are hence not applicable. Tunneling approaches are typically easier to secure because a mobile node and its global mobility anchor are supposed to be from the same administrative domain and can hence be more practicably pre-

configured with the credentials required for mutual authentication. This and other security issues related to mobility in general and route optimization in particular are discussed in section 2.3.2. Another disadvantage of route optimization is that it requires mobility support on the correspondent node side and may hence not always be applicable. Correspondent node support may eventually become ubiquitous, however, given that it is a recommended feature for all IPv6 nodes as per the IPv6 Node Requirements RFC [73].

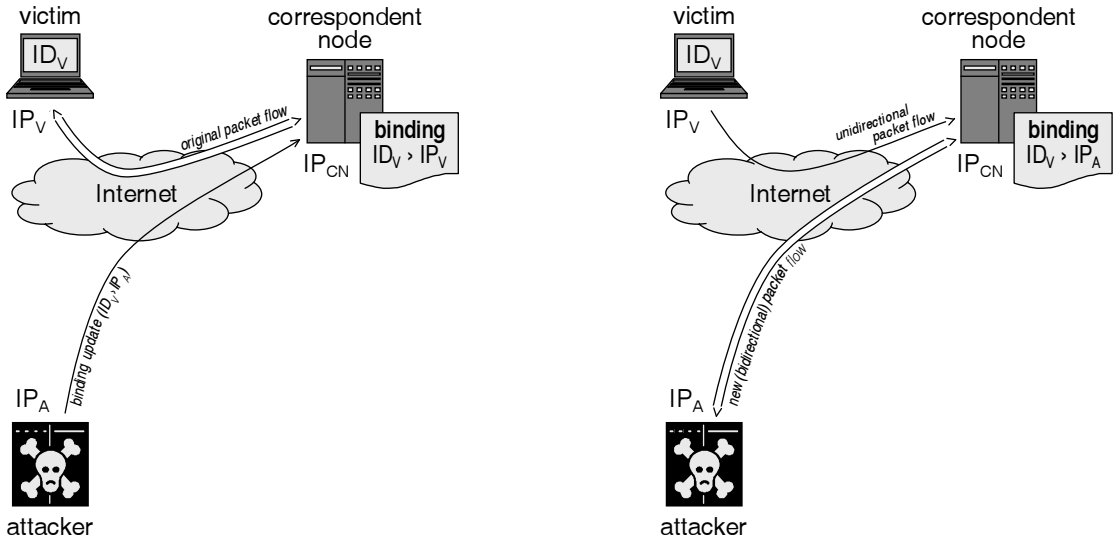
To enable correspondent nodes to contact a roaming mobile node, route optimization is typically supplemented with tunneling or some other mechanism that provides a stable IP address. Mobile IPv6, for instance, incorporates route optimization, but mobile nodes still continue to be reachable via a stable IP address and bidirectional tunneling. Route optimization is also used in the mobility extensions of the Host Identity Protocol, where the binding identifier is cryptographic and non-routable. Mobile nodes can here be reached by help of a stationary *rendezvous server*. This maintains a binding between a mobile node's binding identifier and current on-link IP address, and it can so serve as an indirection mechanism when a correspondent node contacts a mobile node.

### 2.3.2 Security Threats Related to IP Mobility

Any approach to support mobility at the IP layer modifies the traditional way in which packets are routed. This may invalidate assumptions that upper-layer protocols have on IP and thus lead to vulnerabilities to attacks that exploit the assumptions. These threats shape the security design of mobility protocols, and they limit the solution space for mobility protocol optimizations. A solid understanding of them is therefore necessary to assess such optimizations. The following is an overview of threats that are relevant to this thesis. For simplicity, but without loss of generality, the threats are explained from the perspective of route optimization. Similar threats apply when mobility is realized through tunneling or host routes, although they are typically harder to mitigate for route optimization since an a-priori security or trust relationship between an arbitrary pair of mobile and correspondent nodes in general does not exist. An exhaustive synopsis of mobility-related security threats is given in [94] in the context of Mobile IPv6.

#### Impersonation

A classic assumption of upper-layer protocols is that, when a peer is known to "own" a particular IP address, packets sent to that IP address will eventually reach the peer or they get lost in the network. This assumption is based on the generally reliable path selection in the Internet's routing infrastructure for a given packet's IP destination address. If an attacker was to change the way packets are routed, it would at least have to compromise a router on the legitimate path of the packets so as to divert the packets from there. On the other hand, IP mobility protocols establish a level of indirection between a mobile node's binding identifier at upper-layer protocols and the on-link IP address that locate the mobile node in the network. By way of mapping a binding identifier onto an on-link IP address, they thus introduce a mechanism through which the routing of packets can be changed at the edge of the Internet. Therefore, for a packet to eventually reach the intended destination, not only needs the routing infrastructure to be reliable, but also must the respective binding be legitimate.

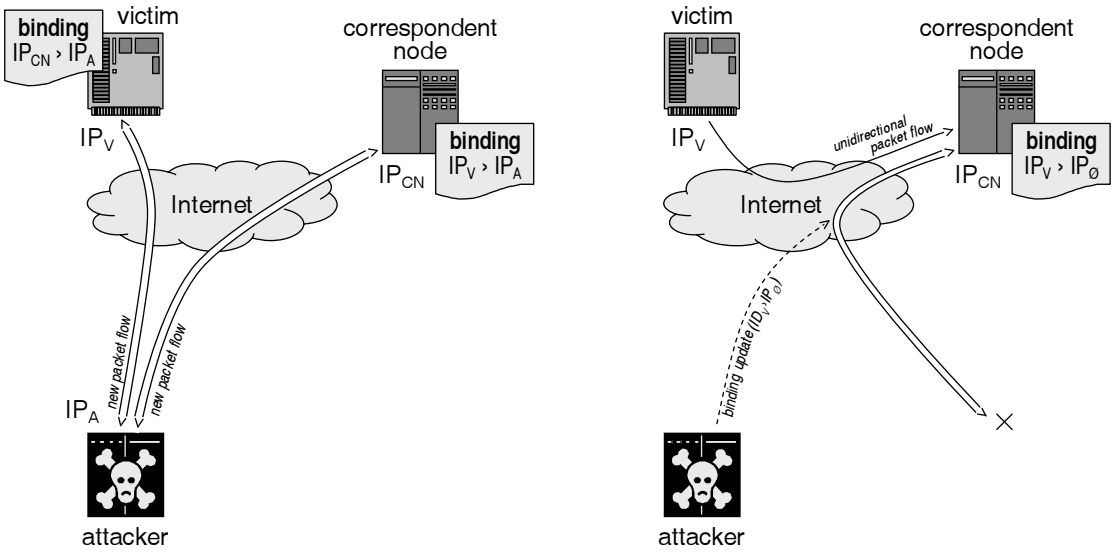



(a) Original packet flow (b) Redirected packet flow

Figure 2.11: Impersonation attack

A mobility protocol would allow an attacker to circumvent the reliable path selection of the Internet's routing infrastructure and launch an *impersonation attack* against a victim if it enabled the attacker to establish a binding at a correspondent node on behalf of the victim. In a typical impersonation attack, the attacker makes upper-layer protocols on the correspondent node side believe that they communicate with the victim, although they in fact communicate with the attacker. This attack is illustrated in figure 2.11. Part (a) shows the original packet flow between the victim's IP address, $IP_V$, and the IP address of a correspondent node, $IP_{CN}$. At some time, the attacker sends the correspondent node a binding update, tricking it into associating the victim's binding identifier, $ID_V$, with the attacker's on-link IP address, $IP_A$. Packets for the victim, which the correspondent node would normally send to $IP_V$, are now redirected to $IP_A$, as shown in part (b) of the figure. Vice versa, the correspondent node's IP layer modifies any packets received from $IP_A$ such that they appear to originate from $ID_V$ at upper-layer protocols. The victim may still send packets to the correspondent node, but any response from the correspondent node is directed to the attacker.

The victim shown in figure 2.11 is a mobile node, so it already has a binding identifier which the attacker can misuse. In general, the same type of attack is also possible against stationary victims, which do not participate in a mobility protocol. The attacker must then use its victim's on-link IP address as a binding identifier when spoofing a binding update for the correspondent node. This is particularly easy if the mobility protocol uses stable IP addresses as binding identifiers, such as Mobile IPv6, because binding identifiers and IP addresses are then from the same name space. The correspondent node would accept such a binding update even if it affects a stationary peer because it generally does not know whether a peer is mobile or stationary. Mobility protocols that do not allow regular IP addresses to be adopted as binding identifiers cannot be misused to impersonate stationary victims. The mobility extensions to the Host Identity Protocol belong to this class.




(a) Man-in-the-middle attack (b) Denial-of-service attack

Figure 2.12: Related impersonation attacks

The issue with impersonation is actually not new to the Internet. An attacker on the communication path between two nodes may already be able to eavesdrop on packets exchanged by the nodes, intercept the packets, and/or modify and eventually forward them. Mobility aggravates the problem in that an insecure mobility protocol may allow even an attacker *off* the communication path to do this, as it happens in figure 2.11. An attacker may also install a false binding before the victim starts using the attacked IP address.

The impersonation attack shown in figure 2.11 can be extended into a true man-in-the-middle attack, where the attacker performs a binding update both with the victim and with the correspondent node. (The roles of the victim and the correspondent node are then actually interchangeable.) Figure 2.12(a) illustrates this attack for the case where both the victim and the correspondent node are stationary. The attacker here uses the correspondent node's IP address, $\mathrm{IP_{CN}}$, as a binding identifier in the binding update for the victim, and it uses the victim's IP address, $\mathrm{IP_V}$, as a binding identifier in the binding update for the correspondent node. The on-link IP address in both binding updates is the attacker's own IP address, $\mathrm{IP_A}$. Packets that the victim and the correspondent node send are now all routed to the attacker. The attacker can read the packets, and possibly forward them to the original recipient, either modified or as is. The two binding updates are not shown in the figure.

An impersonation attack can further be used for denial of service. The purpose of a *denial-of-service attack* in general is to compromise a victim in terms of its ability to communicate and to respond to requests from legitimate peers. When this is accomplished through an impersonation attack, packets that are destined to the victim by some correspondent node are redirected to a random or non-existing IP address so that bidirectional communications become impossible between the victim and the correspondent node. Figure 2.12(b) illustrates this attack, again for the case where both the victim and the correspondent node are stationary.

### Redirection-based Flooding

Appropriate authentication of a mobile node's binding identifier can mitigate impersonation attacks. But a malicious, yet properly authenticated node may still register a false on-link IP address and so redirect packets, that would otherwise be delivered to the malicious node itself, to a third party. This introduces a vulnerability to a new type of denial-of-service attack, redirection-based flooding attacks. A *flooding attack* in general defeats a victim's ability to communicate by overloading the victim with a high number of needless packets. In the specific case of a *redirection-based flooding attack*, the flooding packets originate with one or multiple correspondent nodes of the attacker. The attacker is supposed to receive the packets itself, but it tricks the correspondent nodes, through misuse of a mobility protocol, into substituting the victim's on-link IP address for the binding identifier of the attacker. The packets thus get redirected to the victim. Figure 1.1(b) illustrates a redirection-based flooding attack where the attacker establishes TCP connections with two correspondent nodes for downloading some large data files, and then misuses the mobility protocol to make all correspondent nodes redirect outgoing packets to the IP address of the victim.

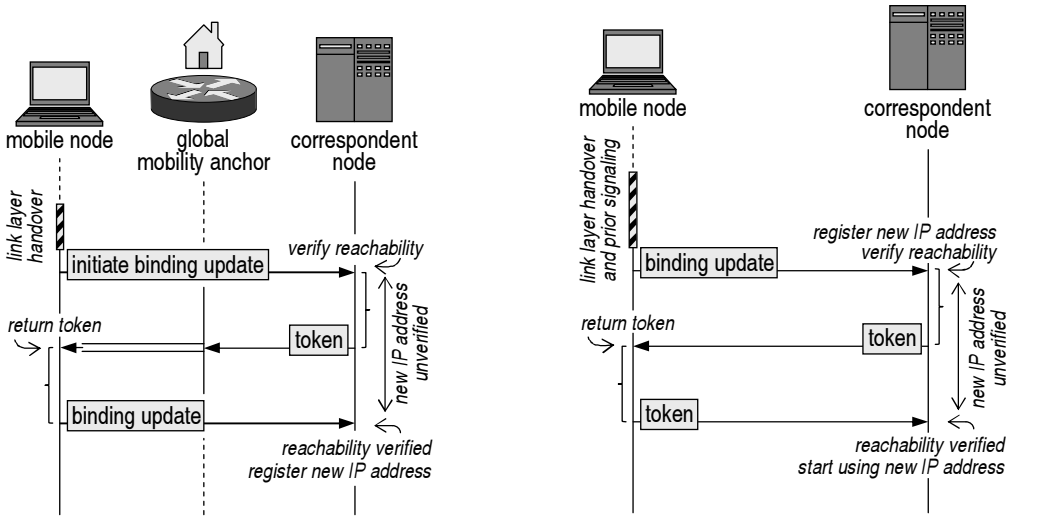
A redirection-based flooding attack may target an entire network, rather than a single node, either by loading the network with packets, or by overwhelming a router or other critical network device further upstream. The attacker then directs the flooding packets against an arbitrary IP address matching a subnet prefix of the victim network or, respectively, against the IP address of the network device in focus. An attack against a network potentially impacts a larger number of nodes than an attack against a specific node, although neighbors of a victim node on a broadcast link typically suffer the same damage as the victim itself.

### Possible Solutions

Impersonation attacks can be prevented by authenticating a mobile node to the correspondent node during a binding update, and at the same time verifying that the mobile node is in fact authorized to add and remove an on-link IP address for the claimed binding identifier. One way to authenticate the mobile node is cryptographically based on secret credentials, preconfigured on both nodes [103]. Pair-wise preconfiguration is relatively straightforward, but it is inappropriate for global use since mobile and correspondent nodes may communicate without prior contact. The technique also causes significant administrative overhead.

Authentication based on asymmetric public-key cryptography does not require pair-wise preconfiguration and hence involves substantially less administrative labor. Here, the correspondent node tests the mobile node's knowledge of the private component of a public/private-key pair given only the public key. The mapping between the mobile node's binding identifier and its public key, in turn, is attested by a *certificate*, issued by a *public-key infrastructure* that both end nodes trust. However, large-scale use of a public-key infrastructure for global mobility management is considered unsuitable for multiple reasons [136], foremost due to scalability and performance concerns.

A third technique to protect against impersonation attacks is to tie a mobile node's binding identifier cryptographically to the public component of the mobile node's




(a) Preventing impersonation (b) Preventing redirection-based flooding

Figure 2.13: Reachability verification for (a) stable and (b) on-link IP address

public/private-key pair, for example, by taking bits for the binding identifier from the output of a hash on the public key [84]. The mapping between the binding identifier and the public key is then verifiable without a public-key infrastructure. This technique is used in Enhanced Route Optimization for Mobile IPv6 [11] as well as in the mobility extensions of the Host Identity Protocol.

Mobility protocols that use a stable IP address as a binding identifier, such as Mobile IPv6, can further verify the mobile node's reachability at the stable IP address in order to validate that the mobile node really owns the binding identifier. *Reachability verification* in general is realized through the exchange of an unpredictable piece of information, which the correspondent node sends to the to-be-verified IP address, and which the node claiming to be reachable at that IP address echoes back. The new on-link IP address is called *unverified* until the correspondent node has received the token back from the mobile node, and it is called *verified* afterwards.

Figure 2.13(a) illustrates how reachability verification can be used for authenticating a mobile node. After the link layer handover has completed, the mobile node initiates the binding update by sending the correspondent node a message. To be able to authenticate the mobile node, the correspondent node sends an unpredictable token to the mobile node's stable IP address. The global mobility anchor of the mobile node receives this token and tunnels it to the mobile node. The mobile node can then use the token to send an authenticated message requesting a binding update. It should be noted that the message exchange depicted in figure 2.13(a) is conceptual and abstracts from some of the details of a typical mobility protocol.

Reachability-based authentication is less secure than cryptographic authentication because it does not prevent impersonation attacks initiated from a position on the path from the correspondent node to the stable IP address. An attacker in such a position could always pass reachability verification for the stable IP address, so it could use the IP address as a binding identifier in a spoofed binding update for the correspondent node. Reachability verification further induces potentially long handover delays due to the end-to-end message exchange. Yet, the appealing prop-

erty of reachability-based authentication is that it works without preconfiguration of shared secret credentials and also without a public-key infrastructure.

Redirection-based flooding attacks, too, can be protected against by reachability verification. In this case, it is the mobile node's on-link IP address that is subject to verification rather than the stable IP address. The way reachability verification is used for this purpose by existing mobility protocols is that no payload packets are sent to the on-link IP address until it becomes verified. This conservative form is henceforth referred to as *standard reachability verification.* Both Mobile IPv6 and the mobility extensions for the Host Identity Protocol apply it.

Figure 2.13(b) illustrates standard reachability verification conceptually. At some point after the link layer handover is complete, the mobile node sends the correspondent node a message to initiate the binding update. The message that is shown in the figure is assumed to already be authenticated, hence some signaling might precede the message to facilitate the authentication, such as the message exchange shown in figure 2.13(a). When the correspondent node processes the message, it registers the mobile node's new on-link IP address and labels it "unverified" for the time being. The correspondent node then sends an unpredictable token to the new on-link IP address, but it refrains from sending any payload packets yet. The mobile node returns the token to the correspondent node once it receives it. The correspondent node, in turn, considers the new on-link IP address legitimate when it receives the token and relabels the IP address "verified". At that point, the correspondent node begins sending payload packets to the new on-link IP address. The mobile node can send payload packets from the new on-link IP address right after initiating the binding update; it does not need to wait for the reachability verification to complete.

The aforementioned vulnerability of reachability verification to attackers on the path from a correspondent node to the IP address being verified is acceptable in the case of flooding attack prevention for three reasons:

1. An attacker on the path towards a victim can trick a correspondent node into sending packets to the victim already in the non-mobile Internet of today.
2. The set of correspondent nodes that an on-path attacker can potentially induce to send packets to the victim is confined to upstream nodes.
3. The attraction of an on-path attack is limited given that the flooding packets also pass the attacker.

Some transport protocols already pursue reachability verification during connection establishment. A TCP responder, for example, chooses a random 32-bit sequence number and sends it to the initiator's claimed IP address. The initiator must send the sequence number back to the responder, allowing the latter to verify the former's reachability.[1] IP mobility protocols defeat the purpose of such transport layer reachability verification because they introduce the ability to redirect packets after connection establishment. An additional reachability verification must hence be pursued at the IP layer whenever a mobile node changes its on-link IP address.

[1] Truly unpredictable sequence number selection was added to TCP a posteriori [18]. The original TCP specification suggested initializing sequence numbers from a monotonically increasing clock, thus allowing an attacker to guess an initial sequence number.

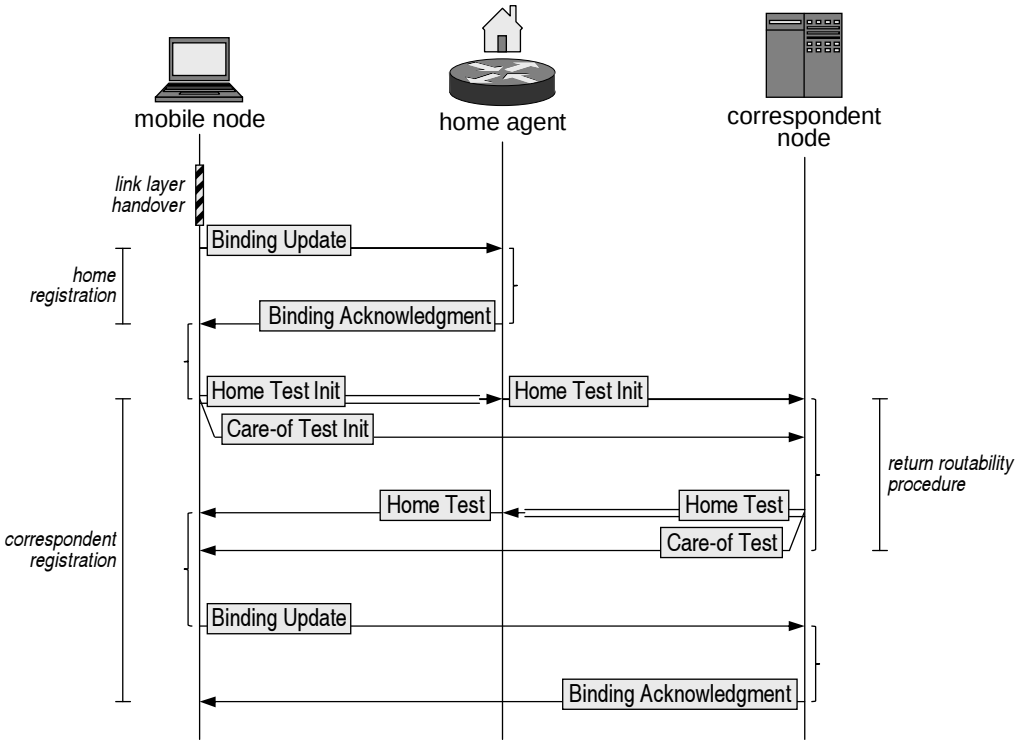



Figure 2.14: Home and correspondent registration in Mobile IPv6

### 2.3.3 Mobility Support in IPv6

*Mobility support in IPv6, Mobile IPv6* [63], is an IP layer mobility protocol providing permanent reachability and session continuity to mobile nodes. Mobile IPv6 uses bidirectional tunneling in combination with a stable IP address as a default packet redirection technique. It further offers route optimization for reduced packet propagation delays between a mobile node and a correspondent node that supports the optimization. A mobile node's stable IP address, called a *home address* in Mobile IPv6 terminology, has a subnet prefix from the mobile node's *home link*. The mobile node can directly communicate via its home address when it attaches to the home link. It operates like a stationary IPv6 node without mobility support in this case.

When the mobile node moves to a foreign access link, it becomes reachable at an on-link IP address, or *care-of address*, that differs from the home address. The mobile node then requests its global mobility anchor, a dedicated router on the home link called a *home agent*, to proxy the mobile node at the home address and provide bidirectional tunneling service. The mobile node initiates this *home registration* with a Binding Update message that it sends to the home agent, as shown in figure 2.14. The home agent creates a binding between the home address and the care-of address when it receives the Binding Update message. The home agent maintains a *binding cache* in which it stores information related to the binding, such as the home address, the care-of address, and the binding lifetime. The binding cache holds a separate entry for each registered binding. The home agent further sets up a bidirectional tunnel to the care-of address, through which the mobile node can continue to communicate via its home address from remote. The home agent finally assigns the binding a maximum lifetime and confirms the home registration with a Binding Acknowledgment message that includes the binding lifetime. The

mobile node must renew the home registration when the binding lifetime elapses. The Mobile IPv6 RFC does not limit the binding lifetime for home registrations. The mobile node establishes a tunnel from the care-of address to the home agent as part of the home registration. Information related to the home registration is stored in a new entry of the mobile node's *binding update list*. This includes the home and care-of addresses, the binding lifetime granted by the home agent, and any state related to pending retransmissions and re-registrations.

The Binding Update and Binding Acknowledgment messages exchanged during a home registration are authenticated and integrity-protected based on an IPsec security association, typically using the IPv6 Encapsulating Security Payload extension headers in transport mode. The credentials required to bootstrap the security associations may be preconfigured onto the mobile node and the home agent. The nodes are assumed to be administered by the same authority, so the preconfiguration should in general be feasible. Technically, IPsec protection alone cannot prevent redirection-based flooding attacks against a spoofed care-of address since it does not incorporate a verification of the mobile node's reachability at the care-of address. This makes home registrations in general misusable for flooding attacks. The security design of the home registration is instead founded on an assumed trust relationship between the mobile node and the correspondent node. This permits the home agent to accept a care-of address from the mobile node without previous reachability verification.

The mobile node retransmits any messages that it sends to the home agent or a correspondent node and for which it does not receive a response within appropriate time. The standard retransmission algorithm calls for the mobile node to wait 1 s before it initiates the first retransmission,[2] and to double the waiting time for each further retransmission. A maximum interval of 32 s is reached after six retransmissions. The mobile node may then continue retransmitting at the same rate.

It is the mobile node's responsibility to update a binding at the home agent whenever its care-of address changes due to a handover. The bidirectional tunnel is then moved to the new care-of address. Each such binding update requires another exchange of IPsec-protected Binding Update and Binding Acknowledgment messages. When the mobile node returns to its home link after a period of roaming, it initiates a *home deregistration*, requesting the home agent to discontinue proxy service, remove any existing binding for the mobile node, and tear down the bidirectional tunnel to the mobile node.

The increased packet propagation latencies that go along with bidirectional tunneling can be avoided if a correspondent node supports Mobile IPv6 route optimization. The mobile node initiates this through a *correspondent registration* with the correspondent node when it receives an encapsulated payload packet that originates from this correspondent node, and no recent indication exists that the correspondent node does not support route optimization. The purpose of the initial correspondent registration is to probe whether the correspondent node supports route optimization and, in case it does, have the correspondent node cache a binding between the

[2] An initial waiting time of 1.5 s is required before the mobile node retransmits a Binding Update message for its home agent if the home agent does not yet have a binding registered for the mobile node. The increased waiting time then allows the home agent to complete Duplicate Address Detection on the mobile node's home address before it sends a Binding Acknowledgment message.

mobile node's home and current care-of address. Figure 2.14 depicts the messages exchanged during a correspondent registration. As with the home registration, these include a Binding Update message and, in this case optionally, a Binding Acknowledgment message. The registrations differ in the way they secure this exchange, however. There is generally no a-priori security or trust relationship between an arbitrary pair of mobile and correspondent nodes based on which a correspondent registration could be protected. So to mitigate the threats described in section 2.3.2, Mobile IPv6 uses a *return routability procedure*. This is based on a verification of the mobile node's reachability at the home address and the care-of address. Reachability at both IP addresses is taken as an indication that the mobile node is the legitimate owner of these two IP addresses, and hence that it is secure to bind the IP addresses. The *home address test* weakly authenticates the mobile node because the home address identifies the mobile node at upper-layer protocols. The *care-of address test* validates the mobile node's liveliness at the claimed point of attachment, so it protects against redirection-based flooding attacks.

The mobile node initiates the home address test with a Home Test Init message, which it reverse-tunnels to the home agent, and which the home agent forwards from the home address to the correspondent node. The confidentiality of the Home Test Init message is protected inside the tunnel, typically using the IPv6 Encapsulating Security Payload extension header. But there is no cryptographic protection on the path between the home agent and the correspondent node. To provide some minimum level of security also on the latter path, the Home Test Init message includes a random *home init cookie* to be returned by the correspondent node. A returned cookie proves that the message was processed by a node on the path from the home agent to the correspondent node, thus protecting against off-path attackers that cannot see the Home Test Init message. This mechanism is ineffective against eavesdroppers on the path from the home agent to the correspondent node. Yet, the motivation here is that such attackers could anyway compromise communications between the mobile node and the correspondent node unless the packets are protected otherwise.

The correspondent node responds to the Home Test Init message with a Home Test message. This is directed to the mobile node's home address and hence again routes via home agent. The Home Test message contains a home keygen token, a home nonce index, and the home init cookie copied from the Home Test Init message. The mobile node considers the returned home init cookie a sufficient proof that the Home Test message was generated by the right correspondent node. The *home keygen token* is a keyed one-way hash on the concatenation of the mobile node's home address, a random *nonce*, and a "0" indicating that the token corresponds to a home address rather than a care-of address. The hash is keyed with a secret known only to the correspondent node. The mobile node uses the home keygen token to prove its reachability at the home address when it eventually sends a Binding Update message to the correspondent node. The *home nonce index* identifies the nonce based on which the correspondent node has computed the home keygen token. The mobile node includes the home nonce index in the Binding Update message to allow the correspondent node to reproduce the home keygen token without having to store it explicitly. This kind of resource preservation is a precaution against denial-of-service attacks that aim at exhausting the correspondent node's memory resources. The Home Test message is sent through the same authenticated and

encrypted tunnel between the home agent and the mobile node as the Home Test Init message.

The care-of address test consists of a pair of messages directly exchanged between the mobile node and the correspondent node, and it does not involve the mobile node's home agent. The mobile node initiates the care-of address test by sending a Care-of Test Init message to the correspondent node, as shown in figure 2.14. This typically happens in parallel with the initiation of the home address test. The Care-of Test Init message includes a random *care-of init cookie* that is to be returned by the correspondent node. This serves as a proof that the message was processed by a node on the path from the mobile node to the correspondent node.

The correspondent node sends a Care-of Test message directly to the mobile node's care-of address when it receives the Care-of Test Init message. The Care-of Test message contains a care-of keygen token, a care-of nonce index, and the care-of init cookie copied from the Care-of Test Init message. The *care-of keygen token* is a keyed one-way hash on the concatenation of the mobile node's care-of address, a random nonce, and a "1" indicating that the token corresponds to a care-of address. The hash is again keyed with the correspondent node's secret. The mobile node uses the care-of keygen token to prove its reachability at the care-of address. The *care-of nonce index* identifies the nonce based on which the correspondent node has computed the care-of keygen token. As with the home nonce index, this is communicated in the protocol so that the correspondent node can reproduce the care-of keygen token without having to explicitly store it.

The return routability procedure enables the mobile node to transact a binding update at the correspondent node. Specifically, the mobile node computes a *binding management key* as a one-way hash on the concatenation of the home keygen token and the care-of keygen token. Knowledge of the right binding management key hence proves the mobile node's reachability at the home address and at the care-of address. Based on the binding management key, the mobile node calculates a message authentication code for a Binding Update message that it subsequently sends to the correspondent node. The Binding Update message further includes the home and the care-of nonce indices received during the return routability procedure. The mobile node may request the correspondent node to return a Binding Acknowledgment message by raising a flag in the Binding Update message.

The home and care-of nonce indices included in the Binding Update message allow the correspondent node to reproduce the home and care-of keygen tokens, respectively, on receipt of the Binding Update message. The correspondent node can then calculate the binding management key in the same way as the mobile node, and use this to verify the message authentication code of the Binding Update message. If the message authentication code is correct, the correspondent node knows that the mobile node has received the Home Test and Care-of Test messages and, consequently, that the mobile node is reachable at the claimed home and care-of addresses. A correct message authentication code also indicates that the message has not been tampered with in flight. The correspondent node can hence assume with reasonable certainty that the mobile node is the legitimate owner of the home and care-of addresses included in the Binding Update message, and that it is secure to bind the two IP addresses. The correspondent node accordingly creates a new entry in its binding cache and begins route-optimizing payload packets. The binding cache of

a correspondent node is fundamentally the same as the binding cache of a home agent. Finally, if requested by the mobile node, the correspondent node confirms the successful correspondent registration with a Binding Acknowledgment message. This message also includes the binding lifetime granted by the correspondent node, and it is again authenticated with the binding management key. The Mobile IPv6 RFC limits the binding lifetime for correspondent registrations to seven minutes in an attempt to ward off time-shift attacks [94]. A mobile node that does not change IP connectivity for this period must refresh any active correspondent registrations. This requires another run of the return routability procedure. The mobile node stores information related to a correspondent registration in its binding update list.

A correspondent node that does not support Mobile IPv6 route optimization sends an ICMPv6 Parameter Problem message when it receives a Home Test Init, Care-of Test Init, or Binding Update message, indicating that it does not understand the payload in the received packets. The ICMPv6 Parameter Problem messages enable the mobile node to determine the lack of support on the correspondent node side. The mobile node caches such information in its binding update list to avoid repeated attempts to initiate route optimization with correspondent nodes that do not support it.

The mobile node updates the binding at each of its correspondent nodes when its care-of address changes during a handover. This involves another care-of address test. The home keygen token from the previous home address test may be reused if it has been obtained within the last 3.5 minutes. Otherwise, the token has passed its lifetime and the entire return routability procedure must be redone. The lifetime of a care-of keygen token is 3.5 minutes as well. But since a care-of keygen token is bound to the care-of address to which it was sent, it typically cannot be reused. The mobile node may reuse a previously obtained care-of keygen token only in the special case where it returns to a previously visited access link and configures an old care-of address again.

The mobile node initiates a *correspondent deregistration* when it eventually returns to its home link. The correspondent node then removes the existing binding for the mobile node, and subsequent payload packets are exchanged via the home address. The binding management key needed to authenticate the Binding Update and Binding Acknowledgment messages exchanged during the correspondent deregistration is a one-way hash on only a home keygen token. The return routability procedure hence reduces to a home address test in the case of a correspondent deregistration.

The Mobile IPv6 specification restricts the scheduling of home and correspondent registrations. Accordingly, the mobile node must send a Binding Update message to its home agent before it initiates the return routability procedure with a correspondent node. The mobile node must also wait for a positive Binding Acknowledgment message from the home agent before it sends a Binding Update message to a correspondent node. The same message order applies to home and correspondent deregistrations. The purpose of these rules is to establish the mobile node's ability to communicate via the home address before a binding for the home address can be cached at correspondent nodes.[3]

[3]While the a-priori transmission of a Binding Update message to the home agent helps the latter to forward home address test messages correctly, there is technically no reason for requiring the

Figure 2.14 illustrates a scenario where the mobile node communicates with only one correspondent node that supports Mobile IPv6 route optimization. The figure is therefore limited to a single correspondent registration. More generally, the mobile node may communicate with multiple correspondent nodes that support the optimization. It then pursues a correspondent registration with each of the correspondent nodes in parallel.

### 2.3.4 Reactive versus Proactive Mobility Management

Mobility support at the IP layer has traditionally been considered a response to link layer handover. Sophisticated protocol optimizations for such *reactive mobility management* can reduce handover delays considerably, but a minimum latency for payload packets to be redirected to the mobile node's new point of attachment is inherent in all reactive approaches: It always takes a one-way propagation time to register a new on-link IP address with a global or local mobility anchor, or with a correspondent node, plus another one-way propagation time for the first redirected payload packet to arrive at the new IP address. At the same time, packets in flight towards the old on-link IP address are lost.

*Proactive mobility management* can further reduce handover delays and handover-related packet loss. It requires a mobile node to anticipate changes in IP connectivity and, if a handover is imminent, obtain a new IP address from the target access link and register this with its global or local mobility anchor, or with its correspondent node, prior to initiating the link layer handover. While efficient reactive mobility management cannot go without link layer notifications that inform the IP layer about changes in link layer attachment, proactive mobility management requires bidirectional interaction between the link and IP layers. The mobility protocol must now be able to issue commands to the link layer and receive events as well as anticipatory information from the link layer. The mobile node must further be able to match link layer information from a discovered access link to subnet prefix information for that link. This typically requires a mapping between the link layer address of a discovered access point and the set of subnet prefixes in use on the link to which this access point connects.

The requirements of reactive and proactive mobility management in terms of cross-layer interaction and network information retrieval can be satisfied with Media Independent Handover Services [3], a to-be standard currently under development by the IEEE 802.21 working group. [71] specifies an alternative interface between the link layer and the IP layer, and other approaches to proactive subnet prefix discovery are described in [70, 69, 77]. Since Media Independent Handover Services will be used as a building block in the frameworks for reactive and proactive mobility management proposed in this thesis, the relevant parts of the draft standard are described in more detail in section 2.4.7.

### 2.3.5 Relation to Multi-Homing

While mobility concerns redirections to previously unknown IP addresses, multi-homing [37] describes a node's ability to redirect packets between multiple IP addresses that it has configured simultaneously. This allows the node to split traffic

mobile node to wait for the completion of the home registration before a Binding Update message can be sent to a correspondent node. The rule is also not needed from a security standpoint since neither the home agent nor a correspondent node can verify that a mobile node conforms to it.

across multiple connections to the Internet for increased quality of service, or to to provide for rapid fail-over in the event one of the connections goes down. One differentiates between *node multi-homing,* where the IP addresses are configured on different network interfaces of the same node, and *access network multi-homing,* where the IP addresses have subnet prefixes that belong to the same access network, but have been allocated from different Internet service providers.[4] Although the policies for IP address selection and switching typically differ between node and access network multi-homing, from the perspective of this thesis, these differences are negligible. The techniques will therefore be more generally referred to as "multi-homing" henceforth.

Given the strong technical relationship between mobility and multi-homing, both techniques can be combined in one protocol. In fact, the mobility extensions for the Host Identity Protocol also support multi-homing, and the NOMADv6 extensions [72] for Mobile IPv6 enable a mobile node to specify a care-of address per packet flow. These extensions have been developed with a focus on node multi-homing, but principally, access network multi-homing could be supported as well.

A side effect of the strong relationship between mobility and multi-homing is that the threats described in section 2.3.2 also apply to multi-homing. The specific threat of redirection-based flooding attacks is even more significant in the case of multi-homing because the attacker could send bogus transport layer acknowledgments without spoofing their IP source addresses: The attacker would register both its own and a victim's on-link IP addresses, and then send spoofed acknowledgments from its own IP address while redirecting the correspondent node's packets to the IP address of the victim. The binding at the correspondent node would then ensure that the acknowledgments are attributed to the packet flow towards the victim. An ability to avoid spoofing its packets' IP source address, also called *IP spoofing* [23], is attractive from an attacker's perspective because routers may verify the topological correctness of the IP source addresses in to-be-forwarded packets and drop packets where the IP source address is incorrect. (The prevalence of such filtering techniques and their relation to redirection-based flooding will be discussed later on in section 3.3.5.) A mobility protocol without multi-homing functionality permits only a single on-link IP address per binding and hence does not allow the attacker to use any other IP source address for the acknowledgments than the registered IP address of the victim.

## 2.4 Protocols Supplementing Mobility

Gaining IP connectivity on a new access link requires a node to discover its neighborhood and configure a new IP address. Many of the protocols that need to be executed for these tasks pertain to stationary nodes just as well as for mobile nodes. But since mobile nodes are required to gain IP connectivity more frequently and oftentimes with active communication sessions, it is important that this happens efficiently. This section explains what steps a node needs to take in configuring IP connectivity on a new access link, and it introduces protocols that are used for this

[4] Another form of access network multi-homing is for an access network to obtain an individual block of IP addresses that does not belong to any particular Internet service provider. Multi-homing is in this case transparent to the nodes, hence node support is not required.

purpose. The section also considers optimized protocols that have been designed to meet the increased efficiency demands of mobile nodes, and it finally explains how proactive mobility management can be realized.

### 2.4.1 Router and Subnet Prefix Discovery

A node that attaches to a new access link learns about local access routers and subnet prefixes during *router discovery* and *subnet prefix discovery*, respectively. The default mechanism for this is defined as part of *Neighbor Discovery* [92]. This protocol calls for access routers to multicast Router Advertisement messages onto a local access link on a loosely periodic basis. A node may listen for advertisements or, if it is unwilling to wait, actively request one by sending a Router Solicitation message. A Router Advertisement message identifies the originating access router and contains a set of subnet prefixes that belong to the local link. The node chooses one of the advertising access routers as a first hop for all packets it sends to destinations off the local link. Neighboring nodes are identified by an IP address that matches one of the subnet prefixes advertised on the local link. Packets destined to a neighbor can be transmitted directly to that neighbor without the help of an access router.

### 2.4.2 Movement Detection

Mobile nodes change their point of attachment as they move. Each such change may be limited to the link layer, in which case the mobile node simply switches between access points that connect to the same access link. The mobile node can in this case continue to communicate at the IP layer as before: It keeps any previously discovered subnet prefixes and configured IP addresses, and sticks to the access router selected as a first hop for packets destined off-link. However, when the old and new access points belong to different access links, the mobile node changes IP connectivity and needs to re-configure IP: It invalidates its current first-hop access router along with any previously discovered subnet prefixes, and it removes existing global IP addresses. The mobile node must also re-run Duplicate Address Detection on its link-local IP address. This is necessary even though the link-local IP address keeps its subnet prefix during handover because a new neighbor may already be using the same link-local IP address. New global IP addresses are subsequently configured as the mobile node learns the subnet prefixes in use on the new access link. At the same time, the mobile node chooses a new first-hop access router. The change in IP connectivity may further cause the mobile node to initiate signaling for an IP mobility protocol.

Changes in link layer attachment can typically be detected straightforwardly through notifications that the link layer may generate when it switches access points. Detecting changes in IP connectivity, *movement detection*, is less trivial. Since different access links have disjunct sets of subnet prefixes—barring the link-local prefix that is valid on every link—movement detection is commonly implemented by analyzing the subnet prefixes advertised in Router Advertisement messages and probing reachability of access routers considered off-link. When the subnet prefixes in use by the mobile node are no longer seen to be advertised, but new subnet prefixes show up instead, the mobile node would typically decide that it has moved to a different access link. On the other hand, received subnet prefixes may also indicate that IP connectivity did not change in spite of a link layer handover, for example, when the

mobile node switches access points that connect to the same link. IP re-configuration should then be avoided.

Movement detection is complicated by the fact that Router Advertisement messages may include incomplete sets of subnet prefixes. Reception of a single advertisement is therefore usually insufficient to decide whether IP connectivity has changed. Mobile nodes also get no indications of an access router's advertising rate. Failure to receive an expected Router Advertisement message therefore does not imply movement either. A typical movement detector would hence draw a possibly premature decision based on a small number of received Router Advertisement messages and, if a change in IP connectivity is assumed, perform Neighbor Unreachability Detection on the first-hop access router to corroborate this.

### 2.4.3 IP Address Configuration

A node may be preconfigured with one or more IP addresses, but more typically, the node auto-configures a link-local IP address and one global IP address per subnet prefix assigned to the access link. The set of on-link subnet prefixes is announced by local access routers in multicast Router Advertisement messages, which are retransmitted on a loosely periodic basis. The node may listen for advertisements or, if it is unwilling to wait, actively request one by sending a Router Solicitation message. IP address auto-configuration can be performed in either stateless or stateful manner, where the modes supported[5] on an access link are indicated in the Router Advertisement messages.

#### Stateless IP Address Auto-Configuration

*Stateless Address Autoconfiguration* [130] is a protocol that enables a node to auto-configure new IP addresses for received subnet prefixes, or for the link-local subnet prefix, without external help, such as from an IP address server. The node chooses an interface identifier—based on the link layer address of the interface to which the IP address is to be assigned [54], or randomly [91], or cryptographically [9]—, and prepends to this the obtained subnet prefix. Stateless Address Autoconfiguration requires the node to subscribe to a *solicited-node multicast group* derived from the desired IP address [29]. This is necessary to ensure that nodes which simultaneously attempt to auto-configure the same IP address can recognize the collision. The subscription is handled by *Multicast Listener Discovery protocol, MLD* [134], a protocol which provides for this purpose a Report message that the node multicasts onto the access link. The node then runs *Duplicate Address Detection,* a sub-protocol of Stateless Address Autoconfiguration, to verify whether the desired IP address is unique: It sends a Neighbor Solicitation message for the IP address to the aforementioned solicited-node multicast group, and listens for a defending Neighbor Advertisement message that any node already using the IP address would send in response. If no responses are received within a period of 1 s, the node assigns the new IP address to the interface. If it receives a Neighbor Advertisement message or another Neighbor Solicitation message for the same IP address, then the node aborts the IP address auto-configuration process and possibly retries with a

[5]There is currently disunity in the*Internet Engineering Task Force, IETF,* on whether the IP address auto-configuration mode advertised in Router Advertisement messages should be interpreted as mandatory or as optional.

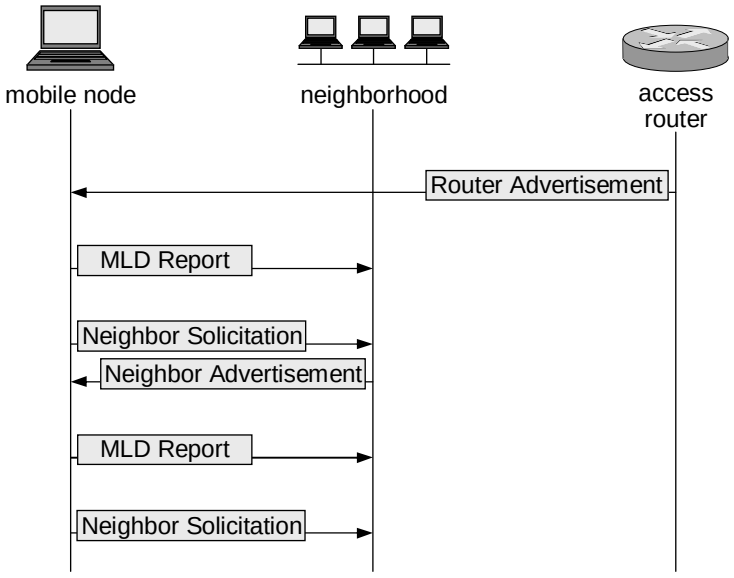



Figure 2.15: Example Stateless Address Autoconfiguration procedure

different IP address. The probability for an IP address collision is small enough [13] to make it negligible, however. Multiple IP addresses can be auto-configured in parallel.

The time-sequence diagram in figure 2.15 exemplifies the use of Stateless Address Autoconfiguration. It shows a node which, after receiving a Router Advertisement message that includes the set of on-link subnet prefixes, attempts to configure an IP address that is already in use. The node then generates a new IP address and tries again, this time with success.

### Stateful IP Address Auto-Configuration

Where access routers specify in transmitted Router Advertisement messages that global IP addresses may be configured in stateful manner, nodes can use the *Dynamic Host Configuration Protocol for IPv6, DHCPv6* [40], to request the assignment of a global IP address from a server. This involves either a four-way handshake or a two-way handshake, depending on the configuration of access routers. The node initiates either exchange with a multicast Solicit message. Per default, DHCPv6 servers that receive the Solicit message respond with an Advertise message. The node collects these responses for a random time of between 1.0 s and 1.1 s, finally selects one of them and sends the originating DHCPv6 server a Request message. The selected server completes the message exchange with a Reply message, including any IP addresses assigned to the node. Alternatively, a DHCPv6 server may be configured to respond to the node's Solicit message directly with a Reply message that contains any assigned IP addresses. This completes IP address assignment with a two-way handshake. Both message exchanges require the node to use a link-local IP address as its IP source address. Consequently, a link-local IP address cannot be auto-configured through DHCPv6 and needs to be obtained via Stateless Address Autoconfiguration beforehand.

To limit the scope of this thesis, it will henceforth be assumed that all IP address auto-configuration is performed in stateless manner. Nonetheless, with the exception

of proactive mobility management (see section 2.3.4), the results of this thesis can be transferred to scenarios where IP addresses are auto-configured through DHCPv6.

### 2.4.4 Neighbor Unreachability Detection

The mapping between IP addresses and link layer addresses may grow stale as nodes shut down or leave the access link, for example, due to mobility. Nodes are recommended to take advantage of connectivity indications from protocol layers above IP in monitoring a neighbor's reachability. Such indications include acknowledgments received for previously transmitted data. Yet, upper-layer indications may not always be available. A node can then use *Neighbor Unreachability Detection* to actively probe a neighbor's liveliness. Neighbor Unreachability Detection is a sub-protocol of Neighbor Discovery. A node interested in a neighbor's reachability sends up to three Neighbor Solicitation messages directly to the link layer address known for that neighbor. Consecutive solicitations are spaced by 1 s during which the solicited node is expected to respond with a Neighbor Advertisement message. If such an advertisement is sent, the link layer address known for the neighbor is still correct. If no advertisement is received after three solicitations have been sent, the link layer address is assumed to be stale and the querying node starts with IP address resolution anew. The potential for packet loss during Neighbor Unreachability Detection is covered by the retransmissions of the Neighbor Solicitation message.

### 2.4.5 Internet Control Message Protocol for IPv6

The *Internet Control Message Protocol for IPv6, ICMPv6* [32], defines error and notification messages that IPv6 nodes exchange to convey or retrieve information relevant at the IP layer. For example, a host that receives a packet it cannot process due to lack of support for the included payload may return an ICMPv6 Parameter Problem message to the sender of the packet. Routers that do not know how to forward a received packet may inform the sender with an ICMPv6 Destination Unreachable message. Protocols at the IP layer or above use ICMPv6 messages to detect error conditions. For example, Mobile IPv6 utilizes ICMPv6 Parameter Problem messages to determine when a correspondent node does not support route optimization.

### 2.4.6 Optimizations

The default protocols for router and subnet prefix discovery, IP address autoconfiguration, and movement detection were designed for use in environments where nodes are primarily stationary. Listen times, desynchronization delays, and rate limitations are hence dimensioned conservatively, precluding race conditions and transmission collisions at the cost of latency. For stationary nodes, this is typically not a problem since they can run a working IP configuration for a long time. But the extended latencies [141, 135] in standard protocols are highly disadvantageous in environments where mobile nodes frequently re-configure IP. Handover delays may then be in the order of seconds. This makes it difficult to support non-real-time or soft-real-time services like file transfers or media streaming, and it precludes any meaningful use of hard-real-time applications such as Internet telephony. A multitude of optimizations have therefore recently been put forth to streamline handover-related activities and reduce handover delays. Particularly promising are the following approaches.

### Router and Subnet Prefix Discovery

Strict rate limitations in the Neighbor Discovery protocol force access routers to space consecutive Router Advertisement messages by a random time of between 3 s and 4 s at least. This leads to considerable delays for router and subnet prefix discovery. Router Solicitation messages may in some cases permit a mobile node to obtain an advertisement faster. But on a durable basis, solicited advertisements, too, cannot be sent more rapidly than once per 3.5 s on average. Less rigid rate limitations make router and subnet prefix discovery more efficient and thereby to also improve movement detection. Accordingly, average advertising rates of between 30 ms and 70 ms were permitted a posteriori for mobile environments [63]. Rate limitations for solicited Router Advertisement messages were not relaxed at the same time, so unless access links are only very sparsely populated with mobile nodes and Router Solicitation messages are transmitted only very infrequently, any solicited Router Advertisement messages are highly likely to be substituted by unsolicited advertisements that are sent at a high rate anyway. The high advertising rates thus in fact redundantize the use of Router Solicitation messages. An obvious disadvantage of this approach to router and subnet prefix discovery is that it comes at the cost of increased bandwidth consumption.

More sophisticated scheduling intervals in access routers can improve router and subnet prefix discovery with respect to both bandwidth consumption and efficiency. *Fast Router Advertisement* [36] is an optimization for Neighbor Discovery that permits a mobile node to solicit an immediate Router Advertisement message. Randomized desynchronization delays are here replaced by an algorithm that neighboring access routers use to determine an order in which they respond to the Router Solicitation message without collision, whereby the access router ranked first responds without delay. The algorithm is seeded with neighboring access routers' link-local IP addresses and the IP source address of the solicitation.

### IP Address Auto-Configuration

Stateless Address Autoconfiguration suffers decreased efficiency due to both, the mandatory 1-s listen time for nodes that have sent a Neighbor Solicitation message, and a random desynchronization delay of up to 1 s for Multicast Listener Discovery Report messages. *Optimistic Duplicate Address Detection* [87], eliminates both of these delays. This protocol is tailored to mobile environments in particular, so desynchronization delays are removed based on the assumption that natural movement patterns already provide sufficient randomness to avoid collisions in most cases. Moreover, the protocol enables a node to use an IP address in parallel with verification. Nodes temporarily change the rules by which they do Neighbor Discovery so as to avoid pollution of neighboring nodes' IP address resolution state with illegitimate information. This implies restrictions on direct packet exchanges with neighboring nodes, but does not impact communications with correspondent nodes off-link.

### Movement Detection

Low rates of Router Advertisement messages in combination with the need to acquire multiple advertisements for complete subnet prefix information are responsible for long movement detection delays in standard IPv6. If Neighbor Unreachability Detection on the current access router is performed as part of movement detection,

these delays increase even further. Moreover, Neighbor Unreachability Detection in this case happens at a time at which a change in IP connectivity may have invalidated all existing IP addresses. So since the mobile node cannot leave its IP source address unspecified during Neighbor Unreachability Detection, it must re-verify the uniqueness of its link-local IP address before it initiates the procedure. Overall, reliable movement detection may thus easily take up to 10 s and more [135].

One reason for the low movement detection performance is that mobile nodes do not know in which intervals they can expect to receive Router Advertisement messages from a particular access router, so they cannot efficiently conclude a change in IP connectivity when such advertisements cease to appear. One important enhancement of Neighbor Discovery was therefore the introduction of an Advertisement Interval option [63] that access routers can include in their Router Advertisement messages to declare an upper bound on the intervals in which they transmit these messages. The advertised upper bound aids mobile nodes in deciding when the absence of advertisements from a specific access router indicates a change in IP connectivity. If combined with the increased advertising rates described above, this increases the efficiency of standard movement detection becomes considerably more efficient.

On the other hand, even with increased advertisement rates and the use of Advertisement Interval options, standard movement detection still fails to detect a change in IP connectivity on the absence of a single expected advertisement, due to the potential for packet loss, or on the presence of a single unexpected advertisement, due to the incompleteness of included subnet prefixes. Standard movement detection hence requires the mobile node to wait for the lack of multiple Router Advertisement messages until a change in IP connectivity can be concluded with reasonable certainty. To solve this problem, the IETF set about developing mechanisms that could quickly detect movement based on a single Router Advertisement message from a new access router, rather than waiting for missing advertisements from an old access router. Two mechanisms originated from this. *Complete Prefix List* [27] works with unmodified legacy access routers. A mobile node maintains a list of learned subnet prefixes, possibly obtained by reception of multiple Router Advertisement messages. After the list has matured for a while, the mobile node can assume a change in IP connectivity with high probability when a newly received Router Advertisement message exclusively contains subnet prefixes not in the list. However, such predictions are based on potentially incomplete information, so the mobile node might assert movement even when none actually occurred.

The *DNA Protocol* [89] integrates Fast Router Advertisement for timely transmissions of solicited Router Advertisement messages. Neighboring access routers additionally choose a certain subnet prefix to serve as a *access link identifier* and be as such indicated in all transmitted Router Advertisement messages. This allows a mobile node to reliably detect changes in IP connectivity based on a single advertisement. Alternatively, the mobile node can explicitly check with access routers as part of the solicitation-advertisement exchange whether a subnet prefix used before a link layer handover, as such called a *landmark*, is still valid after the link layer handover. The DNA Protocol used to refer to Complete Prefix List as a fall-back mechanism for access to links with legacy access routers. More recently, the Complete Prefix List algorithms have been fully integrated into the DNA Protocol. They are now no longer maintained as a separate protocol.

### 2.4.7 Media Independent Handover Services

Efficient IP mobility management requires a mobile node to quickly identify and select suitable access points prior to handover, and to closely synchronize activities at the link and IP layers during handover execution. Both of these prerequisites are hardly met today, however. Classic access point selection techniques are based on signal strength alone. They disregard other important aspects such as operator, service provider, and cost; or link layer properties such as security and quality of service. A mobile node may further be unable to connect at the IP layer despite a working link layer attachment, for example, due to a mismatch in supported IP address configuration mechanisms. The selection of an access point becomes even more challenging if multiple access technologies are involved. And proactive mobility management introduces the additional requirement that mobile nodes must obtain subnet prefix information for a discovered access link so as to auto-configure new IP addresses for that access link prior to handing over to it. Cross-layer interaction and synchronization is a concept that deviates from the traditional view that the responsibilities of different layers in a network stack should be cleanly separated [144].

To approach these challenges, the IEEE chartered its 802.21 working group to develop a set of *Media Independent Handover Services* [3] for improved reactive and proactive mobility management. The standard includes a unified interface separating different heterogeneous technologies at the link layer from protocols at the IP layer or above. This interface defines a set of events and commands that protocols at the IP layer or above can read or issue, respectively, to synchronize their handover-related activities with the link layer. The standard further enables IP layer protocols to acquire information that assists them in access link selection and handover preparations. Overall, the functionality provided by Media Independent Handover Services is split into three complementary services: an event service, a command service, and an information service.

#### Architecture

Pivotal to all Media Independent Handover Services operation is an *MIH function*, which serves as an intermediary between the handover-related activities at the link layer and the layers above. (Here, "MIH" denotes "media-independent handover".) One instance of the MIH function is located between a mobile node's link and IP layers, as shown on the left-hand side in figure 2.16. This shim provides an abstraction from the specifics of different access technologies and presents a homogeneous interface to protocols at the IP layer or above. Interested protocols include mobility or multi-homing protocols, movement detection mechanisms, as well as protocols for neighbor discovery or IP address auto-configuration. For example, the DNA Protocol may send a router solicitation message for renewed router discovery and movement detection when the mobile node switches to a different access point. The DNA Protocol may further interface to Stateless Address Autoconfiguration so as to create a new on-link IP address when the link layer handover to the new access point is found to cause a change in IP connectivity. A mobility protocol such as Mobile IPv6 may finally initiate a binding update for the new on-link IP address. In the reference architecture in figure 2.16, the MD function implements movement detection (so "MD" stands for "movement detection").

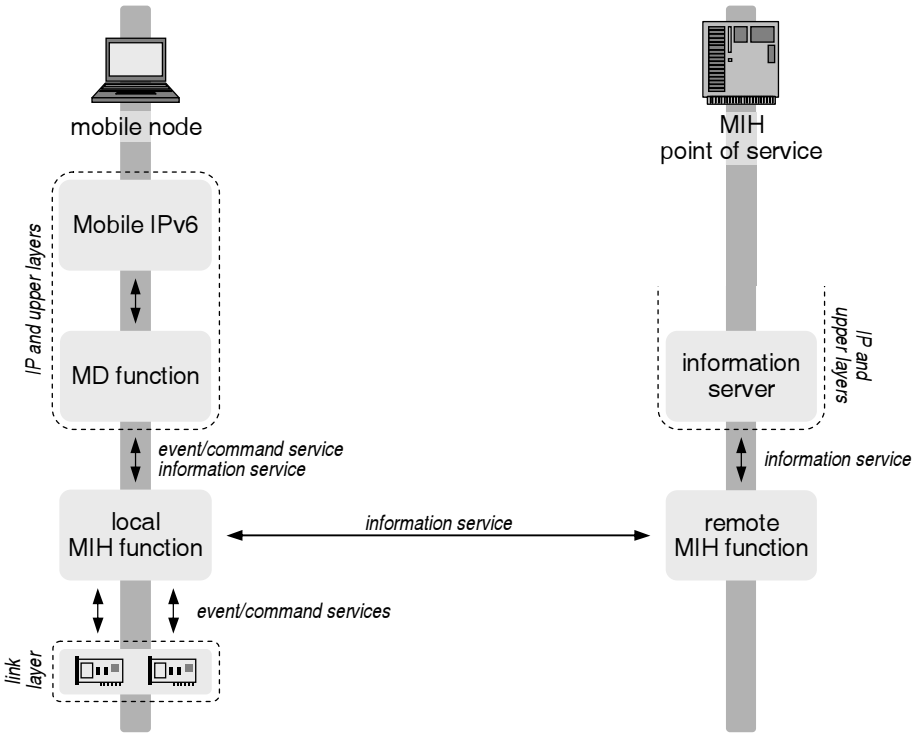



Figure 2.16: Reference architecture of Media Independent Handover Services

A companion entity of the MIH function is located on an *MIH point of service* in the access network. This is depicted on the right-hand side of figure 2.16. The remote MIH function on the MIH point of service may assist the MIH function on a mobile node during handovers that are controlled by the mobile node. It further facilitates network-controlled handovers, where it is in charge of handover-related activities on the mobile node. For remote information retrieval, the MIH point of service may further run an *MIH information server*. Interested mobile nodes can access information from this database via the remote MIH function. Communications between the MIH functions on the mobile node and on an MIH point of service use the *MIH protocol*. This defines a set of packet formats that encapsulate events, commands, and information for transmission via the network.

As a mobile node moves across access links, its peer MIH point of services may change. Media Independent Handover Services define an *MIH function discovery* to aid the mobile node in finding another MIH point of service after attaching to a new link. The draft standard limits the discovery to the link layer, so the mobile node can only find MIH points of service within the same broadcast domain. A mobile node typically discovers new MIH points of service immediately after a handover, but uses the available services only later when preparing for a subsequent handover. This generally provides for a sufficient time frame for MIH function discovery to proceed without delaying handover-related activities.

### Event Service

Protocols at the mobile node's IP layer or above can use the event service to register with the local MIH function for notifications about specific events at the link layer, such as changes in link layer attachment or in the quality of a particular connection.

The MIH function monitors the mobile node's interfaces for the respective events in a technology-specific manner and generates a technology-independent notification for the interested protocols at higher layers as the events occur.

An example for the utility of the event service is movement detection for mobile nodes. Movement detection mechanisms at the IP layer are interested in changes in link layer attachment, so they may subscribe to notifications about these events from the MIH function. The MIH function monitors interfaces at the link layer for changes in attachment and accordingly generates technology-independent MIH Link Up Indication and MIH Link Down Indication messages for the movement detection mechanism. This allows a movement detection mechanism to quickly review the current IP configuration in case of a link layer attachment change, and to re-configure if need be. In proactive mobility management, the MIH function may anticipate changes in link layer attachment and notify protocols at the IP layer about this with a Link Going Down Indication message. This facilitates handover preparations and proactive mobility management at the IP layer.

Events may also originate with a remote MIH function on an MIH point of service in the network. The events are in this case sent via the network using the MIH protocol. For example, an MIH function in the network may send an MIH Link Going Down Indication message to the MIH function on a mobile node when the access point to which the mobile node attaches is about to be shut down for administrative reasons. The MIH function on the mobile node can then inform IP layer protocols about the forthcoming link break.

#### Command Service

The command service enables protocols at the IP layer or above to control handover-related activities at the local link layer or remotely in the network. A command is passed to the local MIH function and is as such technology-independent. If the command is directed to the local link layer, the MIH function translates it into technology-specific commands and sends these to the respective interfaces. The interfaces execute the commands and return a confirmation each, which the MIH function aggregates and forwards to the protocol that invoked the command. A remote command is directed to the MIH function on an MIH point of service in the network. It, too, originates with a protocol at the IP layer or above and traverses the local MIH function. But from there the command is forwarded to the remote MIH function by help of the MIH protocol. The remote MIH function may forward the command to a protocol at a higher layer, or translate it into technology-specific commands for the link layer beneath.

The command service can be used by a protocol at the IP layer or above to find a suitable access point and to control a link layer handover thereto. For example, an IP layer movement detection mechanism or mobility protocol on a mobile node may periodically send an MIH Get Status Request message to the local MIH function for a survey on available points of attachments and their properties such as bandwidth, packet loss rate, and medium access delays. The MIH function translates this command into technology-specific Get Status Request messages, one for each of the mobile node's interfaces, and receives Get Status Confirm messages containing the desired information in response. The MIH function aggregates the responses within a technology-independent MIH Get Status Confirm message and relays this to the

protocol that asked for it. In a similar manner, the MIH Scan Request message can be used by protocols at the IP layer or above to measure the received signal strength of available access points. The local MIH function maps this command onto a set of technology-specific Link Scan Request messages, one for each interface, and receives a set of Link Scan Confirm messages in return. The responses are aggregated and forwarded as an MIH Scan Confirm message to the calling protocol.

Another example for the use of the command service is in proactive mobility management, where the IP layer needs to be in control of link layer detachment and attachment. A movement detection or mobility protocol may here send the local MIH function an MIH Switch Request message identifying the old and selected target access link. The MIH function translates the MIH Switch Request message into a sequence of commands specific to the one or two involved link layer interfaces: If the link layer technology supports make-before-break handovers, a Link Connect Request message for the target access point is sent first, and a Link Disconnect Request message for the old access point is sent second. Where only break-before-make handovers are possible, the sequence of messages is reverse. The link layer responds to the requests with Link Disconnect Confirm and Link Connect Confirm messages, and the MIH function finally sends an MIH Switch Confirm message to the protocol that invoked the handover.

#### Information Service

The information service enables a mobile node to discover auxiliary information about access links within reachability. Such information includes the access technology; access link, operator, and service provider identification; link layer security and quality-of-service properties; permitted IP configuration methods such as Stateless Address Autoconfiguration or DHCPv6; the link layer address of the access point and the channels it is operating on; as well as the IP address of available access routers and the set of subnet prefixes in use on the access link. The information service thus supports the mobile node in preparing a handover and selecting a suitable target access point. One particular use case for the information service is in proactive mobility management, where a mobile node must obtain the subnet prefixes of a new access link before it actually attaches to that link. The mobile node can here use the information service to resolve the link layer address of a discovered access point into a list of subnet prefixes in use on the link to which this access point attaches.

Retrievable information is organized in *information elements*. Protocols at the mobile node's IP layer or above may request these either from the local MIH function or, more generally, via a remote MIH function from an MIH information server in the network. In either case, an interested protocol sends an MIH Get Information Request message to the local MIH function, identifying one or multiple information elements that are to be retrieved. If the requested information is available locally, the MIH function responds directly with an MIH Get Information Confirm message including the respective information elements.

On the other hand, if the requested information is not available locally, the MIH function on the mobile node forwards the MIH Get Information Request message to the remote MIH function on a previously discovered MIH point of service. The remote MIH function translates the received MIH Get Information Request message

into an MIH Get Information Indication message and sends this to an MIH information server to retrieve the requested information. The MIH information server looks up the desired information elements and returns them within an MIH Get Information Response message. The remote MIH function copies the information elements into an MIH Get Information Confirm message and sends it to the local MIH function on the mobile node. The latter finally relays the message to the protocol that requested the included information elements.

## 2.5 Transmission Control Protocol

TCP allows applications on remote nodes to create a virtual connection over which they can exchange data. The protocol offers an abstraction from the packet-oriented routing substrate, delivering data bidirectionally as reliable and ordered byte streams. TCP further controls the rate at which data is transmitted to avoid both buffer overflows at the receiver as well as congestion in the network. As the most popular transport protocol providing these features, TCP is used by nearly all applications that require reliability. Real-time applications such as Internet telephony can cope with sporadic packet losses and hence do not depend on automatic loss recovery. But a significant part of all real-time applications uses TCP nonetheless in order to take advantage of the other salient feature that TCP provides.

Due to its prevalence amongst existing transport protocols, TCP has been subject to extensive research efforts since its initial publication in 1981. This development has led to a diversity of different TCP variants that are in use across the Internet today [78]. The major TCP variants have been standardized by the IETF, but there are also a number of proprietary spin-offs. The following description explains the milestones in TCP evolution with a focus on those standardized TCP versions which a clear majority of TCP clients and servers support today.

### 2.5.1 Original Specification

TCP was originally specified in RFC 793 [107]. The protocol functions entirely symmetric, modulo the initiator-responder approach during connection establishment and tear-down, and hence allows for data to be exchanged bidirectionally. Each byte of data flowing into a given direction is identified by a sequence number. A TCP receiver cumulatively acknowledges the data that it has received without intervening losses. TCP is *self-clocking* in that a new payload packet, or *segment*, may be sent only at connection establishment, after a retransmission timeout, or in response to an arriving acknowledgment.

To provide for the possibility of packet loss, a TCP sender keeps a copy of the data which it has sent to the other end, and for which it still expects an acknowledgment. TCP uses a *retransmission timer* to estimate when previously dispatched segments have been lost and ought to be resent. Specifically, when a TCP sender does not get an acknowledgment from the receiver for the duration of a *retransmission timeout*, all data sent, but not yet acknowledged, will be retransmitted. TCP attempts to approximate the *retransmission timeout period* to the current round-trip time on the transmission path. It periodically measures the time it takes for a transmitted data segment to get acknowledged, and calculates a *smoothed round-trip time* as an exponentially weighted moving average. To accommodate for sudden changes in the

round-trip time, TCP adds to this estimate a multiple of the measured round-trip time variation and uses the sum as the retransmission timeout period.

The interface between TCP and the local application provides two buffers. A sending application writes outgoing data into a *send buffer*. The data is kept in the send buffer until it has been transmitted and an acknowledgment indicating its successful delivery has been received. Data received from the network is placed into a *receive buffer*. TCP uses the receive buffer to sequence the data from incoming out-of-order segments and to finally present this data to the receiving application. While an attempt to write data into a full send buffer can be handled via a local error indication, keeping a remote peer from sending data that might not fit into the receive buffer requires over-the-network coordination. This is the purpose of *flow control*, which allows the TCP receiver to advertise to the peer how much additional data it is prepared to receive. The receiver maintains this amount of data as its *receive window* or *advertised window*, and it communicates this to the peer as part of its acknowledgments. The TCP sender stores the receiver's advertised window in a local variable called the *send window*, and it ensures that the amount of "outstanding" data—that is, any data sent, but not yet acknowledged—does not exceed the send window.

### 2.5.2 TCP Tahoe

Early TCP implementations were prone to burden the network by sending too much data in a single burst, and they reacted poorly to any resulting congestion. The original protocol specification was therefore later amended by a technique called *congestion control*. This steadily monitors the throughput capacity on the transmission path in an attempt to take advantage of the available bandwidth without overloading the network. The improved protocol version became known as *TCP Tahoe*, named after the first implementation in version 4.3 of the BSD operating system.

Congestion control limits TCP's transmission rate by means of a *congestion window*, which dynamically opens as TCP probes the network for additional capacity, and retracts once congestion is detected. A TCP sender can only send a segment if this segment does not increase the amount of outstanding data to a value higher than the current congestion window size. In conjunction with flow control, this means that new data may be sent only as long as it does not increase the amount of transmitted, yet unacknowledged data to more than the minimum of the send window and the congestion window, that is, the TCP sender's *available window*.

Congestion control probes the throughput capacity on the transmission path in either of two modes: *Slow-start mode* initially allows TCP to quickly accelerate data transmission to the point at which the throughput capacity along the transmission path becomes saturated. Slow-start mode begins with a congestion window just large enough to send a single segment, and it opens by one additional segment size with each arriving acknowledgment. This doubles the amount of outstanding data once per round-trip time.

The rapid opening of the congestion window in Slow Start mode typically causes network congestion at some point. Network congestion, in turn, results in packet loss. TCP was designed for wired networks, where, reversely, the occurrence of

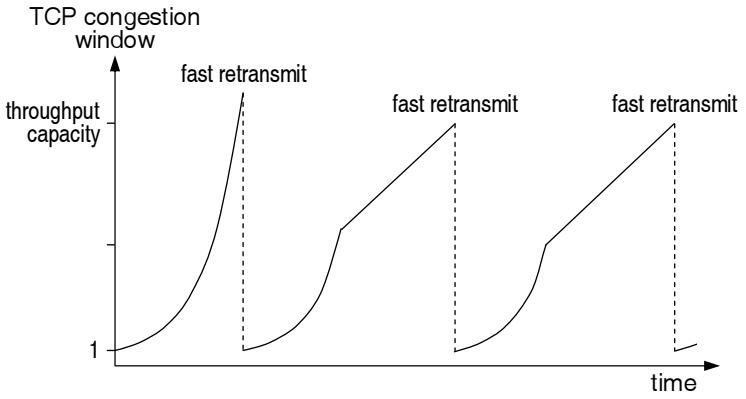



Figure 2.17: TCP congestion window progression in TCP Tahoe

packet loss also oftentimes implies network congestion. TCP hence regards packet loss as an indication of congestion. On detecting packet loss, TCP sets a variable called the *slow-start threshold* to half of the current amount of outstanding data, retracts the congestion window to the size of a single segment, and re-enters Slow Start mode. This time, TCP switches to *Congestion Avoidance mode* when the congestion window reaches the size of the slow-start threshold. The congestion window then opens by the size of one segment during each round-trip time. The transmission rate therefore increases more carefully in Congestion Avoidance mode than it does in Slow Start mode, ensuring that at most a single excess segment can be injected into the network. Slow Start and Congestion Avoidance modes are today mandatory for all TCP implementations [21].

The discovery of packet loss via the retransmission timer is in many cases inefficient since the conservative estimation of the retransmission timeout period results in a long period during which no data can be sent. TCP Tahoe therefore includes a way to more efficiently detect packet loss based on the arrival of three continuous *duplicate acknowledgments*. A TCP receiver retransmits its previous acknowledgment when it receives a segment of data that it is unable to forward to the application due to missing previous data. Such an out-of-order segment may be due to either packet reordering in the network or the loss of a previous segment. Packet reordering becomes increasingly unlikely as a source of the out-of-order segment as more segments arrive without filling the gap in the receive buffer, hence packet loss is presumed after three of these segments have been received. The resulting three duplicate acknowledgments notify the TCP sender of the segment that the receiver is missing, so the TCP sender can perform a *fast retransmit* of this segment without waiting for the expiration of the retransmission timer. It then adjusts the slow-start threshold and the congestion window as described above, and subsequently enters Slow Start mode. TCP Tahoe retransmits only the segment that apparently got lost and continues with the new segment that was scheduled for transmission right before the fast retransmit. Figure 2.17 shows the progression of the congestion window in a typical TCP Tahoe implementation.

### 2.5.3 TCP Reno

The periodic alternation between Slow Start mode and Congestion Avoidance mode that is characteristic for TCP Tahoe leads to decreased throughput because the con-

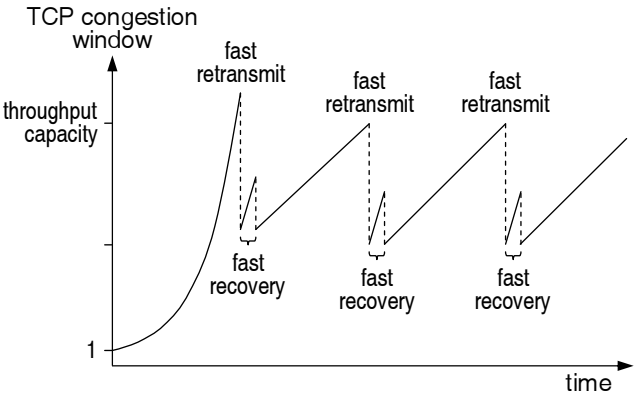



Figure 2.18: TCP congestion window progression in TCP Reno

gestion window shrinks to a single segment size upon each observed packet loss (see figure 2.17). Re-entering Slow Start mode is appropriate when a high number of segments have been lost from a single window of data because the lack of acknowledgments might otherwise impede self-clocking and cause TCP to stall. However, in most cases, packet loss effects an only small number of segments, typically just the one additional segment that the TCP sender emits into the network at the point its available window grows beyond the capacity of the transmission path in Congestion Avoidance mode. Retracting the congestion window to a single segment size and re-entering Slow Start mode is then in general all too conservative.

The advanced functionality of *TCP Reno* enables a more efficient recovery from the loss of a single segment. This TCP variant introduces a new *Fast Recovery mode*, which governs congestion control after a fast retransmit until the TCP sender has assurance that all data outstanding at the time of the fast retransmit has been received at the other end. On entering Fast Recovery mode, TCP Reno sets both the congestion window and the slow-start threshold to half of the amount of currently outstanding data. It waits until the amount of outstanding data has reduced to this threshold and only then continues sending further segments. While in Fast Recovery mode, the amount of outstanding data is limited to the new value of the slow-start threshold.

Duplicate acknowledgments indicate the delivery of new data at the TCP receiver, but they do not specify *which* data was received. Data delivered subsequent to the packet loss hence continues to occupy space in the TCP sender's congestion window until the arrival of a non-duplicate, *full acknowledgment* finalizes Fast Recovery mode. To maintain the current amount of outstanding data in Fast Recovery mode, the TCP sender therefore "inflates" the congestion window as duplicate acknowledgments arrive. TCP exits Fast Recovery mode and returns to Congestion Avoidance mode when all data outstanding at the time of the fast retransmit has been acknowledged by the receiver. The congestion window is then "deflated" back to the size it had at the time Fast Recovery mode was entered. Figure 2.18 shows the typical progression of the congestion window in TCP Reno.

### 2.5.4 TCP NewReno

Fast retransmit and Fast Recovery mode allow TCP Reno to quickly retransmit a single lost segment, but the algorithms fail to efficiently recover when multiple

segments are lost within a single round-trip time. The arrival of a retransmitted segment in the presence of multiple packet losses causes the TCP receiver to generate a *partial acknowledgment*, which confirms the retransmitted segment, but does not cover all of the data outstanding at the time of the fast retransmit. A TCP Reno sender in Fast Recovery mode ignores partial acknowledgments and continues to wait for a full acknowledgment. Since this never arrives when multiple packets were lost, the TCP sender eventually runs into a retransmission timeout and switches to Slow Start mode. On the other hand, partial acknowledgments carry sufficient information for the TCP sender to retransmit the next lost segment from a window with multiple packet losses without waiting for the expiry of a retransmission timer. *TCP NewReno* [49] was the first TCP variant to take advantage of this information. When a TCP NewReno sender receives a partial acknowledgment while in Fast Recovery mode, it retransmits the next segment that appears to be missing and continues as before in Fast Recovery mode.

The TCP NewReno RFC describes two strategies for resetting the retransmission timer in response to partial acknowledgments. A TCP NewReno sender operating the *Slow-but-Steady variant* resets the retransmission timer after each received partial acknowledgment. This helps to avoid a retransmission timeout while TCP recovers from the loss of multiple segments in a single window of data. The Slow-but-Steady variant conforms to recommendations in RFC 2988 that call for restarting the retransmit timer after every packet transmission or retransmission. On the other hand, a partial acknowledgment can only be generated by the TCP receiver upon the delivery of a retransmitted segment, so the Slow-but-Steady variant can repair only one packet loss per round-trip time. It may therefore take a substantial amount of time until all segments lost from a window are retransmitted. The *Impatient variant* of NewReno resets the retransmission timer in Fast Recovery mode only when the first partial acknowledgment arrives. The TCP sender thus eventually falls into a retransmission timeout when a high number of segments were lost from a single window, and it retransmits the remaining packet losses more quickly in Slow Start mode.

### 2.5.5 TCP SACK

TCP's cumulative acknowledgment strategy complicates the recovery from multiple packet losses in a single window of data. Selective Acknowledgment options extend the protocol by a means for explicit acknowledgments, allowing a TCP receiver to augment a cumulative acknowledgment by the specification of one or more continuous chunks of successfully received data that cannot be covered by the cumulative acknowledgment itself due to missing earlier data. Selective acknowledgments facilitate more efficient loss repair at the TCP sender. The TCP variant that makes use of these options is referred to as *TCP SACK*.

RFC 3517 [20] proposes one particular algorithm for implementing TCP SACK. The integral part of it is a function that evaluates the information from received Selective Acknowledgment options to estimate whether a given piece of outstanding data was lost. This allows a TCP sender to selectively retransmit the data considered lost without causing the amount of outstanding, non-lost data to increase beyond the currently available window. The TCP sender can thus retransmit up to a full window of data per round-trip time, leading to a much shorter recovery phase than

comparable packet loss would cause in TCP NewReno. The FreeBSD and Windows operating systems support [123, 33] the algorithm in RFC 3517.

### 2.5.6 TCP Limited Transmit

The fast-retransmit mechanism implicitly relies on the arrival of three consecutive duplicate acknowledgments at the TCP sender, and hence requires three segments following the lost segment to be successfully delivered to the TCP receiver. If less than three segments are delivered after the packet loss, the TCP sender does not perform a fast retransmit. Loss recovery is then triggered through the expiration of the retransmission timer, which is typically less efficient. This may happen in particular when the available window is small.

The *Limited Transmit algorithm* [5] leverages the fact that the first and second duplicate acknowledgments each indicate the successful delivery of a segment to the receiver, hence a reduction in the amount of outstanding data. This allows the TCP sender to emit a new segment upon the arrival of each of the first two duplicate acknowledgments, aiding the TCP receiver to generate additional duplicate acknowledgments and, thus, to eventually trigger a fast retransmit back on the TCP sender side. Limited Transmit can be combined with the TCP variants described afore.

### 2.5.7 Delayed Acknowledgments

The initial TCP specification called for a receiver to generate an acknowledgment for each incoming segment. This strategy was found to be responsible for significant processing overhead on the TCP sender side [30] as well as increased network load [21]. Moreover, some interactive applications like remote login oftentimes generate an immediate response to received data. It would be efficient to piggyback this response onto the acknowledgment that TCP generates for the same data, yet if the acknowledgment is generated promptly upon the reception of a segment, any application layer response typically arrives too late to be piggybacked onto the acknowledgment.

A more sophisticated acknowledgment strategy, *delayed acknowledgments*, calls for a TCP receiver to postpone the transmission of acknowledgments for a while in an attempt to cumulatively acknowledge multiple received segments by means of a single acknowledgment. This reduces the overhead related to the transmission and processing of acknowledgments, and it gives the application time to respond to incoming data. The algorithm was initially proposed in [30], and a slightly modified version was eventually recommended [21] to all Internet hosts. According to the recommendation, the maximum delay for an acknowledgment must be less than 500 ms, although implementations may choose a smaller value, and an acknowledgment must be immediately sent when a second segment arrives in the meantime. Also, a duplicate acknowledgment must be sent directly for each out-of-order segment in an effort to aid a fast retransmit.

# 3. Problem and Solution Space Analysis

Mobility protocols that apply the basic mobility strategies described in section 2.3 without further sophistication have been frequently found [60, 83, 7, 82] to perform poorly due to long handover delays and high handover-related packet loss. A large body of effort on optimizing IP mobility management tackles these problems from different angles, yet mostly at the cost of increased operational and deployment expenses, or in a manner unsuitable for global mobility support. This chapter begins with an analysis of such related work in section 3.1. It is found that none of the examined proposals eliminates the need to verify a mobile node's reachability at a new IP address before packets can be sent to that IP address, and an efficient, secure, and low-cost approach to reducing the adverse impacts of reachability verification at present does not exist. Given that other flooding attack types are already possible in the non-mobile Internet of today and pose the question of whether the existence of those might render the introduction of "yet another" type of flooding negligible, section 3.2 explains the significance of redirection-based flooding with a comparison between existing types of flooding attacks and the potentially new type of redirection-based flooding attacks. It is found that redirection-based flooding attacks would feature a set of properties that classic flooding attacks do not have, and that would be attractive to an attacker. Section 3.3 discusses more efficient measures against redirection-based flooding attacks, some of which have been proposed in the research community. It thereby turns out that most of these measures have weaknesses that defeat their suitability for use on a global basis. A new approach, concurrent reachability verification, is therefore finally presented in section 3.4. This maintains the simplicity and applicability advantages of reachability verification as used today without causing any handover delays itself.

## 3.1 Related Work

The following survey discusses and analyzes existing work on IP mobility management optimizations. At a high level, the optimizations can be split in two main categories: those which streamline handover signaling, and those which improve a

mobile node's ability to communicate bidirectionally during a handover. Optimizations that tackle signaling can further be separated into those which decrease the number of round-trips required for a binding update, and those which reduce round-trip times. Optimizations that improve the mobile node's ability to communicate can be split into route repair locally on the mobile-node side, packet duplication, and dual network attachment.

It should be emphasized that this review of related work focuses on optimizations for mobility management at the IP layer and hence covers only a portion of the work in the mobility management area. Mobility protocols that enable a mobile node to change IP connectivity without losing active communication sessions have been been proposed (and partly standardized) also for other layers of the network protocol stack. One of the earliest ideas of transport layer mobility management was *TCP Migrate* [120], an optional extension to TCP that permits an active connection to be moved to a different pair of IP addresses in case one of the end nodes performs a handover. More recently, mobility management has been proposed [8, 143, 68, 124] for inclusion in the Stream Control Transmission Protocol [125], a next-generation transport protocol developed in IETF. Transport layer mobility management has the advantage over mobility support at the IP layer that, in case of a handover, appropriate adaptation and congestion avoidance techniques can be initiated promptly to deal with a path change.

Another approach towards a mobile Internet that has recently received much attention is to put mobility support into applications. Several proposals in this realm, including [115, 16, 98], are based on the Session Initiation Protocol [112]. Application layer mobility support permits fine-tuned, application-specific handover optimization because the properties of an application and its behavior in the event of a handover are best known by the application itself. On the other hand, an important disadvantage of introducing mobility support at the application layer is that legacy applications remain unsupported. At the same time, the cost for developing new, mobility-aware applications takes extra effort. Moreover, as a performance drawback, application layer mobility support implies the reestablishment of open transport protocol connections during handover. This involves additional signaling and is likely to reduce the quality of an application.

### 3.1.1 Assessment Criteria

The mobility management techniques described subsequently will be evaluated based on the following criteria:

1. *Efficiency* — Changes in IP connectivity may entail adverse effects on protocols at transport protocols and applications in terms of handover delays or handover-related packet loss. These effects reduce application quality and should therefore be as small as possible.

2. *Operational overhead* — Many mobile nodes have limited processing resources and communicate over low-bandwidth links. Extra transmissions of signaling or payload packets, as well as computationally expensive operations should therefore be limited.

3. *Deployment costs* — Requirements for special network infrastructure—be it for new devices, or for hardware or software upgrades—, as well as implied labor for network administration or configuration should be avoided as much as possible. They are costly and constitute a deployment obstacle.

4. *Applicability* — A mobility protocol should be usable for mobile nodes in a wide range of scenarios, preferably universally.

### 3.1.2 Reducing Signaling Round-Trips

One avenue towards more efficient mobility management is a reduction in the number of signaling round-trips that mobile nodes and their correspondent nodes or mobility anchors must go through for a binding update. This type of optimization has in particular been used to replace the long home address test in Mobile IPv6 route optimization by cryptographic mobile-node authentication that can be accomplished in a single round-trip. A natural approach is to accomplish this with a pair of secret keys that are preconfigured into mobile and correspondent nodes, such as in [103]. The mobile node can then send an authenticated Binding Update message directly without having to pursue a home address test first. A disadvantage of using shared keys is that it is applicable only between nodes with a pre-existing relationship. And for those nodes which do have a relationship, the technique requires some configuration for setting up the keys. The existence of shared keys also does not replace reachability verification, leaving the additional round-trip that this requires.

Crypto-based identifiers [84] allow a mobile node to authenticate itself for a binding update without pre-configured credentials. The technique cryptographically and verifiably ties an identifier of the mobile node to a public/private key pair, thus enabling the mobile node to authenticate by proving its knowledge of the private key. Crypto-based identifiers provided the basis for cryptographically generated IP addresses [12], a technique that endows an IP address with the properties of a crypto-based identifier. Where a cryptographically generated IP address is used as a Mobile IPv6 home address, the mobile node can be efficiently authenticated to an unacquainted correspondent node [97]. Reachability verification is still required, however.

### 3.1.3 Localization

Another way to speed up binding updates is to reduce the round-trip times for signaling messages. This can be accomplished by limiting mobility support to a geographical or topological region, and placing the mobility anchor close to mobile nodes. Such *localized mobility management* [132, 110, 121, 50] reduces handover delays and handover-related packet loss. Access links visited by the mobile node are usually part of the same administration, or they belong to different administrations that have agreed to provide roaming service to each other's mobile nodes. Localized mobility management is distinguished from *global mobility management,* which handles movements between access networks that may be administratively separate.

Figure 3.1 depicts a scenario where the mobile node moves across four access links, $L_1$ through $L_4$, belonging to three administratively different access networks, $A$, $B$, and $C$. Each access network has its own local mobility anchor, denoted in the figure

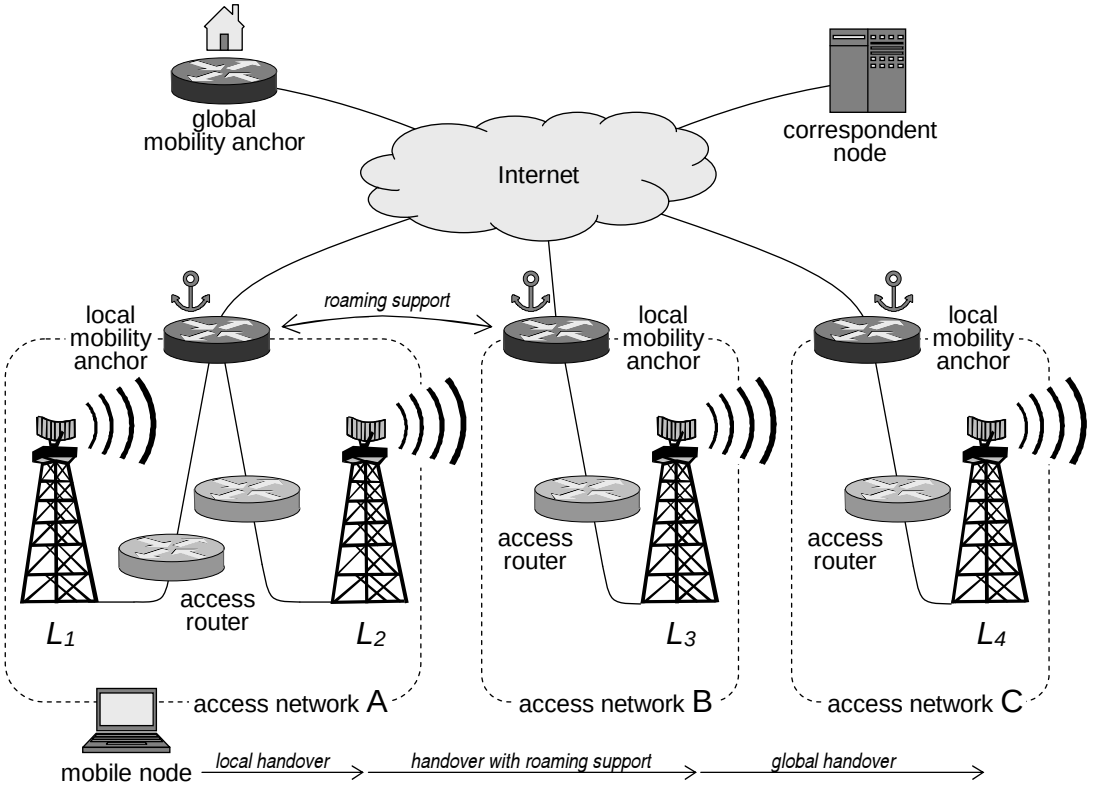



Figure 3.1: Localized mobility management

by a black router with a small anchor symbol attached to it. The mobile node's handover from access link $L_1$ to $L_2$ is managed locally because both of these access networks are from domain $A$. All packets to and from the mobile node go through the same mobility anchor. Access link $L_3$ is from a different administrative access network, $B$. But the handover to $L_3$ can still be handled on a local basis due to a roaming agreement between $A$ and $B$. Access link $L_4$ belongs to yet another administrative access network, $C$, which does not offer roaming service. The mobile node must hence rely on a separate protocol for global mobility management to handle this last handover.

The advantage of localized mobility management is that, due to the relatively short distance between the mobile node and the mobility anchor, the time it takes the mobile node's signaling messages to reach a mobility anchor in the event of a handover, and the latency of the first packet to reach the mobile node's new point of attachment, are relatively small compared to what they generally are in global mobility management. On the other hand, localized mobility management suffers from a number of disadvantages. Deployment costs are increased since the proximity of mobility anchors to mobile nodes requires a higher number of mobility anchors to cover a region of similar extent. An additional mechanism is required to hand a mobile node over from one mobility anchor to another, with the unfortunate consequence of increased protocol overhead. Moreover, due to the close coupling of mobility anchors and access links, localized mobility support can hardly be expected to be available universally. This limits its applicability. Local mobility management per se also does not redundantize reachability verification.

### 3.1.4 Route Repair

The idea of route repair is to enable a mobile node that is in the process of handover to communicate via both the old and the new access router until the packet flow has been fully redirected to the new point of attachment. The mobile node can thus update its binding at a correspondent node or mobility anchor while sending and receiving packets through either the old or the new access router. Typically, the mobile node switches to sending packets from the new IP address once it has initiated the binding update. Packets that are still in flight towards the old IP address are nonetheless continued to be received through the old access router. For route repair to work, the old access router must be informed about the mobile node's new IP address. If the mobile node leaves the old access link without telling its new IP address, it must signal this information back subsequently. Alternatively, the mobile node may be able to proactively determine its new IP address before it moves.

Route repair is usually combined with packet buffering at the old or new access router. If the mobile node informs the old access router about its new IP address prior to handover, the old access router can forward incoming packets for the mobile node directly. Otherwise, the packets must be buffered at the old access router until the mobile node's new IP address is known.

One way to realize route repair is by setting up an individual route for the mobile node between the old and new access router. However, this approach requires support from all routers on the path between the access routers. A more practicable way to implement route repair is through a bidirectional tunnel between old access router and the mobile node, or between the old access router and the new access router. This is how the route repair extensions for both Mobile IPv6 [70] and Mobile IPv4 [69, 77] work. Media Independent Pre-Authentication [41] is a framework for tunnel-based route repair that can be combined with any mobility protocol. It also incorporates a means for mobile nodes to authenticate themselves to a new access router in advance.

Figure 3.2 illustrates the concept of route repair for the case where the tunnel is between the old access router and the mobile node. The mobile node uses route optimization, so it will register its new IP address directly with the correspondent node. In the meantime, the route repair mechanism enables the mobile node to communicate via both the old and the new point of attachment.

Route repair can completely eliminate handover-related packet loss provided that access routers are endowed with sufficient buffer space. Handover delays at the IP layer are close to the link layer handover delay. The propagation latency for packets to pass through the inter-access-router tunnel is subsumed by the link layer handover delay when the tunnel is set up in proactive manner, but otherwise adds to the IP layer handover delay. A suitable access network topology can reduce tunnel propagation latencies.

The shortfalls of route repair are a high operational overhead for access routers in terms of packet buffering and forwarding. The repair further causes a temporarily prolonged route for regular traffic that leads to higher consumption of network resources. If forwarded packets from the old access router are delivered in parallel with packets that arrive directly at the new access router, packet reordering may

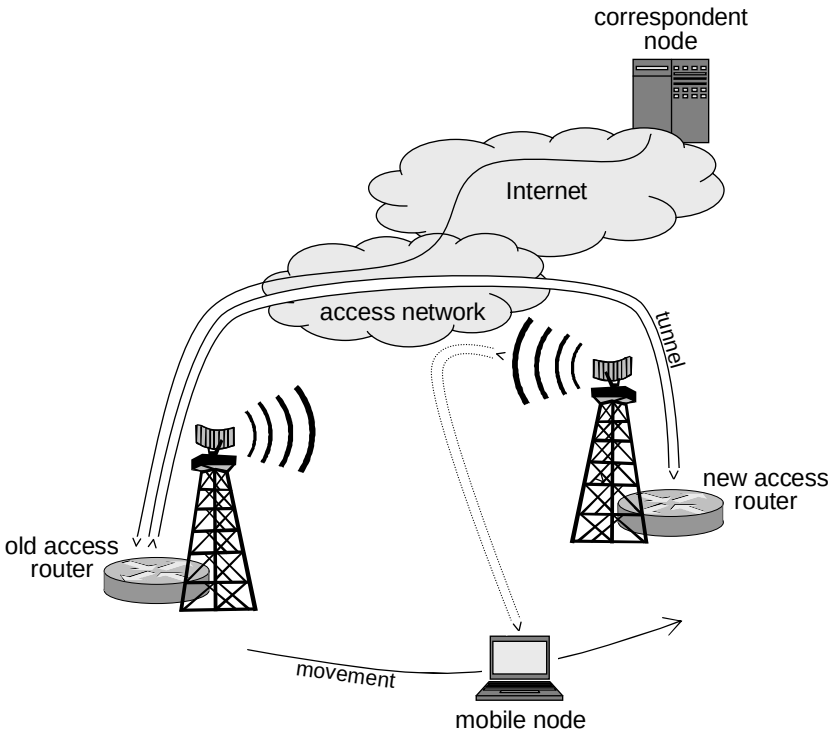



Figure 3.2: Optimizing mobility support with route repair

interact negatively with congestion control algorithms in transport protocols. This has in particular been found to happen for TCP [55]. Route repair further requires that access routers can trust a mobile node with respect to a claimed new IP address. A malicious node could otherwise misuse the packet buffering service to waste substantial memory in access routers and potentially cause denial of service. For the same reason, a trust and security relationship is required between access routers. These requirements may be easy to meet within administratively contiguous access networks—as is the case in the example of figure 3.2—, or across cooperating access networks from separate administrations. But they defeat the applicability of route repair for global mobility management.

### 3.1.5 Packet duplication

Another mechanism to reduce handover delay and handover-related packet loss is for a mobile node's correspondent node or mobility anchor to duplicate payload packets temporarily during handover. One copy is sent to the mobile node's old IP address, and another copy is sent to the new IP address. This mechanism is commonly called *bicasting* [76]. The mobile node requests bicasting with a proactive binding update for its prospective new IP address prior to leaving its old access link. It stops bicasting with another binding update that it sends when it arrives at the new access link.

Bicasting is illustrated in figure 3.3, where a global mobility anchor duplicates packets for the mobile node. Packets are in this case sent through two bidirectional tunnels between the mobile node and the mobility anchor, one to each point of attachment. For route optimization, the responsibility for duplicating the packets is

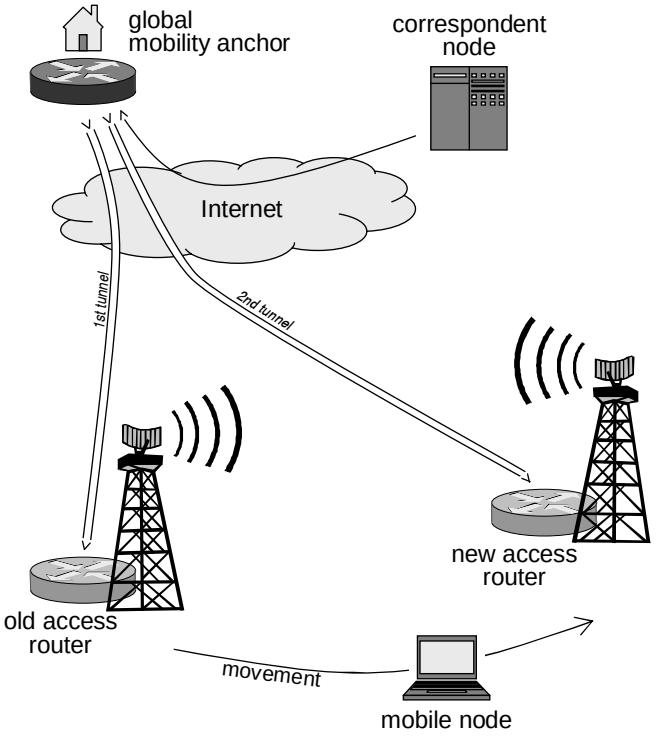



Figure 3.3: Optimizing mobility support with bicasting

with the correspondent node. Bicasting can further be combined with localized mobility management, in which case the local mobility anchor is in charge of bicasting.

Bicasting can effectively reduce handover delays and handover-related packet loss. Both metrics become a function of only the link layer handover delay and of the difference between the round-trip times on the old and the new transmission path. On the other hand, as with route repair, part of this efficiency benefit may be vitiated if packet reordering at the mobile node interferes with transport protocol operation. The duplication of packets during handover furthermore causes additional processing overhead at the correspondent node or mobility anchor, as well as increased consumption of network resources during handover. And finally, a trust relationship is required between a mobile node and the correspondent node or mobility anchor to allow packet redirection to the new IP address without previous reachability verification.

### 3.1.6 Dual Network Attachment

An ability to attach to two access points simultaneously enables a mobile node to continue communications via its old IP address while at the same time performing a binding update with a correspondent node or mobility anchor to redirect traffic to the new IP address. Provided a sufficiently long period during which the signal strength of both access points is strong enough, handover-related packet loss can thus be completely eliminated. A handover delay occurs only if the round-trip time on the new transmission path is longer than the round-trip time on the old transmission path since this would result in a pause in packet delivery. Figure 3.4 shows a mobile node that temporarily attaches to two wireless access links via separate interfaces as it moves from one point of Internet attachment to another.

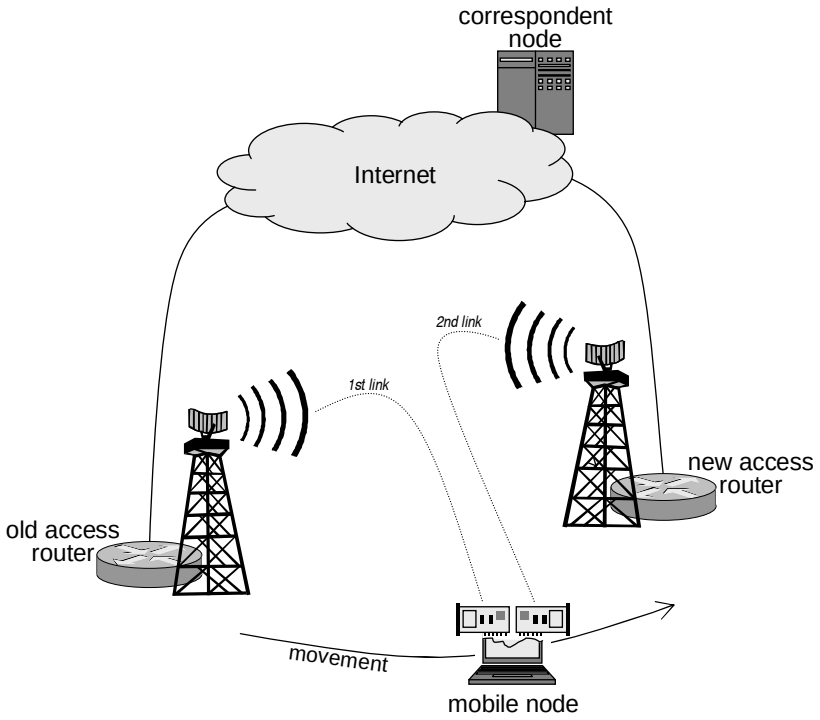



Figure 3.4: Optimizing mobility support with multiple interfaces

In some cases, a single physical interface may be multiplexed across different access points [25]. A single interface is then sufficient for dual network attachment. For most access technologies, however, dual network attachment can be achieved only with two interfaces of the same type [14, 22]. This increases hardware costs and energy consumption, especially when the mobile node has interfaces for different access technologies to support vertical handovers across heterogeneous technologies [1]. Software-defined radios may eventually co-locate multiple access technologies on a single chipset [26], helping to build mobile nodes both cheaper and with better support for heterogeneous handovers. But even then would a mobile node require two software-defined radios if it was to be able to attach to two access link simultaneously.

The full benefit of dual network attachment can be obtained only where the coverage of neighboring access points overlaps to a sufficient extent so that the binding update can complete while connectivity with the old access point is still up. Given a trend towards smaller coverage of any single access point [42], and given that mobile nodes may move at high velocities in cars or trains [48], protocol optimizations are likely to be necessary despite multi-attachment capabilities in order to ensure successful completion of handover procedures within the overlap region.

### 3.1.7 Discussion

The foregoing survey has shown that related work on optimized mobility support seeks either to reduce the delay for binding updates (by decreasing the number of signaling round-trips required, or by reducing the round-trip times of signaling messages through localization), or to salvage payload packets that would otherwise get lost during a handover (through route repair, bicasting, or dual network attachment). The increase in handover efficiency that is thereby achieved comes at the

cost of considerable operational overhead or deployment expenses, which is likely to be a deterrent for global deployment and universal applicability. The new access network services for localized mobility management and route repair furthermore require business relationships between neighboring access network operators. This constitutes yet another hurdle for universal applicability. A low-cost optimization that still effectively improves handover efficiency currently does not exist.

Moreover, none of the existing mobility optimizations addresses the adverse impacts of reachability verification. Reachability verification as built into protocols today exhibits two important disadvantages:

1. The latency involved in the end-to-end token exchange amounts to one round-trip time between the correspondent node and the mobile node. This latency is additional and cannot be subsumed by other handover-related activities. The correspondent node can start the verification only after it has been informed about the mobile node's IP address change, and yet no payload packets may be sent to the new IP address until it becomes verified.

2. Reachability verification can, by definition, only be performed at a time the mobile node is already present at the new IP address. This makes the technique incompatible with proactive mobility management.

Some mobility protocols and optimizations [99, 63, 121, 70, 103] simply omit reachability verification—perhaps because the research community was not yet aware of the threat of redirection-based flooding at the time these proposals were made—, or they circumvent reachability verification by demanding a trust relationship between mobile nodes and their correspondent nodes or mobility anchors. The former is unacceptable as the analysis in the next section will show, and the latter again limits the applicability of a mobility protocol. Trust requirements are in particular inappropriate for use on public servers with a broad audience. Even a business relationship is in general insufficient as a basis for trust. For example, a mobile-phone operator may be able to configure subscribers with secret keys for authorization to a particular service, but it may not be able to vouch that all subscribers use this service in a responsible manner.

An alternative to today's use of reachability verification that is more efficient, but as simple and suitable for global deployment, does currently not exist. This thesis contributes such an alternative.

## 3.2 Significance of Reachability Verification

Given a number of existing avenues to flooding that an attacker can choose from in today's non-mobile Internet, one may wonder if the introduction of redirection-based flooding attacks as one additional possibility makes any difference. The following elaboration explains that redirection-based flooding attacks are indeed of concern due to a possibility to substantially amplify the flooding volume at very low costs on the attacker side. Existing flooding attack types are either limited to much less amplification or demand much more substantial investments for the attacker.

### 3.2.1 Existing Types of Flooding

All types of flooding attacks exploit the Internet's vulnerability to deliver packets to a destination node without asking whether the destination node is actually willing to receive these packets. Delivered packets consume bandwidth at the destination node's point of attachment as well as processing capacities and memory on the destination node itself. The victim of a flooding attack is forced to spend these resources, which may limit its ability to pursue any legitimate communications. Four basic types of flooding attacks can be differentiated in today's non-mobile Internet:

- In a *direct flooding attack*, the attacker itself generates the flooding packets and sends them to the victim.
- In a *reflection attack* [101, 24, 23], the attacker tricks a third node, the *reflector*, into sending flooding packets to the victim. It achieves this by sending the reflector packets with the IP source address set to the victim's IP address.
- *On-path flooding attacks* require the attacker to be on the path from the correspondent node to the victim. From this position, the attacker establishes a connection with the correspondent node on behalf of the victim and initiates a large data file to be downloaded to the victim.
- The attacker in a *distributed denial-of-service attack* [79] takes over control of other nodes by compromising them with viral software, which it distributes by conventional means. It typically exploits an implementation vulnerability in the target node's operating system to inject the software. When it is time to attack, the infected nodes, as such called *zombies,* jointly send packets to the victim. This may be triggered automatically by the zombie software, or remote-controlled by the attacker. The attacker itself usually does not send flooding packets to the victim.

Existing types of flooding attacks differ in the amount of effort that a flooding attacker has to invest in order to impose a certain damage level upon the victim. Two kinds of effort need to be considered: The effort required to prepare and set up an attack, and the effort to actually produce flooding packets throughout the attack. Much of the first kind of effort goes into the development and distribution of the viral software. The preparation required to launch a direct flooding attack, reflection attack, or on-path flooding attack is comparably low since a network of zombies is not required. A metric to measure the second kind of effort is *amplification,* defined as the ratio between the total flooding volume delivered to the victim and the data volume that the attacker itself must generate in order to initiate and sustain the attack. There is no amplification for direct flooding attacks, and usually none or only very limited for reflection attacks. So in order to notably harm the victim, the attacker itself must therefore send more packets than the victim can handle. Bandwidth is an ample resource for many attractive victims [31], however, which limits the practicality of direct flooding and reflection attacks. On-path flooding attacks provide more substantial amplification because it is the correspondent node rather than the attacker which generates the flooding packets. The amplification in distributed denial-of-service attacks depends on the size of the zombie network, which can be substantial [56].

Attacks further vary in whether they require the IP source address in flooding packets to be spoofed. Such *IP spoofing* [23] may not be possible if routers on the path from the attacker to a correspondent node verify the topological correctness of the IP source addresses in to-be-forwarded packets and drop packets where the IP source address is incorrect. Ingress filtering [47, 126] are two techniques that can be used for this purpose. IP source address verification provides reasonable protection only when deployed ubiquitously throughout the Internet or at least to a wide extent because it must be enforced on the attacker side and fails to protect a potential victim. A reasonable level of deployment would make on-path flooding attacks more difficult because not only would the attacker have to find network access on the path from a correspondent node to the victim, that network access would also have to be without IP source address verification. The threat of distributed denial-of-service attacks would be reduced as well because it would become more difficult for an attacker to find zombies with unfiltered network access. Direct flooding attacks and reflection attacks are more robust to partial deployment of IP source address verification because the attacker can choose an access network that does not filter spoofed IP source addresses. Recent measurements [86] of considerable denial-of-service activity utilizing false IP source addresses indicates that ample opportunity for IP spoofing remains in today's Internet.

### 3.2.2 Utility of Redirection-Based Flooding

Redirection-based flooding attacks are attractive from an attacker's perspective because they require comparably little preparatory overhead and still have a potential for high amplification. There are three indications that corroborate this:

1. The packet flow generated by a correspondent node can be much larger than the flow of packets the attacker sends to that correspondent node. The attacker may be required to forge transport layer acknowledgments to uphold a packet flow. But acknowledgments are typically much smaller compared to the packets they pertain to. For example, based the common upper packet size limit of 1500 byte [35], a packet carrying a TCP segment is 25 times larger than a typical TCP acknowledgment. Besides, an acknowledgment may confirm the receipt of more than one packet.

2. A correspondent node can be a powerful Web or media server, in which case its Internet attachment may be broader in bandwidth than the attacker's.

3. Multiple correspondent nodes can be tricked into flooding a single victim.

To maximize amplification, a flooding attacker may combine redirection-based flooding and distributed denial of service. Each zombie is then programmed to redirect one or multiple correspondent nodes' packet flows to the victim. A zombie itself does not necessarily generate any flooding packets. The attacker can thus benefit from correspondent nodes that it was unable to infect with its viral software. This can be an advantage for the attacker if those correspondent nodes have higher-bandwidth Internet attachment than an average zombie. For example, a Web or media server may be an attractive target for the attacker, but would typically be well protected by up-to-date anti-virus software. Private desktops may be easier to infect [53], but

they are usually more sparsely attached to the Internet, in particular with respect to upstream bandwidth. Increasingly popular DSL subscriptions [111], for instance, usually offer much smaller upstream than downstream connectivity.

Another advantageous property of redirection-based flooding attacks is that they make it easy for an attacker to find a point of attachment to launch the attack from, and a set of correspondent nodes that can be gamed into contributing to the attack:

1. The attacker can launch the attack from a location aloof any flow of flooding packets. This is in particular important when the attacker attaches to a half-duplex link, where downstream flooding packets would compete with packets that the attacker sends upstream, such as fake acknowledgments.

2. Redirection-based flooding attacks enable the perpetrator to recruit correspondent nodes anywhere in the Internet. The attacker does not have to be on the path between a correspondent node and the victim. The mobility support required for the attack is expected to be implemented by most mobile and stationary nodes as a standard feature in the future.

A last property of redirection-based flooding attacks that is of potential interest to an attacker is that they may enable the attacker to hide its identity from the flooding packets despite the deployment of protection mechanisms against IP spoofing on the network side. If the mobility protocol includes multi-homing support and does not identify the attacker in redirected packets, then the attacker can uphold a flow of redirected packets through bogus acknowledgments without spoofing IP spoofing as explained in section 2.3.5. This may be attractive for the attacker in view of ongoing efforts [102] to eliminate IP spoofing through network support on a global basis. It could be attractive even if the attack does not yield any amplification because it would facilitate a sustained reflection attack that does not expose the attacker's identity despite network-side IP spoofing prevention.

On the other hand, redirection-based flooding attacks also have their own shortfalls, which may limit their usefulness for an attacker in certain situations:

1. A redirection-based flooding attack is likely to stall unless the attacker sends acknowledgments on behalf of the victim. This may require the attacker to spoof its IP source address depending on the mobility protocol.

2. Since the attacker is usually not on the path from the victim to a correspondent node, it cannot intercept error messages that the victim may send to the correspondent node in response to receiving flooding packets. The error messages may cause the correspondent node to abort the redirected packet flow.

3. As noted above, a mobility protocol may identify the attacker in redirected payload packets, keeping the attacker from staying incognito.

4. Direct flooding and distributed denial-of-service attacks may gain additional leverage if flooding packets mimic legitimate requests and trigger expensive operations at the victim. TCP SYN attacks [23], for example, use bogus TCP SYN segments to exhaust memory and processing capacities at a server. No

similar effect is typically possible for redirection-based flooding attacks, where the flooding packets are of no meaning to the victim and hence simply get dropped.

Overall, redirection-based flooding attacks feature properties that are likely to be attractive to a flooding attacker, and that other types of flooding attacks do not have. It is hence important to preclude redirection-based flooding attacks as a new flooding vehicle through appropriate protection in mobility protocols.

## 3.3 Protection Alternatives Against Redirection-Based Flooding

While the previous section has corroborated the need to protect against redirection-based flooding attacks, doing it by standard reachability verification brings about undesired performance penalties. These call for more efficient, yet as secure protection against redirection-based flooding. Several solutions are conceivable, some of which have been publicly proposed. This section analyzes the strengths and weaknesses that these alternatives have. It begins with a definition of the assessment criteria to ensure fair comparison.

### 3.3.1 Assessment Criteria

The first four criteria in assessing the suitability of mechanisms to protect against redirection-based flooding attacks relate to a mechanism's security properties:

1. *Reliability of IP address verification* — It should be impossible, or at least sufficiently impracticable, for an attacker to pass reachability verification for a spoofed IP address and redirect packets to that IP address.

2. *Early availability of new IP address* — A mobile node's new IP address should be available as a destination for payload packets from the correspondent node as early as possible. Negative impacts on application quality due to long reachability verification delays can thus be avoided. For efficient reactive mobility management, the new IP address should therefore become available once autoconfigured on a new access link.

3. *Support for proactive mobility management* — Where the mobile node is able to obtain an IP address for a prospective new access link while still connected to the old access link, reachability verification for the new IP address should not preclude proactive mobility management. The correspondent node should be allowed to begin using a new IP address already shortly before the mobile node becomes reachable there.

4. *Verifiability for the correspondent node* — The correspondent node should either verify the mobile node's IP address itself, or there should be a way for the correspondent node to determine whether the IP address has undergone verification and what the result of that verification was.

A mechanism is further assessed based on its administrative and trust prerequisites, which directly translate into its suitability for global deployment:

5. *No relationship between mobile and correspondent node* — IP address verification should function without a special relationship between the mobile node and the correspondent node, be it in terms of security, trust, or administration.

6. *Independence from special infrastructure* — No special infrastructure support should be required for the verification of a mobile node's IP address. This includes third parties trusted for certification purposes, like a public-key infrastructure.

Finally, each mechanism is rated according to a set of technical criteria:

7. *Low overhead* — The mechanism should introduce no or only minor extra signaling overhead since bandwidth is a sparse resource for many wireless access technologies. Mobile nodes may further be confined to scarce memory and non-volatile storage, so state requirements should be small as well. Computationally expensive algorithms, such as those involving public-key cryptography, are inappropriate for mobile nodes with limited processing and battery capacities and should hence be avoided.

8. *Facile integrability into mobility protocols* — The mechanism should be generic enough to be easily integrable into different mobility protocols. Changes in protocol operation should be limited.

9. *Easy implementation and deployment* — IP address verification should be inexpensive to implement and deploy. Requirements for hard- or software upgrades of network entities as well as the introduction of new network entities should be avoided.

The preference to avoid a special relationship between mobile and correspondent nodes, and to also remain independent from special infrastructure, is based on the desire to permit global use of an IP address verification mechanism. Just as the standard return routability procedure in Mobile IPv6 does with pure IPv6 support and a home agent alone, other IP address verification mechanisms, too, should not make stronger demands that could limit their applicability.

### 3.3.2 Trusted Communities

One way to avoid the impact of reachability verification is to waive it for a community of mobile nodes that are trusted not to lie about their on-link IP addresses. The concept of trusted communities is applied in Mobile IPv6 home registrations, where the home agent never verifies a care-of address during a binding update. There is also a Mobile IPv6 extension [103] for use of route optimization in trusted communities.

However, the concept of trusted communities is incompatible with global mobility management, which implies that mobile nodes may communicate with correspondent nodes without an a-priori relationship. The technique is in particular inappropriate for use on public servers with a broad audience. But even a trusted community is not necessarily immune against subversion of some of its members by viral software. Infected nodes may exploit the correspondent node's trust for the purpose of redirection-based flooding.

### 3.3.3 Temporary Buffering

Another possible way to reduce the adverse impact of reachability verification is to enable correspondent nodes to buffer payload packets destined to a mobile node during a handover until reachability verification completes. Buffering can reduce the number of packets that are lost during the handover. It functions without special infrastructure and is conceptually easy to integrate into existing mobility protocols.

One disadvantage of temporary buffering is that communications are still disrupted during reachability verification. For a reliable transport protocol like TCP, this may cause spurious retransmission timeouts. Real-time applications with a requirement for timely packet delivery typically operate over unreliable transport protocols such as UDP, which do not retransmit. But the additional delivery delay for buffered packets may render the packets stale when eventually processed by the receiving application. A second disadvantage of packet buffering is memory requirements. A correspondent node must devote a potentially significant amount of memory to a mobile node which it may not even trust. This makes the correspondent node vulnerable to denial-of-service attacks that attempt to exhaust its memory. These observations suggest that packets buffering during reachability verification is infeasible for global mobility management.

### 3.3.4 Temporary Redirection to Stable IP Addresses

If the mobility protocol provides a stable IP address for mobile nodes, a correspondent node may send payload packets to the stable IP address when verifying the mobile node's reachability at the on-link IP address. The latency of reachability verification can thereby be masked. In Mobile IPv6, for instance, a mobile node could deregister its binding at the correspondent node shortly before, or immediately after a handover, and so have the correspondent node direct subsequent payload packets to the home address until the registration for the new care-of address completes.

On the other hand, the process of redirecting payload packets to the stable IP address involves itself some latency. Updating the binding at the global mobility anchor to the new on-link IP address takes a round-trip time between the mobile node and the global mobility anchor. And for the mobile node to receive payload packets from the correspondent node via its stable IP address, it takes a one-way time from the mobile node to the correspondent node to inform the correspondent node that it should divert subsequent payload packets to the stable IP address, plus a one-way time from the correspondent node to the global mobility anchor at the stable IP address and a one-way time from the global mobility anchor to the mobile node to deliver the first redirected payload packet. The delay this involves may be considerable, especially if the global mobility anchor is topologically distant from the mobile node and the correspondent node. Finally, once reachability verification is complete, the correspondent node must be requested to redirect payload packets from the stable IP address to the new on-link IP address. This takes another round-trip time between the mobile node and the correspondent node until the first payload packet has arrived at the mobile node via the direct path. Additional signaling may be necessary to authenticate any of the aforementioned binding updates, such as a home address test in Mobile IPv6.

Depending on the flexibility permitted by the mobility protocol, it may be possible to parallelize the binding updates for the global mobility anchor and the correspondent

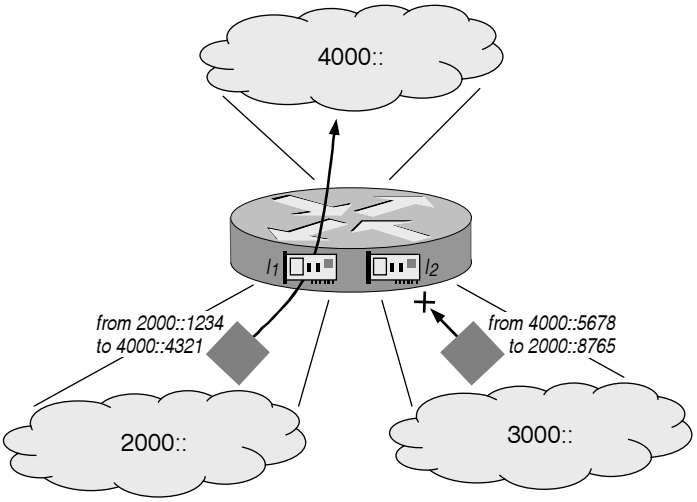



Figure 3.5: Operation of ingress filtering

node. Some mobility protocols require both tasks to be performed in sequence, however. For example, the Mobile IPv6 RFC stipulates that the binding update during a correspondent registration or deregistration be deferred until the respective home registration has been acknowledged.

Another disadvantage of temporary redirection to a stable IP address is an additional propagation latency for payload packets that go via the global mobility anchor since this may have an adverse impact on transport protocol performance and application quality. Transport protocols typically use round-trip time measurements as a basis for their retransmission and congestion control mechanisms, so the increased propagation latency of diverted packets may cause spurious retransmission timeouts on the sender side. Round-trip time measurements become subverted already by mobility in general, as the properties of the end-to-end path may change in case of a handover. But the problem gets aggravated when a packet flow is temporarily redirected from a direct routing path to a path via a global mobility anchor.

### 3.3.5 IP Source Address Filtering

A frequently proposed mechanism for reachability verification is to rely on the access network infrastructure to verify a mobile node's claimed on-link IP addresses. *Ingress filtering* [47] is the currently most widely supported such technique. Routers inspect the IP source address prefix in each packet they forward, and they drop a packet if the subnet prefix has not been allocated to the network where the packet is coming from or cannot be reached via that network. In the simplest form, filters are set up manually. A common automated mechanism is *Unicast Reverse Path Forwarding* [126]. Given a to-be-forwarded packet, this technique calls for a router to check in its routing table whether the route leading back to the subnet prefix of the packet's IP source address points to the interface through which the packet arrived. If so, the packet is forwarded; otherwise, the packet is dropped. The router in the illustration of figure 3.5 applies Unicast Reverse Path Forwarding. It forwards a packet with IP source address 2000::1234 received from interface $I_1$, but drops a packet with IP source address 4000::5678 received on interface $I_2$.

An important advantage of ingress filtering is that it checks IP addresses without the latency of end-to-end reachability verification. The technique is also stronger in that it provides protection against attackers on the path between a correspondent node and the claimed IP address, where end-to-end reachability verification may wrongly classify a spoofed IP address as being correct. On the other hand, ingress filtering suffers from a number of disadvantages that discourage its use as an alternative to reachability verification. These are as follows:

1. *Coarse-grained filters* — Ingress filtering is most effective on access routers. Routers that use the technique further upstream necessarily apply less and less precise filters as subnet prefixes are aggregated inside the network. Given that ingress filtering today is primarily used at the border between an access network and an Internet service provider [15], it necessarily provides only coarse-grained protection.

2. *No coverage of packet payload* — Some mobility protocols require or permit new IP addresses to be communicated within the payload of signaling packets. Ingress filtering fails to detect if those IP addresses are spoofed because the technique only verifies the IP source addresses in IP headers. This affects, for example, the mobility extensions of the Host Identity Protocol. Mobile IPv6 usually interprets the IP source address of a mobile node's Binding Update message as the new care-of address, but the presence of an Alternative Care-of Address option in the message overwrites this default behavior.

3. *Incompatibility with proactive mobility management* — Given that ingress filtering does not cover the payload of a packet, it may be tempting to change mobility protocols so that new IP addresses are always carried as an IP source address in the signaling packet's IP header. This, however, would preclude proactive mobility management, where a registered IP address does not topologically belong to the access link from which the binding update is pursued.

4. *Need for universal deployment* — Ingress filtering can prevent an attacker from spoofing its IP source addresses from a particular point of attachment. But since it must be enforced on the attacker side, the technique cannot protect potential victims of redirection-based flooding attacks unless it is deployed universally, or at least to a wide extent. Correspondent node, too, must rely on the prevalence of ingress filtering given that they receive no feedback on whether the technique is in use at the mobile-node side.

5. *Missing incentives* — There is little incentive for access network operators and Internet service providers to deploy ingress filtering other than conscientiousness [6]. Legal or regulatory prescription as well as financial motivation does not exist. On the contrary, deploying ingress filtering for multi-homed access networks complicates network topology design and may hence rather deter an access network operator from deploying the technique. A corrupt operator or Internet service provider might even have a financial incentive not to deploy the technique if an attacker is willing to pay for the relay of IP packets with false IP source addresses. A similar issue exists with email spam, which also typically uses bogus or unreachable source addresses.

This discussion sheds doubt on whether ingress filtering can serve as a feasible replacement for end-to-end reachability verification. Financial incentives, technical difficulties, or administrative hurdles may prevent a large or even universal deployment of the technique and hence undermine its protectiveness.

### 3.3.6 Heuristics

Another conceivable approach to the verification of on-link IP addresses is to implement a heuristic for misuse detection on the correspondent node side. A heuristic can prevent, or at least effectually discourage redirection-based flooding attacks. The challenge is to find a reliable heuristic, however: On one hand, the heuristic must be sufficiently rigid to quickly respond to malicious intents on the mobile-node side. On the other hand, it should not have a negative impact on a legitimate mobile node's communications. A legitimate mobile node may at times behave in a suspect manner, changing its on-link IP addresses rapidly when moving at a high speed or ping-ponging between different points of attachment. This may result in IP addresses not being successfully verified. It may therefore be hard for a correspondent node to determine whether a mobile node's behavior is legitimate, or whether it indicates an ongoing or imminent attack.

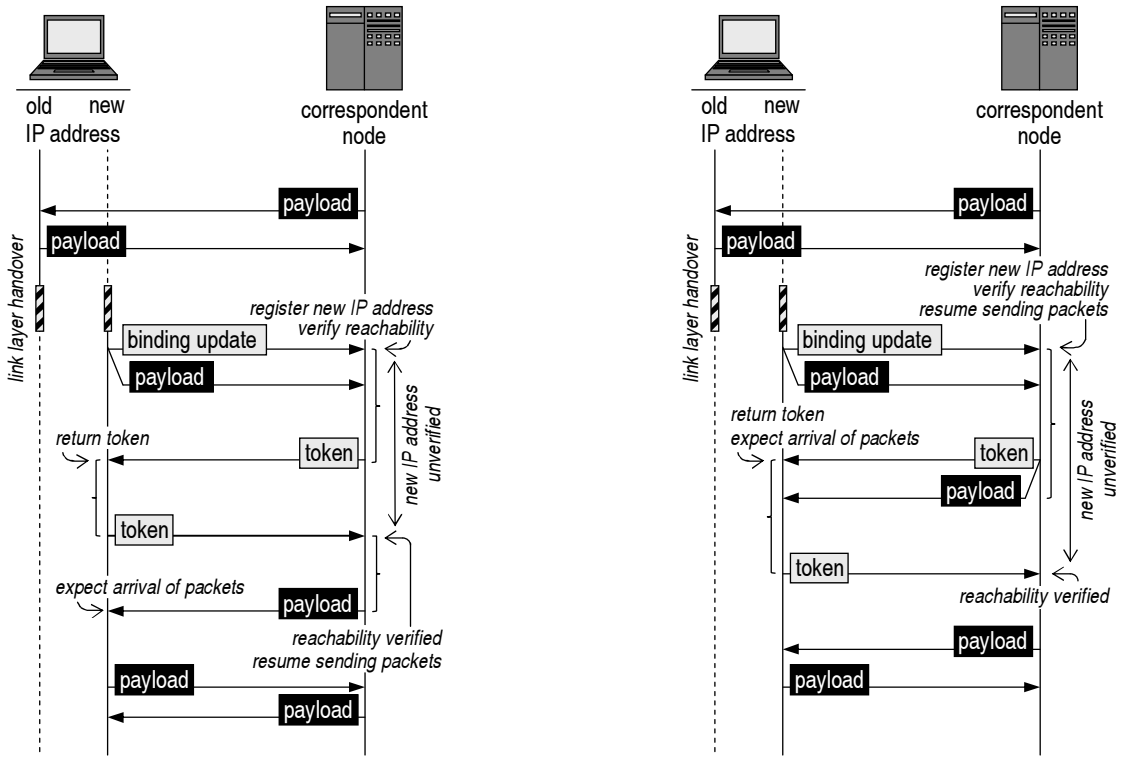
Another problem with heuristics is that a correspondent node is unable to prevent misbehavior, but can only react to it. If sanctions are imposed quickly, attacks may simply not be worthwhile. Yet premature measures should be avoided as well. An attacker may further be able to use multiple identities or exploit multiple correspondent nodes for an attack, thus complicating the design of a feasible heuristic further.

## 3.4 Proposed Solution

All of the previously discussed alternatives to reachability verification have been found to suffer from disadvantages that make them difficult to use for global mobility management. This comes as a result of special assumptions on the relationship between mobile and correspondent nodes, prerequisites for special infrastructure that does not exist universally, or increased requirements on memory resources. None of the alternatives provides the simplicity of reachability verification that would render it suitable for global deployment.

This poses the question of whether standard reachability verification could be modified such that it no longer precludes bidirectional packet exchange temporarily. Performing reachability verification proactively prior to handover is possible only when the mobile node has two interfaces, so that it can obtain a new IP address on the new access link and have the correspondent node verify it while at the same time continuing communications on the old access link. This again would limit the optimization to a subset of mobile nodes since some mobile nodes might be too small, too low-cost, or too low-powered to carry two interfaces per supported network technology.

So if reachability verification is to remain reactive, but still should allow for immediate bidirectional communications after a handover, then it must take place in parallel with regular payload packet exchange. Like standard reachability verification, such *concurrent reachability verification* calls for a correspondent node to probe




(a) Standard reachability verification

(b) Concurrent reachability verification

Figure 3.6: Standard vs. concurrent reachability verification

a mobile node's new IP address by sending there a piece of unguessable data that the mobile node must return. It is different, however, in that the correspondent node is allowed to send payload packets to the new IP address while that IP address is still unverified. Concurrent reachability verification permits prompt bidirectional communications after a handover. It also facilitates proactive mobility management. But the technique is secure if and only if the intermediate phase during which the new IP address is unverified can be protected from misuse for illegitimate packet redirection.

Figure 3.6 juxtaposes standard reachability verification, in part (a), and concurrent reachability verification, in part (b), in the case where the correspondent node initiates the verification after processing a binding update. In both parts of the figure, the mobile node initiates a binding update once the link layer handover is complete. It also continues transmitting payload packets to the correspondent node at that point. When the correspondent node receives the binding update, it starts reachability verification by sending an unguessable token to the mobile node's new IP address. The correspondent node in (a) of the figure waits until it receives the token back from the mobile node before it resumes payload packet transmission. Bidirectional communications resume only then. The correspondent node in part (b) supports concurrent reachability verification. It is able to send payload packets to the mobile node once it has processed the binding update.

A common misconception is that the phase during which the new IP address is unverified could be secured by defining a maximum permitted duration, and having the correspondent node block the new IP address if it cannot be successfully verified

meanwhile. The problem with this approach is that a malicious node could repeatedly re-register a spoofed IP address, or toggle between spoofed IP addresses, so as to effectively extend the permitted duration infinitely. Another common misconception is that concurrent reachability verification could easily be supplemented with some algorithm for misuse detection on the correspondent node side. This approach suffers the same problems that have been identified in section 3.3.6.

This thesis proposes the use of concurrent reachability verification in combination with the credit-based approach introduced in chapter 4 as a solution for efficient and secure reachability verification in reactive and proactive mobility management. This combination will be shown to uphold the advantages of reachability verification in terms of universal applicability and deployability, low resource requirements, and easy integrability into various mobility protocols—and hence to constitute a viable alternative to standard reachability verification for improved handover performance.

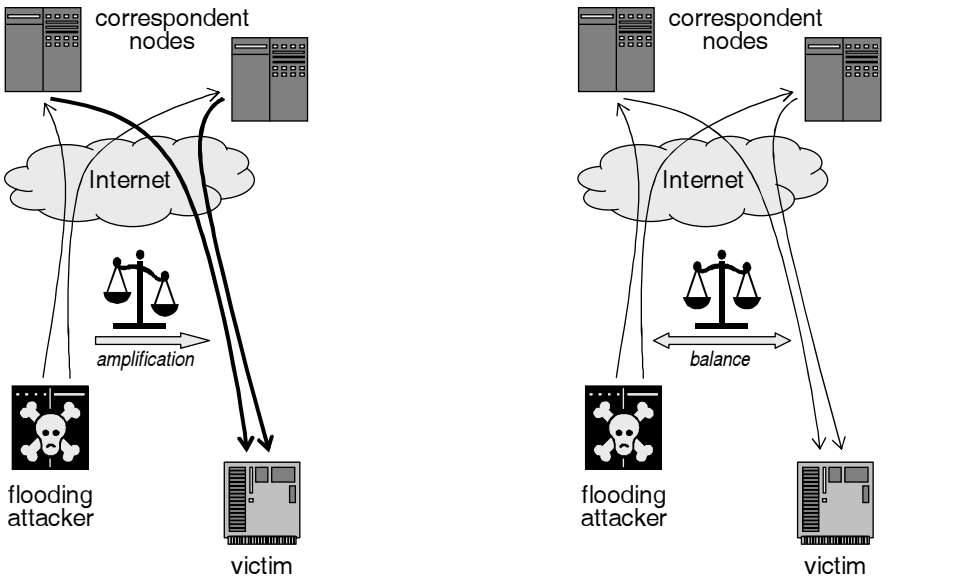
# 4. Credit-Based Authorization

Given the need for a reliable and efficient alternative to standard reachability verification, which is verifiable by the correspondent node, globally applicable, independent of specialized infrastructure, low in overhead, and easy to implement and integrate into existing mobility protocols, concurrent reachability verification was proposed in the previous chapter. It was further shown that concurrent reachability verification requires an additional mechanism to prevent malicious packet redirection while a correspondent node sends payload packets to a new on-link IP address that has not yet been verified. This chapter introduces *Credit-Based Authorization*, a credit-based mechanism that serves this purpose. The objectives in the design of Credit-Based Authorization directly arise from the assessment criteria listed in section 3.3.1.

## 4.1 Approach

Rather than denying packet redirection to an unverified IP address altogether—which is what standard reachability verification does—, the goal in designing Credit-Based Authorization is to permit such redirection, and yet to defeat any attractive properties that illegitimate redirection might have for a flooding attacker. The discussion in section 3.2 has shown that such attractive properties are a potential for amplification and for sustained reflection without amplification, but with the option of hiding the attacker's identity in the presence of network-side IP spoofing prevention.

To prevent amplification, Credit-Based Authorization calls for a correspondent node to monitor the effort that a mobile node spends in communicating with the correspondent node, and to not spend more effort in sending payload packets to an unverified on-link IP address than it has recently seen the mobile node expend. The permission for the correspondent node to send a limited amount of payload packets to an unverified IP address enables immediate resumption of bidirectional communications once the mobile node has registered a new on-link IP address with the correspondent node after a handover. However, if what appears to be a mobile node is in fact an attacker that tricks the correspondent node into redirecting payload




(a) Without Credit-Based Authorization (b) With Credit-Based Authorization

Figure 4.1: Effect of Credit-Based Authorization on redirection-based flooding

packets to the IP address of a victim, Credit-Based Authorization limits the correspondent node's effort such that the redirection does not produce amplification. A flooding attack can therefore be at most as effective as a direct flooding attack.

Figure 4.1 illustrates the effect of Credit-Based Authorization on amplification. Part (a) shows a redirection-based flooding attack in the absence of Credit-Based Authorization. The attacker here tricks two correspondent nodes into sending packets to the same victim, thus producing a flow of flooding packets that is higher in rate and volume than the two packet flows from the attacker to each correspondent node taken together. On the other hand, both correspondent nodes in part (b) of the figure perform Credit-Based Authorization, so together they do not spend more effort in sending packets to the victim than the attacker has spent in communicating with them. The flooding attack in this case hence fails to yield amplification.

To prevent sustained reflection, Credit-Based Authorization calls for a correspondent node to determine the maximum effort for sending packets to a mobile node's unverified IP addresses at the time the mobile node registers such an IP address, and to not increase this limit any more until all of the mobile node's IP addresses have become verified. It so becomes impossible for an attacker to trick a correspondent node into durably sending packets to a spoofed, unverified IP address.

In consequence, Credit-Based Authorization eliminates the two salient properties of redirection-based flooding attacks that constitute the attacks' usefulness from an attacker's perspective. Credit-Based Authorization thereby ensures that a flooding attack can never be more effective than a direct flooding attack. An attacker can usually[1] flood its victim directly anyway, so the additional effort to set up and coordinate a redirection-based flooding attack no longer pays off for the attacker.

[1]An exception is the special case where the victim of a flooding attack is inside a firewall-protected intranet, and the correspondent node is in a demilitarized zone. If the correspondent node can reach the victim, as is the case in one possible firewall configuration, then an attacker on the Internet could trick the correspondent node into redirecting a packet flow to the victim,

Credit-Based Authorization thereby renders redirection-based flooding attacks futile and eliminates them as a realistic flooding vehicle.

### 4.1.1 Credit Concept

The measuring and limiting of effort is technically realized through *credit*, which a correspondent node maintains to put its own effort in relation to the effort of a mobile node. Starting with an initial credit of zero, the correspondent node assigns the mobile node credit for the effort the mobile node naturally spends by sending or receiving payload packets to the correspondent node, and it reduces the credit in proportion to its own effort for sending payload packets to an unverified on-link IP address of the mobile node. Credit is measured in bytes. The transmission of a big payload packet is thus rated higher than the transmission of a small acknowledgment.

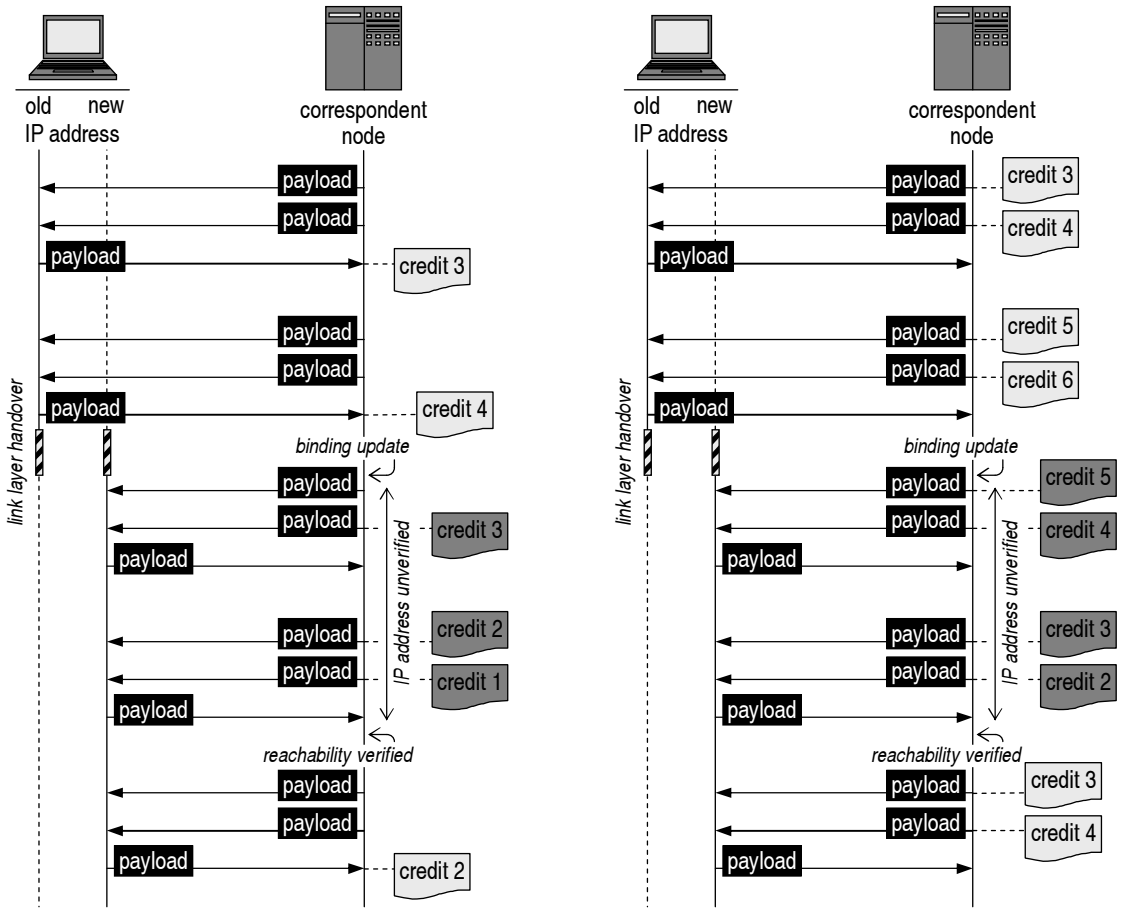
Credit cannot be negative. The correspondent node stops sending to an unverified IP address if a payload packet queued for transmission would cause the mobile node's credit to decrease below zero. By default, such a payload packet is dropped. Although there is no constant upper credit limit, acquired credit reduces over time according to an aging function that will be defined in section 4.1.2.

The credit concept implies two algorithms that a correspondent node must implement—one for credit assignment and one for credit deduction. The credit deduction algorithm is always the same: Whenever the correspondent node sends a payload packet to an unverified IP address, it reduces the recipient mobile node's credit by the size of this packet in terms of bytes. The correspondent node can continue to send payload packets to the unverified IP addresses as long as the packets would not cause the mobile node's credit to reduce below zero. Meanwhile, the mobile node's reachability at the new IP address is verified. No credit is consumed for payload packets that are sent to a verified IP address. The credit assignment algorithm can be drawn from a much broader solution space. A mobile node expends effort both for sending packets to the correspondent node as well as for receiving packets from the correspondent node. So the correspondent node may give the mobile node credit for payload packet transmission or for payload packet reception. This gives rise to two credit assignment algorithms, which, combined with the common credit consumption algorithm, form two Credit-Based Authorization modes:

1. *Inbound mode* — The correspondent node assigns the mobile node credit for each payload packet byte received from the mobile node while all of the on-link IP addresses in the mobile node's binding are verified. This credit is reduced by each payload packet byte that the correspondent node sends to an unverified on-link IP address.

2. *Outbound mode* — The correspondent node assigns the mobile node credit for each payload packet byte sent to the mobile node, and assumed to be received and processed by the mobile node, while all of the on-link IP addresses in the mobile node's binding are verified. This credit is reduced by each payload packet byte that the correspondent node sends to an unverified on-link IP address.

even though the attacker could not reach the victim directly. However, the recommended way of configuring a firewall is to deny all traffic from a demilitarized zone into the protected intranet.




(a) Inbound mode

(b) Outbound mode

Figure 4.2: Credit-Based Authorization modes

It is conceivable to apply a trade-off between Inbound mode and Outbound mode, where the correspondent node assigns part of the credit for received packets and part of the credit for packets it sends. This thesis focuses on Inbound mode and Outbound mode because the study of a trade-off between the modes would not provide fundamental additional results.

To mitigate the threat of sustained redirection-based reflection attacks, a mobile node is given credit only when none of its on-link IP addresses is currently undergoing reachability verification. A reflection attacker thus cannot acquire new credit for packets that go via a verified on-link IP address, owned by the attacker itself, while the correspondent node sends packets to a spoofed unverified IP address that routes to a victim.[2]

Figure 4.2(a) illustrates the operation of Inbound mode with the simplification that each payload packet is worth exactly one credit. The correspondent node maintains a binding for the mobile node with an associated credit account. The current amount of credit available is indicated on the snippets on the right side of the figure. There is one snippet per change in the credit. The light-shaded snippets show how the

[2]Limiting credit assignment to traffic exchanged via IP addresses in Verified state would fail to mitigate the problem because the attacker earns credit with its own IP address in the described scenario and hence could easily pass reachability verification for this IP address.

credit increases with each payload packet that the correspondent node receives from the mobile node while the registered on-link IP address is verified. At some point, the mobile node undergoes a handover and updates the binding at the correspondent node to a new on-link IP address. The new IP address is initially unverified, so the mobile node's credit reduces by the size of each payload packet that the correspondent node sends to the new IP address, as shown on the dark-shaded snippets. At the same time, the correspondent node does not increase the credit for any payload packets received from the mobile node. The correspondent node would stop sending to the unverified IP address if a payload packet queued for transmission caused the mobile node's credit to decrease below zero, but this situation does not occur in the figure. The mobile node's reachability at the new IP address is meanwhile verified. The mobile node always has an interest in proving its reachability at the IP addresses quickly as the existence of an unverified IP address is costly in the sense that payload packets received via such an IP address consume the mobile node's credit. Once reachability verification completes, the correspondent node again sends payload packets to the mobile node without reducing the mobile node's credit, and the credit grows again with each payload packets that the correspondent node receives from the mobile node.

The correspondent node shown in figure 4.2(b) uses Credit-Based Authorization in Outbound mode—again with the simplification that each payload packet is worth exactly one credit. It assigns the mobile node credit for each payload packet sent to a verified on-link IP address of the mobile node. This credit is supposed to reflect the effort that the mobile node spends on receiving and processing the packets. The correspondent node starts to consume the credit when the mobile node performs a handover and replaces the existing on-link IP address by a new, unverified on-link IP address. The credit is then reduced with each outgoing payload packet until the new IP address becomes verified. After reachability verification, the credit grows again as it did before the handover.

The primary advantage of Inbound mode is that measured effort compares straightaway to the effort that an attacker would have to spend in a direct flooding attack or a reflection attack. Inbound mode is hence the natural approach to protecting against redirection-based flooding. The disadvantage of Inbound mode is that it can limit the performance of applications with asymmetric traffic patterns during reachability verification. The problem occurs when a mobile node receives much more data from the correspondent node than it sends back to the correspondent node. Streaming applications are a dominant example where content is transferred one-way only, and the reverse path carries only much sparser control traffic and acknowledgments. The large data stream typically goes from the correspondent node to the mobile node. This leads to excessive credit consumption when the mobile node's IP address is unverified. Inbound mode may not allow the mobile node to earn this much credit since new credit is given for the comparably small flow of payload packets from the mobile node towards the correspondent node. Outbound mode better accommodates such traffic asymmetry. It works well for applications with both symmetric or asymmetric traffic patterns.

A disadvantage of Outbound mode, however, is that it itself does not provide a means for the correspondent node to determine which payload packets a mobile node has received and should be credited, and which payload packets were lost and should not

be credited. Outbound mode hence needs to be accompanied with a supplementary mechanism that provides the required feedback, enabling the correspondent node to throttle credit assignment in the event of increased packet loss. This leads to a higher complexity of Outbound mode compared to Inbound mode. (Section 4.1.4 introduces a spot-check-based feedback mechanism for the correspondent node.)

### 4.1.2 Credit Aging

The concept of Credit-Based Authorization requires a mobile node to obtain credit before using that credit during a handover, so there is necessarily some time lag between credit acquisition and credit consumption. In practice, the time required to build up an amount of credit sufficient for a handover may exceed the time of the handover itself because credit acquisition may be slower than credit consumption. This is in particular the case if the correspondent node runs Credit-Based Authorization in Inbound mode and the mobile node sends at a lower rate than the correspondent node. The lifetime of acquired credit must therefore be generous enough so that the credit can eventually be turned into value during a subsequent handover.

On the other hand, the lifetime of credit must be limited so as to prevent a malicious node with a slow Internet connection from saving up credit for a while and thus provisioning for a burst of redirected flooding packets that does not relate to its own upstream capacity. Bursts of flooding packets can have a severe impact on a victim's ability to communicate even when they are only short. They can temporarily compromise the victim's responsiveness to the requests of legitimate peers, or slow down ongoing data transfers. Periodic flooding "pulses" [74, 75] have in particular been found to cause TCP connections to stall.

The problems with bursts of flooding packets can be mitigated in that correspondent nodes gradually decrease a mobile node's acquired credit over time according to an exponential aging function. This *credit aging* prevents an attacker from slowly building up high amounts of credit over a long time, which could eventually be turned into one or multiple high-bandwidth burst of redirected flooding packets. Credit aging is orthogonal to credit assignment and consumption. Any combination of credit assignment and credit consumption algorithms can be combined with a latitude of credit aging algorithms, ranging from very short-living credit that decays rapidly to more slowly aging credit which remains available for a longer time.

### 4.1.3 IP Address States

For a correspondent node to execute Credit-Based Authorization and, when necessary, limit the effort it spends in sending packets to a mobile node, the correspondent node must keep track of whether the mobile node's registered on-link IP addresses are verified or unverified at a particular point in time. This calls for the correspondent node to associate each on-link IP address with an *IP address state*, which may be Verified or Unverified. The state of an IP address is solely maintained by the correspondent node and does not need to be communicated to the mobile node.

A typical mobility protocol provides for a single on-link IP address per binding, but in general, the mobile node may be enabled to register multiple IP addresses simultaneously. For example, Mobile IPv6 bicasting [76] enables a mobile node to receive

the same payload packets via an old and a new care-of address during handover to reduce packet loss. Mobility protocols with multi-homing functionality, such as the NOMADv6 extensions for Mobile IPv6 or the mobility extensions for the Host Identity Protocol, allow a mobile node to register multiple on-link IP addresses with a correspondent node. And multi-homing protocols in general support multiple on-link IP addresses per mobile node for the purpose of load balancing and recovery from path failures. To accommodate the heterogeneity of protocols hat might integrate Credit-Based Authorization, the technique was developed based on an IP address model that permit multiple on-link IP addresses per mobile node. The correspondent node links all on-link IP addresses of a mobile node to the same binding. It may send all payload packets for the mobile node to a single IP address, or it may demultiplex the packets across different IP addresses. Similarly, the mobile node may send payload packets to the correspondent node from a single IP address or from different IP addresses.

Techniques like the NOMADv6 extensions for Mobile IPv6 enable the mobile node to partition multiple on-link IP addresses into those at which it is willing to receive payload packets from the correspondent node and those at which it is not. The correspondent node may accordingly send all payload packets for the mobile node to a single IP address, or it may demultiplex the packets across different IP addresses. Similarly, the mobile node may send payload packets to the correspondent node from a single IP address or from different IP addresses. Since only those on-link IP addresses to which payload packets are sent require reachability verification, the IP address model chosen for the development of Credit-Based Authorization differentiates two types of on-link IP addresses: A *preferred IP address* is an on-link IP address via which payload packets can be exchanged in both directions, and an *alternative IP address* is an on-link IP address that is used only for traffic in the direction from the mobile node to the correspondent node. Preferred IP addresses require reachability verification, so they are in either Unverified or Verified state. Alternative IP addresses do not need to be verified. They are maintained in a third IP address state, Alternative state. A belated reachability verification becomes necessary when a formerly alternative IP address is re-designated as preferred.

While a single preferred IP address is expected to suffice in the majority of use cases, a mobile node may also combine a preferred IP address and an alternative IP address so as to split traffic over separate, possibly non-duplex paths. For example, the mobile node may receive payload packets via a one-way broadband satellite link, and use a slower UMTS connection for the reverse path. The IP destination address terminating the path to the mobile node is then preferred for the correspondent node, while the IP source address for packets on the reverse path is alternative. A multi-homed node may register IP addresses from different Internet service providers, but use only one of them at a time as a preferred IP address for bidirectional communications with its correspondent node and keep the other, alternative IP addresses as backups. In case of a path failure upstream, the node would re-designate one of its alternative IP addresses as preferred. Mobile IPv6 is limited to a single care-of address per home address. The care-of address is preferred as it can be used for traffic in both directions. However, the NOMADv6 extensions augment Mobile IPv6 by support for flow-based multi-addressing. Similarly, the mobility extensions for the Host Identity Protocol enable mobile nodes to register multiple preferred and alternative on-link IP addresses per binding. The correspondent node may in

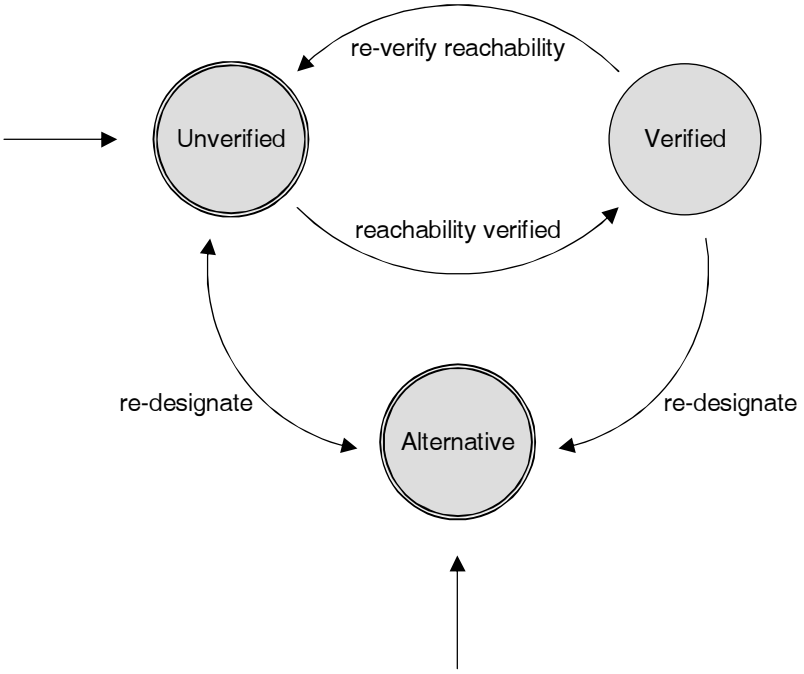



Figure 4.3: IP address states

practice end up sending payload packets to only a subset of the mobile node's preferred IP addresses. But it is henceforth assumed, without loss of generality, that exactly those IP addresses to which the correspondent node sends payload packets are preferred, while all others are alternative.

Figure 4.3 illustrates in a state machine the correspondent node's modus operandi for managing each of a mobile node's on-link IP addresses. The machine has three states—Unverified, Verified, and Alternative—of which Unverified and Alternative are possible start states. Which state an on-link IP address is initially assigned to depends on its designation at the time the mobile node registers a new on-link IP address. If the new IP address is preferred, then the start state is Unverified and concurrent reachability verification is initiated. The state of the IP address changes from Unverified to Verified when the mobile node's presence at the new IP address is confirmed. If the new IP address is registered as an alternative IP address, then the start state is Alternative and no reachability verification is needed.

A mobility protocol may decide to periodically reverify a mobile node's reachability at an on-link IP address that has already been verified previously, and it may use Credit-Based Authorization to secure the intermediate phase during which this IP address is again in Unverified state. If the correspondent node is to initiate the verification procedure, it sets the state of the IP address to Unverified when the procedure begins, and back to Verified when the verification completes. Alternatively, the responsibility for initiating the verification procedure may be with the mobile node, in which case the mobile node implements a timer indicating when a new reachability verification is due. An identical timer causes the correspondent node to set the IP address state to Unverified at roughly the same time. The correspondent node moves the state back to Verified when reachability has been verified.

A preferred IP address may be re-designated alternative, in which case the IP address moves from Unverified or Verified state to Alternative state. Any pending reachability verification then becomes obsolete. An alternative IP address may further be re-designated preferred. The IP address then moves from Alternative state to Unverified state, and the mobility protocol initiates concurrent reachability verification. Mobility protocols which do not support multiple on-link IP addresses per binding, and those which are limited to only preferred IP addresses, use a state machine with only Unverified and Verified states. The Alternative state as well as its ingress and egress transitions are then absent.

### 4.1.4 Packet Loss Estimation

A correspondent node operating in Inbound mode approximates a mobile node's effort for sending payload packets on the basis of the payload packets that it receives from the mobile node. Packet loss on the path from the mobile node to the correspondent node will therefore cause the correspondent node to *underestimate* the mobile node's effort. Modulo the typically negligible probability of packet duplication along the path, Inbound mode thus ensures that the credit assigned to a mobile node represents a lower bound on the effort that the mobile node has actually spent.

Outbound mode seeks to allot new credit to a mobile node based on the size of payload packets that the mobile node has received, but the correspondent node is limited to measuring this on the basis of the payload packets it sends. If a router along the path from the correspondent node to the mobile node drops packets due to congestion, the mobile node receives less packets than the correspondent node has originally sent. This leads the correspondent node to assign the mobile node credit for packets that were lost and that have never been delivered, independently of whether the packets were sent to an IP address in Verified state or in Unverified state. The correspondent node hence tends to *overestimate* the mobile node's effort in the presence of packet loss. This constitutes a vulnerability: An attacker could deliberately position itself behind a bottleneck router or bottleneck link, and obtain credit for packets from the correspondent node that are in fact lost at the bottleneck. The attacker could eventually convert this credit into a flooding attack against a victim. Outbound mode is hence in this study combined with an IP layer feedback mechanism, *IP address spot checks*. Through the periodic exchange of random numbers unpredictable for the mobile node, IP address spot checks enable the correspondent node to probabilistically determine how many of the packets sent to the mobile node are actually received and processed at the other end. This, in turn, permits more accurate credit assignment.

It should be noted that there is an argument that supports the suitability of Outbound mode even without IP address spot checks and as such questions the need for IP address spot checks. The argument is as follows: Packet loss is a result of congestion, and congestion at the bottleneck would alert the administrator in charge, who may be able to trace the attacker down quickly. An obvious first step for the administrator would be to contact a representative at the correspondent node domain, where the flooding packets originate, and demand the packets to be stopped. Since the attacker must authenticate to the correspondent node during registration, it should further be possible to track the attacker down from the correspondent node site. The mobility protocol may also identify the attacker or the attacker's mobility

provider in all flooding packets, in which case the administrator could take action against the attacker directly. For example, Mobile IPv6 includes the authenticated home address of a mobile node in an IPv6 Routing extension header in all redirected packets. Measures against the attacker are in general easier to put through the closer the congested bottleneck is to the attacker. Chances that the bottleneck is near the edge of the Internet and therefore close to the attacker's point of attachment are actually high, as bandwidth is typically less an issue in the core of the Internet than it is at the edge. Beyond that, a bottleneck further towards the core of the Internet would make it more difficult for the attacker to involve multiple correspondent nodes in its attack. A suitable bottleneck router or link would then have to be located such that all paths from a correspondent node to the attacker go through the bottleneck. This argument is founded on the existence of reactive countermeasures against deliberate congestion and packet loss caused by an attacker. On the other hand, any reactive strategy fails to satisfy the objective of Credit-Based Authorization to *preclude* amplified, redirection-based flooding attacks. The use of IP address spot checks is a proactive measure and thus prevents these attacks up front.

### 4.1.5 Protection Against Redirection-Based Reflection

The ability for a mobile node to simultaneously register multiple on-link IP addresses with a correspondent node may introduce a vulnerability to sustained redirection-based reflection attacks, as explained in section 3.2.2. Standard reachability verification solves this problem. But the combination of concurrent reachability verification with Credit-Based Authorization may re-introduce a vulnerability if an attacker could replenish its credit while the credit was at the same time consumed for packets illegitimately redirected towards a victim. This is specifically the case if a correspondent node assigns a mobile node (or an attacker taking the role of a mobile node) credit for its effort in communicating via one on-link IP address in Verified state, and uses the credit for sending payload packets to a different on-link IP address in Unverified state. An attacker can then register with the correspondent node both its own IP address and the IP address of a victim, send packets to the correspondent node from its own IP address, and direct the correspondent node to send packets to the victim's IP address. Due to the credit concept, the attacker would be unable to amplify the flood against the victim. But it could wage a sustained reflection attack without having to spoof its IP source address. This can be an attractive property from the attacker's perspective where IP source address filtering is performed on the network side, or where an alternative measure against the use of spoofed IP source addresses is in place.

Since the attacker is unable to pass reachability verification for the victim's IP address, this IP address will always remain in Unverified state. Any packets that the correspondent node sends to the victim hence consume the attacker's credit. However, if designed without caution, both Inbound mode and Outbound mode might enable the attacker to replenish the credit while the correspondent node uses the credit. In Inbound mode, this would happen if the correspondent node assigns new credit for payload packets received from the attacker via the attacker's own IP address, and in Outbound mode it would happen if the correspondent node assigns new credit for payload packets it sends to the attacker's own IP address. This threat

of sustained redirection-based reflection attacks can be eliminated if a mobile node is given credit only when *all* of the mobile node's preferred IP addresses are verified.[3]

Redirection-based reflection attacks are not a problem in mobility protocols which limit the number of on-link IP addresses per binding to one. A reflection attacker is then required to register the IP address of its victim, for which it is unable to pass reachability verification (unless it is located on the path from the correspondent node to the victim). The registered IP address hence remains in Unverified state, and the attacker cannot replenish its credit at the correspondent node.

### 4.1.6 Failure of Reachability Verification

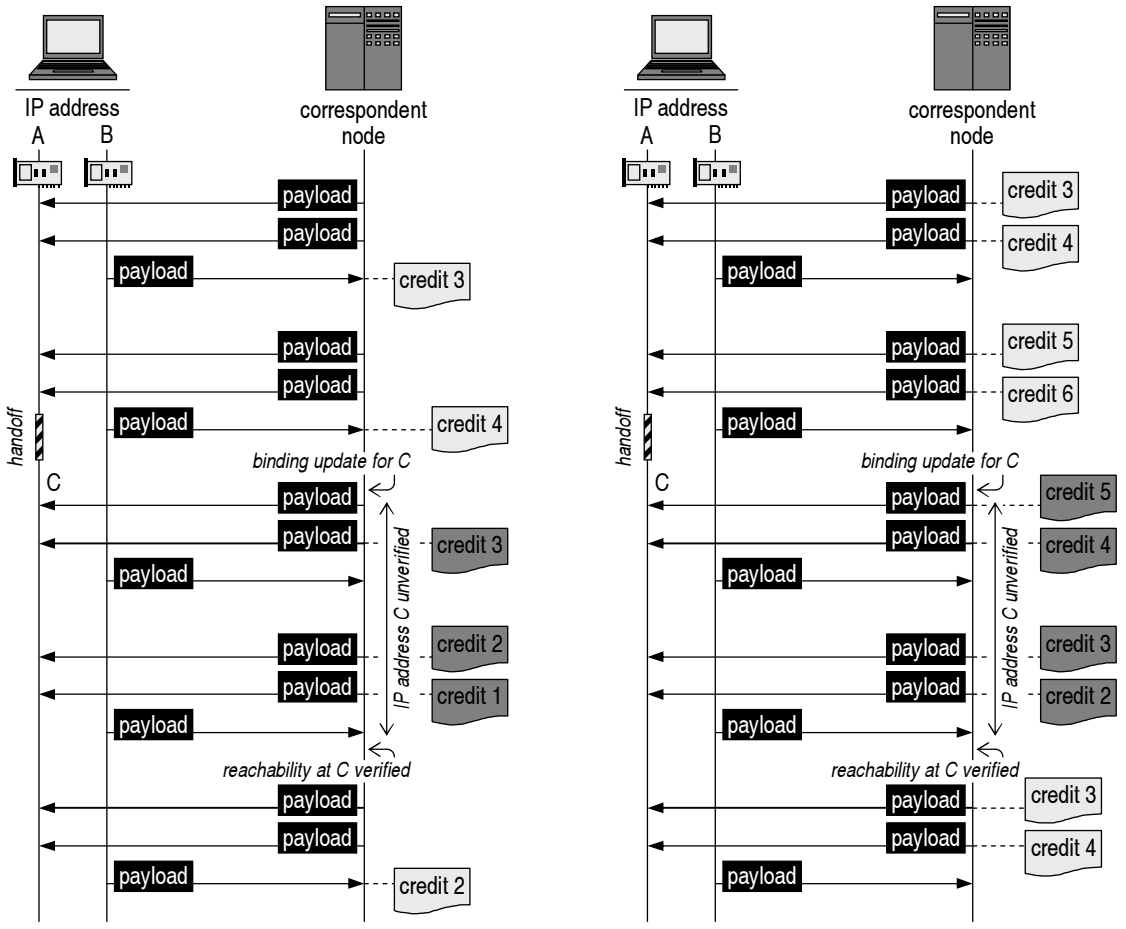
Reachability verification may fail due to lack of cooperativeness on the mobile-node side, or for reasons outside the influence of the mobile node, for example, due to packet loss on the path between the mobile node and the correspondent node. The correspondent node is in general unable to distinguish between accidental and deliberate reachability verification failures. However, this does not give an attacker an advantage because the resulting deferral of reachability verification does not increase the amount of data the correspondent node can send to the unverified IP address that the attacker has registered at the correspondent node. A flooding attack against an spoofed IP address hence cannot be extended through deferred or repeatedly unsuccessful reachability verification.

### 4.1.7 Multi-Homing Support

The generic IP address model based on which Credit-Based Authorization was designed facilitates its use in multi-homing protocols. IP address changes for the purpose of multi-homing may not be as critical as those for mobility support because multi-homed nodes are in general reachable through multiple IP addresses simultaneously. A handover from one IP address to another can therefore be prepared through proactive reachability verification. However, it may not always be desired for a multi-homed node to register all available IP addresses with a correspondent node in advance. A handover from a failing IP address to an alternative one then becomes technically identical to a mobility-related handover. The multi-homing protocol can in this case improve handover performance through the support of concurrent reachability verification and Credit-Based Authorization.

Figure 4.4 illustrates the operation of Credit-Based Authorization Inbound mode for a node that is both mobile and multi-homed. In both parts of the figure, the mobile node has initially registered two IP addresses, $A$ and $B$, with the correspondent node. The IP addresses are from different network interfaces, so the replacement of the IP address on one network interface after a handover does not imply that the IP address on the other network interface changes as well. The mobile node has requested the correspondent node to send any payload packets to IP address $A$; IP address $B$ only serves as the source of the payload packets that the mobile node sends. The mobile node acquires credit for each of its payload packets that the correspondent node receives as long as both IP addresses are verified. At some point, one of the mobile

[3]Limiting credit assignment to traffic exchanged via IP addresses in Verified state would fail to mitigate the problem because the attacker earns credit with its own IP address in the described scenario and hence could easily pass reachability verification for this IP address.




(a) Single-homed mobile node

(b) Multi-homed mobile node

Figure 4.4: Credit concept in Credit-Based Authorization

node's network interfaces hands over to a different link. In figure 4.4(a), this is the interface where the mobile node receives packets at, so IP address $A$ gets replaced with a new IP address $C$. The binding at the correspondent node then contains one verified IP address and one unverified IP address after the binding update, and the mobile node no longer gets any credit for received payload packets. In part 4.4(b) of the figure, the mobile node's outbound interface undergoes handover, and it adopts IP address $C$ in replacement for IP address $B$. The correspondent node again stops assigning new credit to the mobile node. But in this case, the credit does not decrease either because no payload packets are sent to IP address $C$. The credit hence remains indifferent until the mobile node's reachability at IP address $C$ becomes verified, at which point it begins to increase again based on the payload packets the correspondent node receives from the mobile node.

## 4.2 Credit Management

A correspondent node with support for Credit-Based Authorization associates each binding that it maintains for a mobile node with a credit counter. This counter is used for all on-link IP addresses that the mobile node has registered with the correspondent node. The following sections explain how a credit counter is initialized,

and how it increases and reduces depending on the state of the IP addresses and on the Credit-Based Authorization mode.

### 4.2.1 Initialization

A correspondent node initializes the credit counter to zero when it creates a new binding cache entry. This is most reasonable given the implications that positive or negative initialization values would have. Initialization with a positive value would certainly accommodate a legitimate mobile node whose first handover happens early after initialization. Such a mobile node may otherwise be unable to save enough credit for the entire phase of reachability verification. Yet initializing the credit counter to a positive value would also allow an attacker to redirect data worth the advance in credit without spending comparable effort, which in turn would constitute a serious security flaw. On the other hand, initialization with a negative value would bring about an additional hurdle for an attacker, but would impede concurrent reachability verification for a legitimate mobile node before the credit disadvantage has been made up for. A negative offset also does not have any strong advantage from a security perspective because it would not contribute to the objective of Credit-Based Authorization to preclude amplification in redirection-based flooding.

### 4.2.2 Inbound Mode

A correspondent node operating in Inbound mode credits a mobile node's effort based on the size of the packets that it receives from the mobile node while all of the mobile node's preferred IP addresses are in Verified state. The credit is consumed according to the size of the packets that the correspondent node sends to one of the mobile node's preferred IP addresses in Unverified state.

The activity diagram in figure 4.5 illustrates the actions that the correspondent node performs when it receives a packet from the mobile node. The correspondent node first locates the binding cache entry maintained for the mobile node and verifies that the IP source address of the received packet is currently registered as an on-link IP address. The correspondent node then determines whether all preferred IP addresses listed in the binding cache entry are in Verified state. If this is so, the correspondent node increases the credit counter associated with the binding by the size of the received packet, measured in bytes. This procedure is independent of whether the packet's IP source address is in Verified or in Unverified state.

The steps that the correspondent node takes when it has a packet to send to a mobile node are illustrated in the activity diagram in figure 4.6. The correspondent node first locates the binding cache entry maintained for the mobile node and chooses a preferred IP address to which the packet is to be sent. The selection of the IP destination address for the outgoing packet occurs according to rules set forth by the mobility protocol. It is assumed that these rules favor preferred IP addresses in Verified state over those in Unverified state so that credit is consumed only if there is no on-link IP address in Verified state to which packets for the mobile node could be sent. If the outgoing packet is a signaling packet of the mobility protocol itself, or the selected IP destination address is in Verified state, the correspondent node sends the packet without changing the mobile node's credit. Otherwise, the correspondent node proceeds depending on the current value of the credit counter relative to size

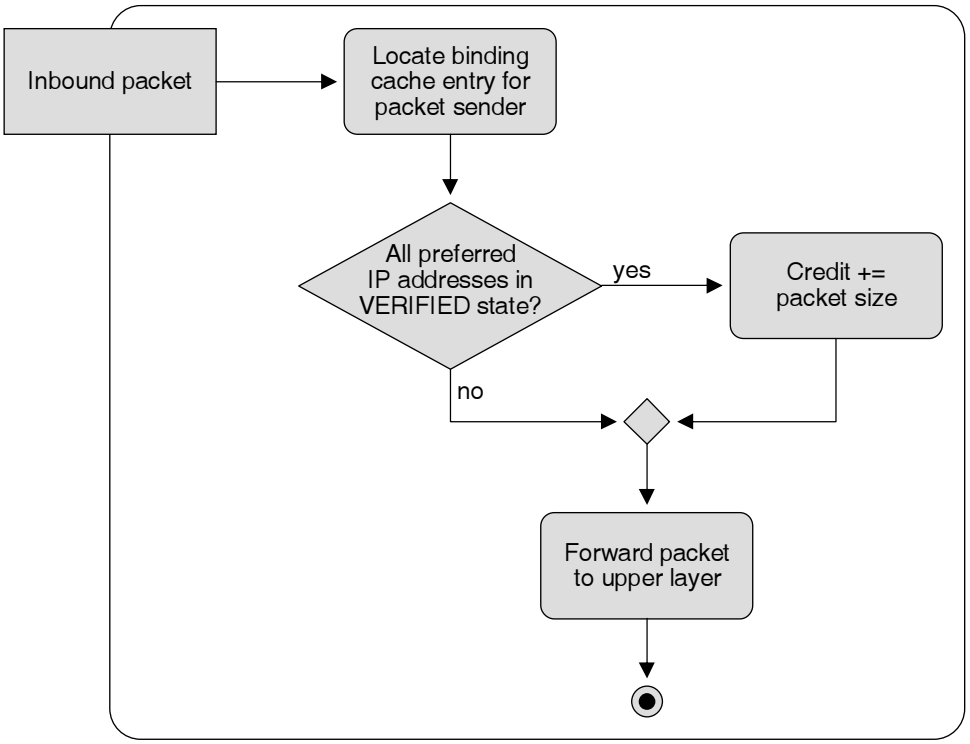



Figure 4.5: Correspondent node operation for packet reception in Inbound mode

of the packet: The packet can be sent if its size, measured in bytes, is smaller than or equal to the value of the credit counter. The correspondent node then reduces the credit counter by the packet size and sends the packet. Otherwise, the credit counter is too low for the packet to be sent to an on-link IP address. The packet may then be salvaged if the mobility protocol supports an IP address to which packets for the mobile node can be sent, but which is not an on-link IP address of the mobile node itself. This may be a stable IP address that the mobile node is permanently reachable at, like a Mobile IPv6 home address. The correspondent node may send the packet to such an IP address if one is available. If none exists, the correspondent node discards the packet or buffers it until reachability verification succeeds.

It is a matter of policy in which way the correspondent node handles outgoing packets in case insufficient credit prohibits their direct transmission to one of the mobile node's on-link IP addresses. One natural input for the correspondent node in deciding how to handle outgoing packets are the requirements of active communications applications. For example, stable IP addresses typically imply longer packet propagation delays due to a sub-optimal routing path, which can have negative impacts on applications with stringent delay and jitter requirements. The correspondent node may hence conclude that sending the packets to a stable IP address is not an option given the application through which it communicates with the mobile node. The correspondent node may instead choose to discard or buffer the packets. Packet buffering, in turn, makes the correspondent node susceptible to memory-overflow attacks and should hence be avoided in general, as explained in section 3.3.3.

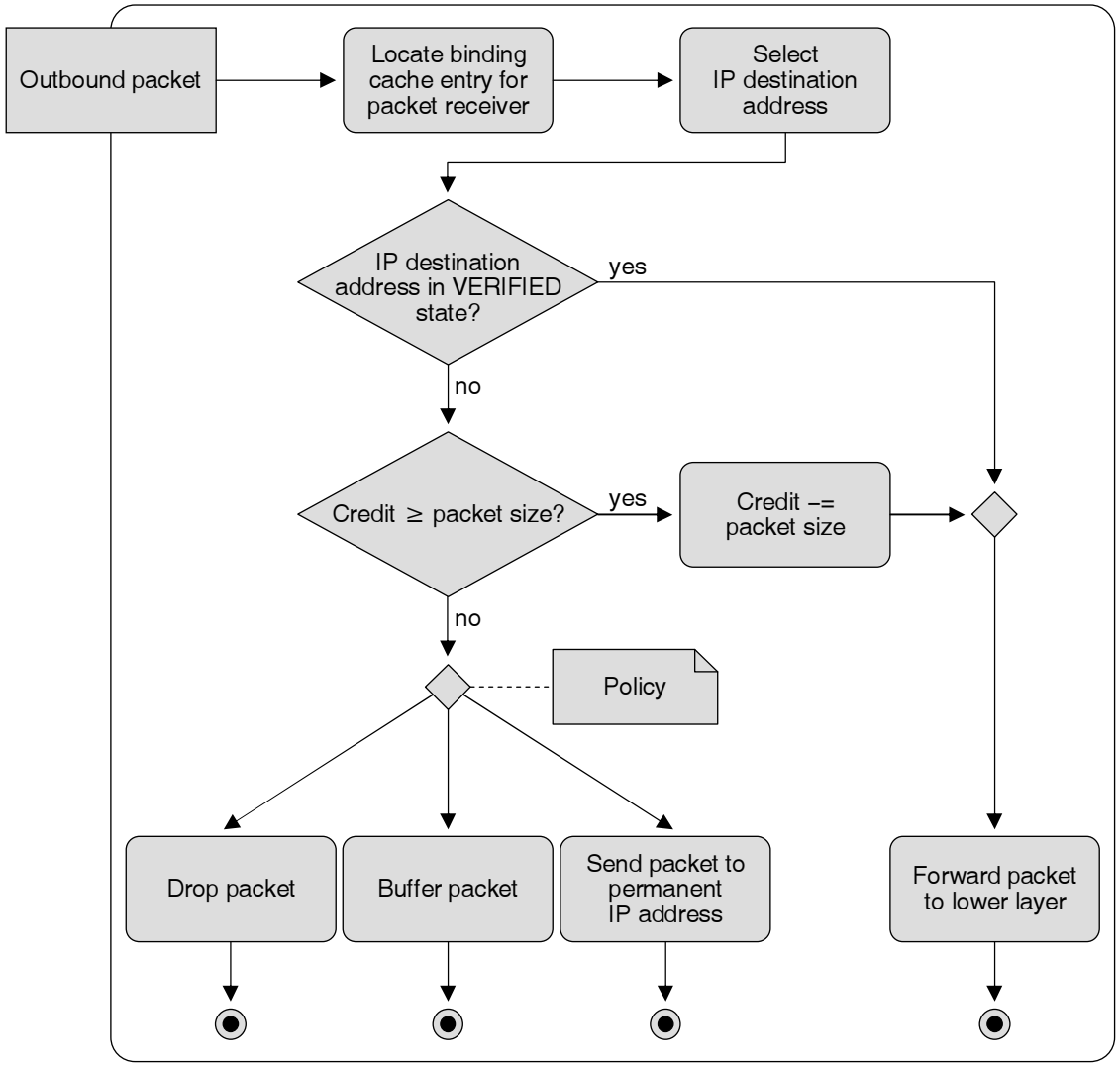



Figure 4.6: Correspondent node operation for packet transmission in Inbound mode

### 4.2.3 Outbound Mode

A correspondent node operating in Outbound mode credits a mobile node's effort based on the size of the packets that the correspondent node sends to the mobile node, and that the mobile node successfully receives and processes, while all of the mobile node's preferred IP addresses are in Verified state. The credit is consumed according to the size of the packets that the correspondent node sends to any of the mobile node's preferred IP addresses in Unverified state. The credit counter does not change when the correspondent node receives packets from the mobile node.

Figure 4.7 shows an activity diagram specifying the procedure that the correspondent node goes through when it sends a packet to the mobile node. This resembles the procedure for outgoing packets in Inbound mode, but includes the assignment of new credit when all preferred IP addresses are in Verified state. The correspondent node first locates the binding cache entry maintained for the mobile node and selects an IP destination address based on rules set forth by the mobility protocol. As in Inbound mode, preferred IP addresses in Verified state should be favored over those in Unverified state. Then, if all preferred IP addresses are in Verified state, the correspondent node increments the credit counter associated with the binding by the size of the outgoing packet, measured in bytes, and sends the packet.

The correspondent node does not increment the credit, but still sends the packet, if one or more preferred IP addresses are in Unverified state, yet the one IP destination address selected for the outgoing packet is in Verified state. In the event that the selected IP destination address is currently in Unverified state, the correspondent node behaves as follows: The outgoing packet can then be sent if and only if its size, measured in bytes, is smaller than or equal to the current value of the credit counter. Provided this is the case, the correspondent node reduces the credit by the packet size and sends the packet. If the credit counter is too low for the packet to be sent to an on-link IP address, the correspondent node may be able to send the packet to a stable IP address through which the mobile node is permanently reachable, or it may drop the packet or buffer it until reachability verification succeeds. This demands the same performance and security considerations as those described in section 4.2.2.

Figure 4.8 summarizes the activities due when the correspondent node receives a packet from the mobile node. No actions specific to credit management are required in this case. The inbound packet is normally processed by the mobility protocol and passed to the upper layer subsequently.

According to the findings of section 4.1.4, it is recommended to combine Outbound mode with a mechanism that enables the correspondent node to account for packet loss on the path to the mobile node. Packet loss on the path from the correspondent node to the mobile node may otherwise cause the correspondent node to assign the mobile node credit for data that was lost and that has never been delivered, leading to inappropriate credit assignment in favor of the mobile node. A mechanism that provides this functionality is introduced in section 4.3. It detects packet loss on the path from the correspondent node to the mobile node in a probabilistic manner and provides the correspondent node with reliable feedback of how much effort the mobile node has spent on packet reception.

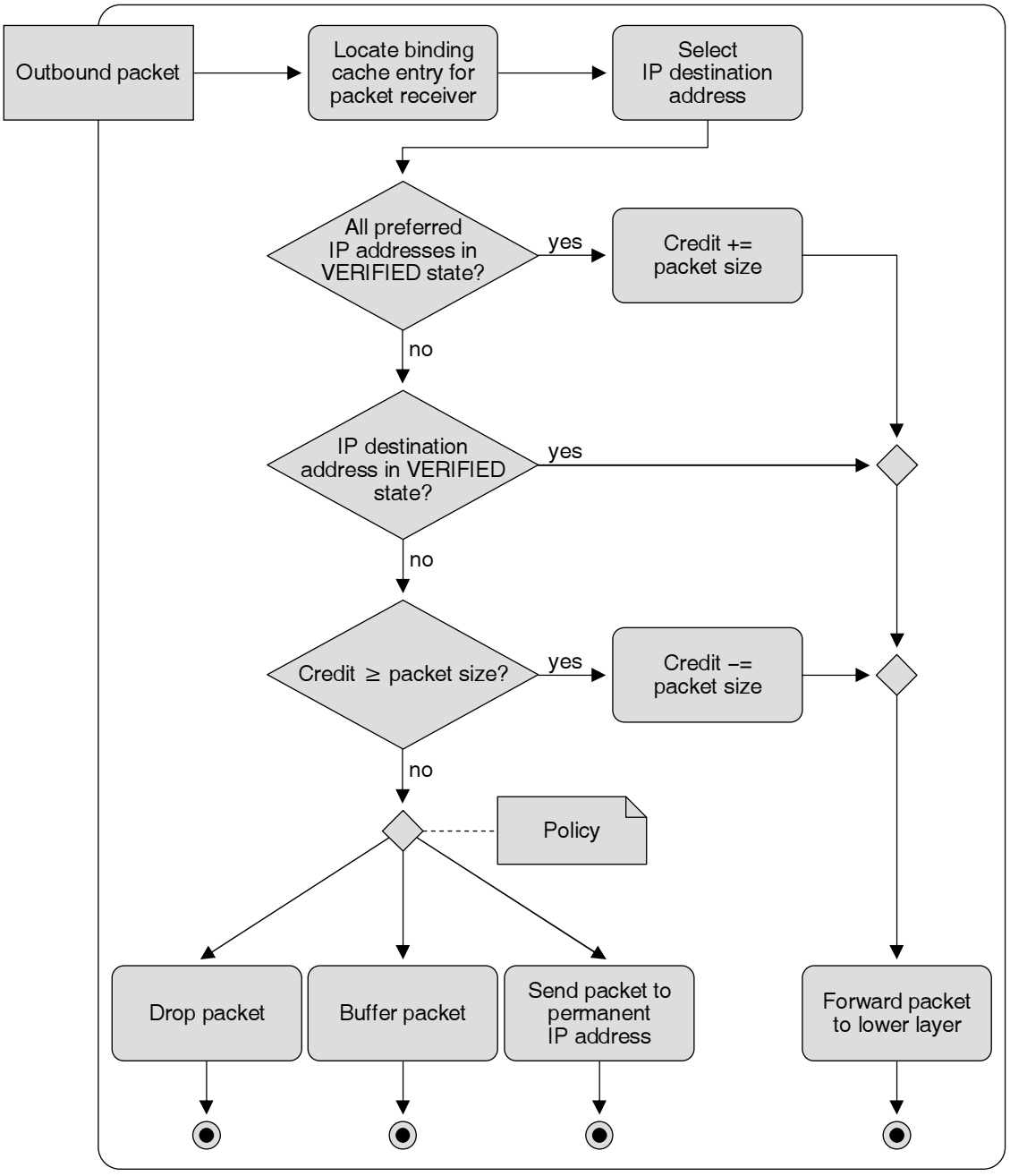



Figure 4.7: Correspondent node operation for packet transmission in Outbound mode

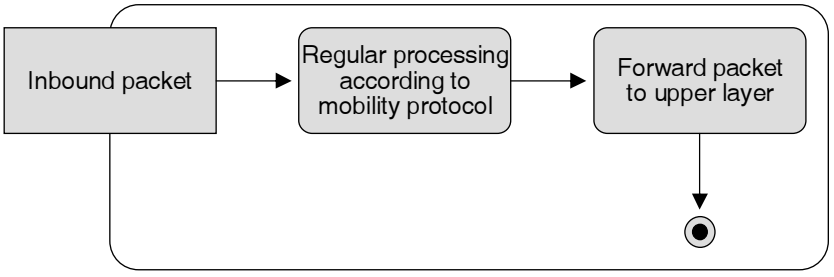



Figure 4.8: Correspondent node operation for packet reception in Outbound mode

### 4.2.4 Credit Aging

If credit had an infinite lifetime, a malicious node with poor up-link capacity could build up credit at a very slow speed and over a long period, and spend this credit during a much shorter period on redirecting a burst of payload packets to the IP address of a victim. To avoid this possibility to create bursts of redirected packets, a correspondent node ensures that all credit counters that it maintains gradually decrease over time. Each credit counter is multiplied with a *credit aging factor,* less than one, once per *credit aging interval.* Credit aging limits the total credit that a mobile node can earn relative to the rate at which the mobile node can replenish its credit. It thereby enforces an upper bound on the data volume that the correspondent node can send at once to an IP address in Unverified state.

Choosing appropriate values for the credit aging factor and interval is important for three reasons:

1. From a security perspective, credit aging should be frequent and rigid, so that acquired credit can be kept only for a short period.

2. From a performance perspective, credit aging should be lax enough to enable a mobile node to acquire the amount of credit it needs during a handover. Credit that is lost due to aging reduces the mobile node's ability to receive payload packets at an unverified IP address, which may lead to reduced application quality during handover.

3. Credit aging should be computationally efficient. The constant values for the credit aging factor and interval should preferably be powers of two, or complements of a power of two, because integer or floating-point arithmetic can thus be replaced by more efficient bit shifting operations.

Obviously, security and performance are conflicting targets and a trade-off between them needs to be effected. The right trade-off, in turn, depends on the Credit-Based Authorization mode. In Outbound mode, new credit can be acquired as quickly as it is consumed, so building up an amount of credit that is large enough to permit continued communications throughout the entire phase of concurrent reachability verification takes the same time as the phase of concurrent reachability verification takes itself. On the other hand, credit acquisition may be slower than credit consumption if the correspondent node operates in Inbound mode. Credit aging must then be tailored in special care of asymmetric application traffic patterns. This calls for laxer credit aging in Inbound mode.

In the following, credit aging factor and interval values will be defined for Inbound mode and Outbound mode. These definitions are based on three assumptions:

1. The minimum time between subsequent handovers is 60 s. (More frequent handovers can typically better be handled at the link layer directly, in which case IP connectivity does not change.)

2. The maximum time that the mobile node's IP address is in Unverified state during one handover is 1 s. (The length of the phase of concurrent reachability

verification is given by the round-trip time between the mobile node and the correspondent node. Global round-trip times in today's Internet are typically much less than 1 s [34].)

3. The traffic patterns of an application run between the mobile node and the correspondent node is no more asymmetric than a TCP-based file transfer from the correspondent node to the mobile node. The ratio between the correspondent node's transmission rate and that of the mobile node is here calculated to be 35.7. This is based on the common upper packet size limit of 1500 byte [35] for TCP segments, and a TCP acknowledgment size of 84 byte, comprising an IPv6 header, an IPv6 Destination Options extension header with a Mobile IPv6 Home Address option, and a TCP header. The usual policy of a TCP receiver to acknowledge every other segment then leads to an asymmetry of factor $\frac{84}{2 \cdot 1500} = 35.7$.

The actual asymmetry of TCP may be different if a mobility protocol other than Mobile IPv6 is used.

The following function $C(t)$ expresses the amount of credit that a mobile node has acquired during credit aging intervals that precede $t$, if $F$ is the credit aging factor, $I$ is the credit aging interval, and $R$ is the rate at which the mobile node obtains new credit. That is, $R$ is the mobile node's sending rate if the correspondent node operates in Inbound mode, and it is the mobile node's reception rate if the correspondent node operates in Outbound mode. The auxiliary variable $n$ is the number of full credit aging intervals that fit into the time period $t$. Credit that was acquired during the current aging interval, and that was not yet aged at time $t$, is not reflected in $C(t)$. It would lead to a zigzag curve inappropriate to indicate the amount of credit available throughout a handover in case that handover spanned more than one credit aging interval.

$$\begin{aligned} C(t) &= IR \sum_{i=1}^{n} F^i \\ &= IF \frac{1 - F^n}{1 - F} R \\ n(t) &:= \left\lfloor \frac{t}{I} \right\rfloor \end{aligned}$$

For Inbound mode, the mobile node acquires credit at a slower rate than it consumes the credit. Based on the assumed maximum asymmetry of a TCP-based file transfer, the ratio between credit consumption rate and credit acquisition rate would be 35.7 $s \cdot R$ at most. A feasible pair of values for the credit aging factor and interval then would satisfy the approximation $C(60\ s) \approx 35.7\ s \cdot R$. The approximation used here is due to the additional objective to choose values that facilitate efficient computation. The actual value of $C(60\ s)$ is hence expected to be slightly different than $35.7\ s \cdot R$.

In the case of Outbound mode, the mobile node acquires credit at the same rate as it consumes the credit during a handover. The values for the credit aging factor and interval could therefore in principal be chosen such that $C(60\ s) = 1\,\mathrm{s} \cdot R$. However,

thus rigid credit aging was found to lead to credit shortages when the application run between the mobile node and the correspondent node is TCP-based. The reductions in the available credit during aging then cannot be repaired at a reasonable speed when TCP throughput, hence credit acquisition, decreases temporarily due to packet loss. For this reason, the credit aging factor and interval values chosen for Outbound mode should be such that $C(60\ s)$ is slightly higher.

The following table lists the values for the credit aging factor and interval that are used in this thesis. These values satisfy the aforementioned requirements. For Inbound mode, the mobile node's available credit after 1 minute is $C(60\ s) = 37.0\,R$, which is slightly more than the $35.7\,R$ that the mobile node would requires per handover. For Outbound mode, $C(60\ s) = 4.0\,R$, a value that was found to provide the amount of credit a mobile node needs during a handover, even if credit acquisition temporarily decreases in the event a TCP-based application experiences packet loss.

| | Inbound mode | Outbound mode |
|---|---|---|
| Credit aging factor ($F$) | 7/8 | 1/2 |
| Credit aging interval ($I$) | 16 s | 4 s |

## 4.3 IP Address Spot Checks

IP address spot checks are a mechanism for correspondent nodes to probabilistically estimate the packet delivery ratio on the path to a particular mobile node. Spot checks supplement the Outbound mode of Credit-Based Authorization. They provide the correspondent node with the information necessary to reliably assign a mobile node credit for its effort in receiving packets from the correspondent node. The model underlying IP address spot check is described next; a functional description follows in the sections thereafter.

### 4.3.1 Protocol Model

IP address spot checks are a natural generalization of reachability tests. To successfully pass a spot check, a mobile node must receive an unknown, usually random or pseudo-random *spot check token* from a correspondent node and send it back to the correspondent node. The unpredictiveness of the tokens makes spot checks robust to spoofing. IP address spot checks are initiated by the correspondent node. They are executed periodically, once per *spot check interval*. At any one time, the mobile node's effort for receiving a packet from the correspondent node is attributed to the last spot check initiated before the packet was sent. Whether or not the mobile node receives credit for this effort hinges upon its ability to return the respective spot check token back to the correspondent node within a predefined *spot check token lifetime*. Each spot check token is consequently worth the effort expended during one spot check interval: If the mobile node returns the token in time, it will receive credit for the effort spent during the respective spot check interval. If it fails to do so, no credit will be given. Provided that the spot check interval is reasonably small, IP address spot checks thus facilitate a probabilistic credit assignment proportional to the amount of data the mobile node has actually received.

The correspondent node temporarily keeps track of the effort that the mobile node has supposedly expended during a particular spot check interval in the form of a

*deposit*. Conceptually, a separate deposit is opened for each new spot check, and it is turned into credit when the respective spot check token returns back at the correspondent node in time. Packets sent to the mobile node before the token from the most recently initiated spot check has returned are accounted for by incrementing the deposit that was opened for that spot check. A deposit increases by the size of each packet sent to the mobile node until either the spot check token from the respective spot check is received back from the mobile node, in which case the deposit is turned into credit, or until the next spot check is initiated and a new deposit is opened. Packets sent to the mobile node after the token from the most recently initiated spot check has returned are accounted for by incrementing the mobile node's credit directly. In general, the correspondent node maintains multiple deposits at the same time, but only the one opened most recently may be incremented. A deposit is lost if the mobile node fails to return the spot check token within a predefined period. This period is longer than the default rate at which spot checks are initiated, so the correspondent node generally keeps multiple deposits for a particular mobile node at any given time. Only the most recently opened deposit can increase, however. Older deposits remain constant either until the respective spot check token returns and the deposit can be added to the credit counter, or until the spot check token expires and the deposit is forfeited before the token is returned.

The correspondent node caches a copy of each token that it sends. This allows the correspondent node to validate a token when it later returns from the mobile node. The mobile node buffers a received spot check token until a local application delivers the next outgoing packet for the correspondent node. When this happens, the mobile node inserts the token, and possibly other tokens to be returned to the correspondent node, into the packet and sends it. The ability to return multiple tokens in one packet absorbs a potentially lower rate of outgoing packets at the mobile node compared to the rate of outgoing packets at the correspondent node. As the correspondent node receives from the mobile node packets with included spot check tokens, it checks the tokens against the copies stored locally. Each valid token then causes the deposit corresponding to it to be turned into credit.

Figure 4.9 shows a time line of spot-check-related events that occur on the correspondent node side. The captured time window includes the initiation of five spot checks, numbered #1 through #4. The respective spot check intervals, tokens, and deposits are numbered accordingly for the benefit of easier reference. The correspondent node initializes deposit #1 at the time it sends the first spot check token. This also marks the beginning of spot check interval #1. Deposit #1 increases by the size of each packet that the correspondent node sends to the mobile during this interval. For one reason or other, the mobile node does not manage to deliver spot check token #1 back to the correspondent node before the first spot check interval ends. This may be due to long propagation delays on the path to the mobile node, on the path back to the correspondent, or on both paths; or it may be due to caching delays inside the mobile node. Fortunately, the lifetime of token #1 is longer than the spot check interval, so when spot check token #2 is sent out, deposit #1 will still be kept for a while. The dispatch of spot check token #2 marks the onset of spot check interval #2, for which the correspondent node now opens a new deposit. Each outgoing packet's size is now added to deposit #2. A bit later, the correspondent node receives spot check token #1 back from the mobile node, whereupon it adds deposit #1 to the mobile node's credit counter. Deposit #2 continues to increase

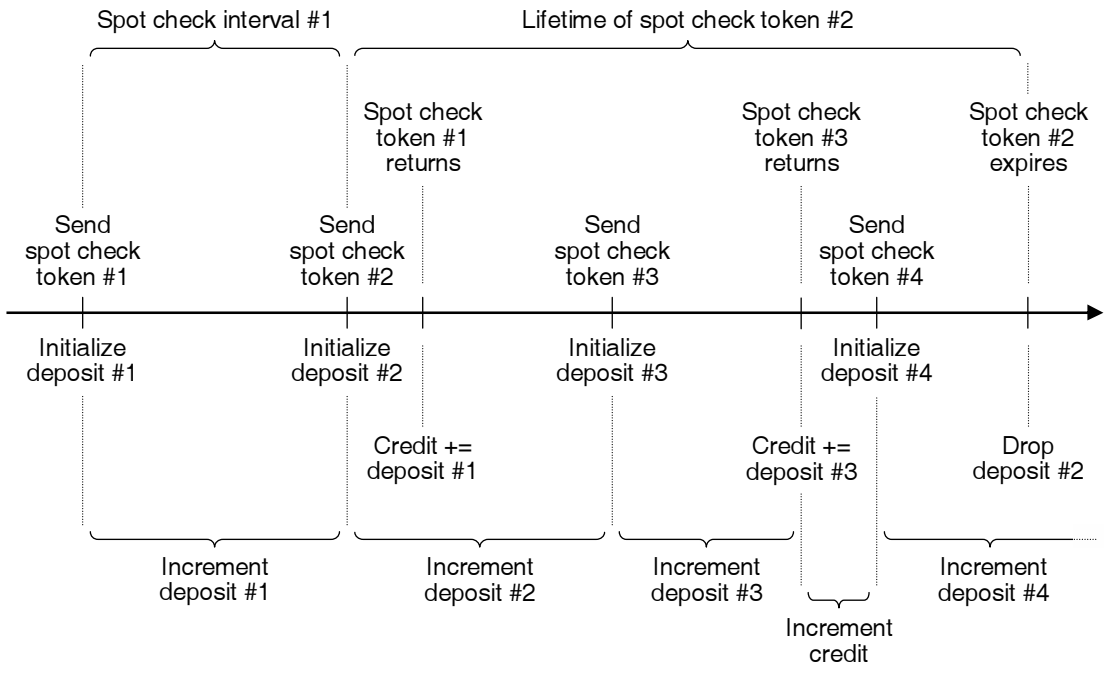



Figure 4.9: Time line of spot-check-related events on the correspondent node side

until the correspondent node sends out spot check token #3. This is when the third spot check interval begins. From then on, new deposit #3 will be incremented, while deposit #2 remains as is. Spot check token #3 returns to the correspondent node already during the same spot check interval in which it was sent, even though token #2 is still missing at this point. The correspondent node may take this as evidence that token #2 was lost—a projection which would later turn out to be true in this case. The correspondent node now turns deposit #3 into credit. Since the third spot check interval is still unfinished, but spot check token #3 did already return, the correspondent node now begins to increase the mobile node's credit, rather than a deposit, for each packet it sends to the mobile node. This continues until spot check interval #4 begins, at which time a new deposit will be created. Spot check token #2 never comes back to the correspondent node, so the correspondent node deletes deposit #2 when the lifetime of token #2 is eventually over.

Spot checks are executed separately for each preferred IP address of the mobile node, although they all eventually contribute to the increasing the single credit counter in the mobile node's binding. The spot checks pause while an IP address is in Unverified state because no credit is then assigned to the mobile node anyway. Given that IP address spot checks are executed in parallel with regular communications, they can be piggybacked onto other signaling or data packets that mobile and correspondent nodes exchange as part of their regular communications. This reduces the bandwidth overhead required for spot checks, which is a paramount objective given the generally much higher frequency of spot checks compared to reachability verification. In-band spot checks are further guaranteed to take the same routing path as regular data packets, whereas separate signaling packets used for out-of-band spot checks might be routed differently, in which case the spot checks would fail to probe packet loss rates on the actual data path.

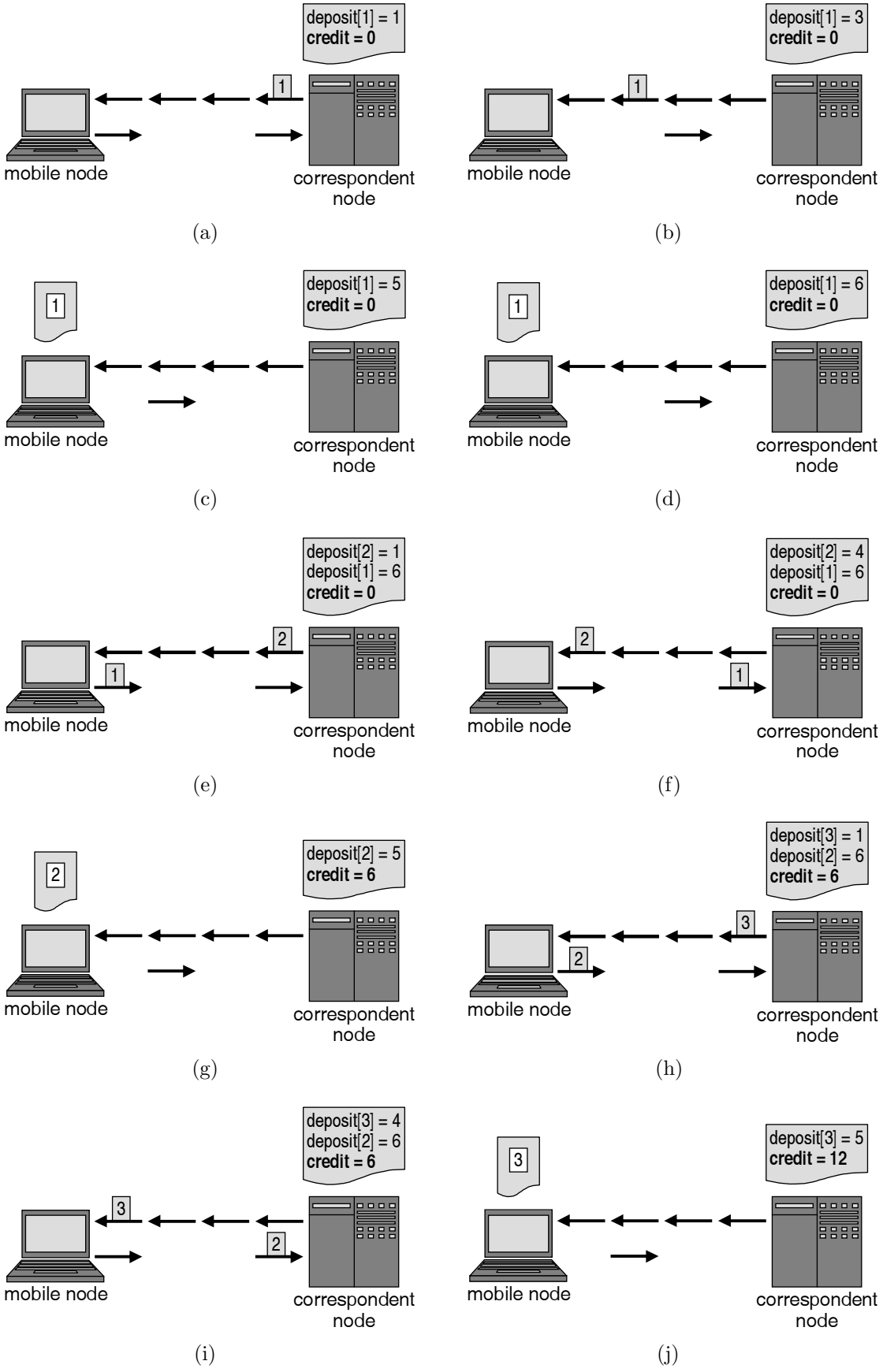



Figure 4.10: Spot check token exchange between mobile and correspondent node

Figure 4.10 illustrates the exchange of spot check tokens between a mobile node and a correspondent node. The correspondent node sends packets to the mobile node at a constant rate, and the mobile node returns an acknowledgment for every third packet received from the correspondent node. The correspondent node initiates the first spot check in figure 4.10(a), where it attaches token #1 to one of the packets it sends. A deposit is opened at this time to account for the mobile node's potential effort in receiving the packets from and including this spot check up to and excluding the forthcoming spot check. The deposit is simplified in the example in that it shows the number of packets sent rather than the sum of the packet sizes. It hence accounts for only the packet that includes token #1 at the time spot check is initiated. The deposit increases as the correspondent node sends more packets to the mobile node, as shown in the snap shot of figure 4.10(b). In figure 4.10(c), spot check token #1 reaches the mobile node, which caches the token until it sends the next acknowledgment to the correspondent node. The correspondent node has now sent 4 packets following the one that included token #1, so the deposit associated with this token equals 5 by this time and grows to 6 in figure 4.10(d). It takes until figure 4.10(e) for the mobile node to send an acknowledgment to which it attaches the cached token. The correspondent node initiates the next spot check at the same time, inserting spot check token #2 into its next packet. This causes the correspondent node to open a second deposit. From now on, deposit #2 will be increased for each packet the correspondent node sends, and deposit #1 will remain constant at 6. Token #2 approaches the mobile node in figure 4.10(f) and is received and cached by the mobile node in figure 4.10(g). Simultaneously, token #1 returns to the correspondent node. The correspondent node compares the received token against the token it originally sent to the mobile node and, since the tokens match, adds deposit #1 to the mobile node's credit. After sending two more packets, the correspondent node generates the next spot check, as shown in figure 4.10(h). The mobile node returns token #2 along with the next acknowledgment at the same time. The packets make their way in figure 4.10(i) and arrive at their respective recipient in figure 4.10(j). The correspondent node finally adds deposit #2 to the mobile node's credit after having verified that token #2 is correct.

### 4.3.2 Generating Spot Check Tokens

It is important that mobile nodes cannot derive a spot check token other than from a received Spot Check option that contains the token. A correspondent node must ensure this by complying to the following rules in generating spot check tokens:

1. Spot check tokens must be unguessable to mobile nodes, preferably random or pseudo-random.

2. Spot check tokens sent to a particular IP address of a mobile node must be independent of other tokens sent to the same IP address.

3. Spot check tokens sent to one IP address of a mobile node must be independent of tokens sent to a different IP address of the same mobile node.

4. Separate token name spaces must be used for different mobile nodes.

Failure to comply with any of these rules would enable an attacker to pass multiple spot checks without receiving tokens for all of them. This is obvious if the attacker could derive a missing spot check token based on tokens received earlier or shortly afterwards at the same or at different IP addresses. Similarly, the attacker and multiple other mobile nodes simultaneously communicating with the same correspondent node may be located on a common routing path, for example, when they attach to the same broadcast access link. If the spot check tokens that the correspondent node uses for different mobile nodes would interdepend, the attacker could eavesdrop on neighboring mobile node's spot check tokens and infer the tokens that it was supposed to receive itself from the correspondent node. In both cases, the attacker could cause the correspondent node to believe that it received packets which in fact it did not. The correspondent node would then assign the attacker more credit than appropriate.

A simple correspondent node implementation could satisfy the above rules by generating a new pseudo-random token whenever a spot check is due. However, the computation of numbers with good pseudo-random properties is non-trivial [2]. The associated requirement for processing resources may be delicate not only for correspondent nodes with sparse processing capacities, such as handheld devices with correspondent node functionality, but also for well-provisioned servers which cannot afford to dedicate part of their resources on tasks other than the transmission of data packets. It is hence advisable to optimize the generation of spot check tokens as much as possible.

A more sophisticated token generation technique is the one Mobile IPv6 uses for home and care-of address tests. This allows a correspondent node to recycle pseudo-random nonces for the generation of multiple home or care-of address tokens. The approach in addition has the appealing property that it does not require the correspondent node to explicitly store the tokens sent to mobile nodes. Given an index to the appropriate nonce, a token can simply be recalculated on demand. This is an advantage in Mobile IPv6, where the correspondent node would otherwise have to allocate new memory for unauthenticated[4] mobile nodes. A high number of Home or Care-of Test Init messages, sent from different IP addresses by one or multiple attackers, could then exhaust the correspondent node's memory capacities and render it incapable to continue regular communications. On the other hand, the ability to recalculate a token is rather subordinate in the context of IP address spot checks because spot checks are executed only after a mobile node has authenticated. Also, the amount of memory the correspondent node must devote to spot checks for a particular mobile node only depends on the number of preferred IP addresses that the mobile node has registered. The memory required for a single preferred IP address is constant as it only depends on the spot check interval and the spot check token lifetime. The correspondent node may hence bound the memory it may need for a particular mobile node by posing a limit on the number of preferred IP addresses per binding. This facilitates explicit storage of spot check tokens at the correspondent node.

Extending binding cache entries to each include the spot check tokens recently sent to the respective mobile node allows the correspondent node to efficiently verify

[4] An exception are binding refreshes, which require a return routability procedure for mobile nodes that have already been authenticated.

returned tokens. Incorrect, possibly bogus tokens can thus be discarded without much effort. The approach has another advantage: While recalculating tokens on demand can protect a correspondent node against denial-of-service attacks against its memory resources, at the same time it makes the correspondent node vulnerable to attacks against its processing capacities. An attacker could keep a correspondent node busy by sending it a high number of bogus packets, each requiring the correspondent node to recalculate a token before the packet can be identified as fake. The attacker could so impede the correspondent node in serving legitimate peers. Such an effect can be attained in Mobile IPv6 through high volumes of incorrectly authenticated Binding Update messages. During the security design of Mobile IPv6, this vulnerability was traded against the memory exhaustion attack described above, and it was eventually found to be the lesser evil. Spot checks do not introduce a vulnerability to memory exhaustion attacks, however, so it is wise to store spot check tokens explicitly and thus avoid waste of processing resources in recalculating them. Furthermore, reachability verification occurs much less frequently for an IP address than spot checks do. Computational efficiency for spot checks is hence also in the absence of an attack more important than storage efficiency.

Nonetheless, the Mobile IPv6 concept of reusing pseudo-random nonces in the generation of home or care-of keygen tokens for multiple mobile nodes makes sense also in the case of IP address spot checks as it reduces the processing resources required for pseudo-random number computations. Transferred to spot checks, the concept requires a correspondent node to generate a fresh pseudo-random *spot check nonce* of 64 bit length every spot check interval. When it is time for the next spot check, the correspondent node calculates a spot check token based on the current spot check nonce and the on-link IP address to which the spot check is sent:

$$\text{spot check token} \;=\; \mathit{first}_{\,64}\,(\mathit{SHA\text{-}1}\ (\text{on-link IP address} \,|\, \text{spot check nonce}))$$

Here, $\mathit{first}_{\,64}$ returns the first 64 bits of the *SHA-1* hash, and the pipe symbol ("|") denotes the concatenation of the strings to its left and right. For the correspondent node to quickly retrieve its local copy of the respective spot check token when a token is received back from the mobile node, each token is tagged by a running *spot check token index*. The token index is sent along with the token itself, both to the mobile node and from there back to the correspondent node. The absence of the token index would require the correspondent node to compare a received spot check token against its local copies of multiple tokens recently sent to the mobile node. An attacker could take advantage of this by sending the correspondent node high volumes of packets with incorrect tokens, each of which would require the correspondent node to perform a number of comparisons, and finally conclude that the received token is fake.

Generating a new nonce every spot check interval further ensures that the same spot check nonce is not used more than once for the same IP address. The mobile node may otherwise receive two or more equal spot check tokens in a row. As a spot check token further depends on the IP address to which the token is sent, a mobile node with multiple IP addresses receives independent tokens at each of those. Tokens received at one IP address hence cannot compensate for tokens missed at a different

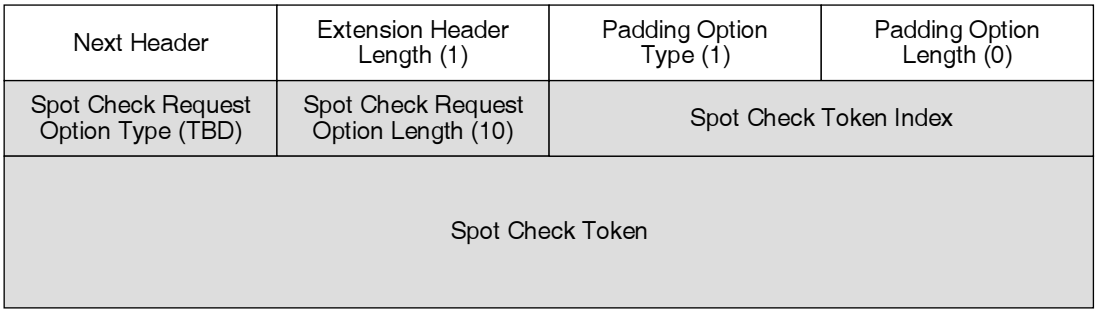



Figure 4.11: Spot Check Request option within an IPv6 Destination Options extension header

IP address. Spot check tokens sent to different mobile nodes are taken from separate name spaces for the same reason. Tokens of neighboring mobile nodes are therefore unusable for an eavesdropper.

Given that the mobile node may cache a received spot check token for a while until it returns the token to the correspondent node, the IP address from which the token is returned may be different than the IP address for which the token was generated. This may happen if the mobile node receives the token shortly before a handover and caches it until after the handover. For example, delayed acknowledgments in TCP may hold the last acknowledgment for an uneven number of segments, received before a handover, until after the handover. If the last segment received before the handover contains a spot check token, then the mobile node is likely to return the token after the handover. However, the correspondent node can still validate the token independent of the IP address from which the token was returned because it does not need to recalculate the token.

### 4.3.3 In-Band Transport

The Destination Options extension header from the IPv6 protocol specification provides a suitable transport for spot checks. Two new options are defined for this purpose. The correspondent node sends a spot check token and the associated token index to the mobile node in a *Spot Check Request option*. This is depicted in figure 4.11 together with the containing extension header and a leading Padding option. Padding is needed to align the Spot Check Token field on its natural 64-bit boundary [54]. The Spot Check Request option includes the token index in a 16-bit *Spot Check Token Index field* that follows the mandatory Option Type and Option Length fields, as well as the token generated by the correspondent node in a 64-bit *Spot Check Token field*. The mobile node returns one or multiple cached tokens along with the respective indices in a *Spot Check Reply option*. This option, illustrated in figure 4.12, has a similar structure as the Spot Check Request option, but it may have more than one pair of Spot Check Token Index and Spot Check Token fields, depending on how many tokens are waiting for return at the mobile node. The Spot Check Token Index fields are grouped together, and so are the Spot Check Token fields. All fields can thus be aligned on their natural 16-bit or 64-bit boundaries, respectively, without requiring padding inside the option. A Padding option may still have to precede the Spot Check Reply option, depending on the presence of other options in the same Destination Options extension header as well

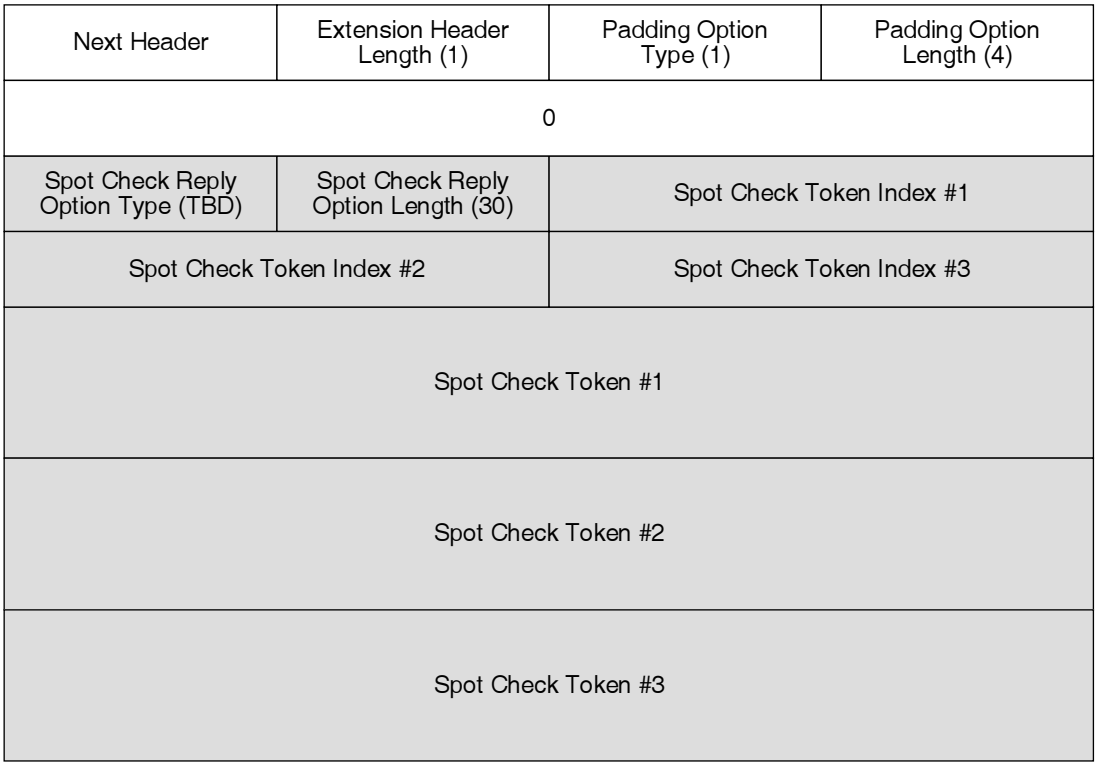



Figure 4.12: Spot Check Reply option including 3 spot check tokens and their indices within an IPv6 Destination Options extension header

as the number of tokens included in the Spot Check Reply option. For example, a Padding option of two bytes length is needed in figure 4.12. The number of tokens, $N$, included in a Spot Check Reply option can be derived by dividing the value in the Option Length field by the sum of the Spot Check Token Index field length and the Spot Check Token field length:

$$N = \frac{\text{value in Option Length field}}{10}$$

The Spot Check Request option has an alignment requirement of $8n + 4$, assuring that the Spot Check Token Index and Spot Check Token fields fall on their natural 2-byte and 8-byte boundaries, respectively, for efficient processing in 64-bit machines. This requires padding in most cases. E.g., if the Spot Check Request option is the first non-padding option in the Destination Options extension header, as in figure 4.11, a 2-byte Padding option must be inserted between the extension header fields and the option itself. The alignment requirement for a Spot Check Reply option depends on the number of tokens contained in the option. If a multiple of 4 tokens is included, the option has an alignment requirement of $8n + 6$, and it must be preceded by 4 bytes of padding in the absence of a foregoing non-padding option within the same extension header. If the number of included tokens is $4k+1$, $4k+2$, or $4k + 3$, then the alignment requirement amounts to $8n + 4$, $8n + 2$, and $8n$, respectively. Padding of length 2 bytes, zero, and 6 bytes is accordingly needed unless another non-padding option within the same extension header necessitates a

different quantity. Figure 4.12 shows a Spot Check Reply option with three included spot check tokens. An implementation can dynamically calculate the offset $m$ by which a given Spot Check Reply option should be aligned based on the number of spot check tokens, $N$, included in the option as

$$m(N) \quad = \quad 6 - 2\,(N \; mod \; 4)$$

The alignment requirement for the option is then $8n + m$. The number of padding bytes, $p$, to be prepended to the option in case there is no foregoing non-padding option then amounts to

$$p(m) \quad = \quad (m - 2) \; mod \; 8$$

An IPv6 Destination Options extension header including a Spot Check Request or Reply option must be the last extension header in the packet. This ensures that the options are only evaluated by the end nodes.

Option type values for Spot Check Request or Reply options must be registered at the *Internet Assigned Numbers Authority, IANA* [58]. Both options are non-critical, i.e., a packet including an option should not be discarded if the receiving end node does not recognize the option. The first two bits of both Option Type values must therefore be 00 so as to encode these semantics. Furthermore, Spot Check Request or Reply options do not change en-route, and their contents need not be masked in the calculation or verification of an IPsec Authentication extension header's integrity check value [66]. The third bit in the Option Type values must hence be 0.

### 4.3.4 Protocol Configuration

The preciseness of credit assignment with IP address spot checks can be traded with overhead through the interval, SpotCheckInterval, between the initiation of the previous spot check of a particular mobile node and the due time for the next spot check. Thus, spot checks for a particular mobile node occur once per SpotCheckInterval at most. A spot check may be initiated later than its due time if lack of outgoing packets defers spot check initiation, or one of the mobile node's preferred IP addresses is in Unverified state. Independent of the spot check frequency is the maximum time, MAX_SPOT_CHECK_DURATION, which the correspondent node may wait for a specific spot check token to be returned by the mobile node. The purpose of MAX_SPOT_CHECK_DURATION is to account for potential delays that it might take the mobile node to return a received spot check token. Generally, a mobile node cannot return a received spot check token immediately to the correspondent node. It rather depends on the application or transport protocol when the next packet, to which the token can be attached, is sent to the correspondent node. Depending on the asymmetry of traffic, the rate at which the mobile node sends packets to the correspondent node may also be much lower than the rate at which the correspondent node sends into the reverse direction. The correspondent node does not accept a returned token that was sent more than

MAX_SPOT_CHECK_DURATION time ago. If the mobile node fails to return a token within this time, the correspondent node may delete its locally stored copy of the token, freeing the memory allocated for it. While SpotCheckInterval is a variable that the correspondent node can choose, possibly in consideration of the services it provides to mobile nodes, MAX_SPOT_CHECK_DURATION is a constant. Fixing the value allows a mobile node to discard any stale spot check tokens which it was unable to return to the correspondent node in time. Moreover, a lower bound of MIN_SPOT_CHECK_INTERVAL is defined for SpotCheckInterval. The mobile node can use this in conjunction with MAX_SPOT_CHECK_DURATION to determine the maximum number of spot check tokens that it may have to memorize. The following values are suggested for the described protocol configuration variables and constants:

| | |
|---|---|
| SpotCheckInterval | 0.2 s |
| MIN_SPOT_CHECK_INTERVAL | 0.02 s |
| MAX_SPOT_CHECK_DURATION | 4 s |

The minimum spot check interval of MIN_SPOT_CHECK_INTERVAL implies a very aggressive spot checking behavior. The interval is short enough to add spot check tokens to all packets transmitted in a typical Internet telephony application. A longer spot check interval should be sufficient in most cases, however. The selection of an appropriate default value for the SpotCheckInterval configuration variable is an important design choice. The higher the frequency of spot checks, the fine the granularity by which the correspondent node can determine the level of congestion on the path towards the mobile node. A low value for SpotCheckInterval would therefore be advantageous. On the other hand, each spot check implies an associated processing and storage overhead both at the mobile node and at the correspondent node. The frequency of the spot checks should therefore be limited in order to contain this overhead. The proposed default value for SpotCheckInterval was chosen in consideration of this trade-off.

The value of the MAX_SPOT_CHECK_DURATION parameter must be sufficiently high so that a mobile node does not fail a spot check due to time constraints. However, the optimum limit on the spot check duration not only depends on the round-trip time between the correspondent node and the mobile node, but also on the availability of packets which can carry a spot check token both from the correspondent node to the mobile node and vice versa. Neither the round-trip time nor the availability of packets is known in advance, so MAX_SPOT_CHECK_DURATION should be set to a value high enough to accommodate most situations. At the same time, the longer MAX_SPOT_CHECK_DURATION, the more tokens the correspondent node may have to memorize for any given value of SpotCheckInterval. This calls for a low value of the parameter. MAX_SPOT_CHECK_DURATION also governs the window within which tokens sent to one IP address can be returned from a different IP address, for which the tokens are then turned into credit. The parameter should be low also from this perspective. The proposed value was selected so as to effect a suitable compromise.

### 4.3.5 Conceptual Data Structures

A correspondent node keeps the spot check tokens that it has recently sent to a particular mobile node in a *spot check token list*, which is integrated with the binding

| Spot Check Token Index | Spot Check Token | Time Stamp | Deposit | Returned |
|---|---|---|---|---|
| 0004 | 775c3d53643f5c5f | 16:23:54.593 | 1088 | 0 |
| 0005 | 5c67595b7127203a | 16:23:54.375 | 21760 | 0 |
| 0006 | 3e64353e474e2727 | 16:23:54.124 | 16320 | 0 |
| 0007 | 433c7c275925383e | 16:23:53.901 | 19584 | 1 |
| 0008 | f920584b725f6936 | 16:23:53.669 | 14144 | 1 |
| 0009 | 2f6f67139329662d | 16:23:53.447 | 17408 | 1 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 0022 | 174304f72465f763 | 16:23:50.238 | 18496 | 0 |
| 0023 | 68305c2350242143 | 16:23:49.996 | 15232 | 1 |

current time: 16:23:54.602

Figure 4.13: Spot check token list

cache entry that the correspondent node maintains for the mobile node. A new spot check token is added to the token list whenever the correspondent node initiates a spot check of the mobile node. This may happen up to every SpotCheckInterval, provided that packets are promptly available to carry the tokens. Each spot check token may have to be memorized for a maximum of MAX_SPOT_CHECK_DURATION time, so the token list must hold a number of $N$ tokens, where

$$N = \left\lceil \frac{\text{MAX\_SPOT\_CHECK\_DURATION}}{\text{SpotCheckInterval}} \right\rceil$$

Entries of the spot check token list are 5-tuples. Besides a Spot Check Token field for the token itself, an entry contains a Spot Check Token Index field with a running token index for efficient access to a particular token received back from the mobile node, a Time Stamp field with a time stamp of when the token was sent to the mobile node, a Deposit field counting the credit that the mobile node will be given provided that it successfully returns the token to the correspondent node, and a Returned flag indicating whether or not this has already happened. The due time for the next spot check of the same mobile node can be derived by adding SpotCheckInterval to the value in the Time Stamp field of the currently newest entry in the spot check token list. A spot check is typically initiated a bit later than its due time, however, since applications may not promptly deliver a packet which could carry a spot check token. Figure 4.13 shows an exemplifying spot check token list. A value of 0.2 s is assumed for SpotCheckInterval, which is the period suggested and justified in section 4.3.4. The token list holds 20 entries to accommodate the selected rate of spot checks over a period of MAX_SPOT_CHECK_DURATION time. The entries are stored in the order of increasing age, but implementations may use different list management strategies. In the example, the most recent spot check was already due at time 16:23:54.575—that is, SpotCheckInterval plus the time stamp of the entry with index

5—, but only at time 16:23:54.593 became an outgoing packet available into which a spot check token could be inserted. The spot check token with index 22 was not returned to the correspondent node within MAX_SPOT_CHECK_DURATION time, so the deposit worth 18496 bytes will not be turned into credit.

A token list entry's Deposit field is initialized to zero when the entry is created, and its Returned flag is cleared. Whenever a packet is sent to the mobile node while the Returned flag in the currently newest entry in the token list is cleared, the value in the Deposit field of that entry is incremented by the size of the outgoing packet in bytes. Once the token is received back from the mobile node, the correspondent node adds the value of the Deposit field to the credit counter in the same binding cache entry and sets the Returned flag. Packets sent to the mobile node while the Returned flag in the currently newest token list entry is set are accounted for by adding their size to the credit counter directly.

At any given point in time, the correspondent node generates spot check tokens for all mobile nodes it communicates with based on a spot check nonce, which it keeps, along with a time stamp of when the nonce was created, in a *spot check nonce cache*. Accordingly, the nonce cache has a Nonce field and a Time Stamp field, and it is global to the entire binding cache. A spot check nonce is valid for SpotCheckInterval time since it may be used only once per mobile node. An expired nonce gets renewed on demand when the next token is to be generated. Mobile IPv6 implementations with support for spot checks may colocate the caches for spot checks nonces and the nonces used for the return routability procedure. However, given that spot check nonces are generally renewed much more frequently than the nonces used for the return routability procedure, it is advisable to separate the caches.

A mobile node buffers the spot check tokens received from a correspondent node in a *spot check token buffer*, which is integrated with the binding update list entry maintained for the respective correspondent node. A cached token is removed from the cache once a local application delivers a packet with which the token can be returned to the correspondent node. Entries of the spot check token buffer comprise a Spot Check Token field for the buffered token itself, a Spot Check Token Index field with the corresponding token index, and a Time Stamp field with the time the token was received. Tokens older than MAX_SPOT_CHECK_DURATION time are purged from the token buffer. They need not be sent back to the correspondent node as they would no longer be accepted anyway. Spot check token buffer entries may grow stale if there is lack of packets with which the tokens could be returned to the correspondent node. In general, however, the period of MAX_SPOT_CHECK_DURATION should be sufficiently long to return all received tokens even in the presence of very sparse traffic in the direction from the mobile node to the correspondent node.

The maximum size of the spot check token buffer depends on the frequency at which the correspondent node initiates spot checks. This may be as high as one token every MIN_SPOT_CHECK_INTERVAL time. Since tokens are buffered for up to MAX_SPOT_CHECK_DURATION time, a conservative approach would be to dimension the spot check token buffer to hold a number of $N'$ entries, where

$$N' = \left\lceil \frac{\text{MAX\_SPOT\_CHECK\_DURATION}}{\text{MIN\_SPOT\_CHECK\_INTERVAL}} \right\rceil$$

On the other hand, the actual usage of a spot check token buffer is likely to be less than $N'$ entries. For one reason, packets that can carry a buffered token back to the respective correspondent node are generally available earlier than MAX_SPOT_CHECK_DURATION time after the token was received. The default spot check interval of SpotCheckInterval is also significantly larger than the minimum value. So a token buffer smaller than $N'$ entries may still prove sufficient. Even the unlikely case of a buffer overflow would have no serious consequences as it would simply cause the mobile node to drop some of the credit that it might otherwise earn. Mobile nodes are therefore rather flexible with respect to the amount of memory they allocate to spot check token buffers, a property that facilitates deployment of IP address spot checks on mobile nodes with limited memory resources. A sophisticated approach would be to learn the arrival rate of spot check tokens from a particular correspondent node on the fly and redimension the token buffer accordingly. A constant upper limit of $N'$ entries should be placed on the size of the token buffer nonetheless, so as to fend off memory exhaustion attacks through large quantities of bogus packets that include arbitrary spot check tokens and appear to be originating with the legitimate correspondent node.

### 4.3.6 Correspondent Node Operation

All spotcheck-related activities of a correspondent node are triggered by either the arrival of an outgoing packet from an upper-layer protocol, or the reception of a packet from the network. Figure 4.14 shows an activity diagram for the procedure anchored at the arrival of an outgoing packet. Given such a packet, the correspondent node first locates the binding cache entry for the mobile node that is to receive the packet and verifies whether a spot check is due for the mobile node. A spot check is due if and only if both the mobile node's preferred IP address is in Verified state, and the time stamp of the currently newest entry in the spot check token list is at least SpotCheckInterval time old. If any of these conditions remain unsatisfied, the packet will be sent as is. The correspondent node accounts for the mobile node's (potential) effort in receiving the packet by adding the size of the outgoing packet, measured in bytes, either directly to the credit counter in the binding cache entry maintained for the mobile node, or to the value in the Deposit field in the currently newest entry in the spot check token list. The credit counter is incremented if the spot check token most recently sent to the mobile node has already been received back. This condition is indicated by the Returned flag in the currently newest token list entry being set. If the token has not been returned so far, the value in the Deposit field is incremented. The correspondent node will transfer the deposit to the credit counter once the spot check token from the newest entry in the token list returns back from the mobile node.

If the correspondent node determines that a spot check is due, it takes the following steps. It first checks to see whether the current spot check nonce needs to be renewed. If the difference between the value of Time Stamp field in the nonce cache and the current time is smaller than SpotCheckInterval time, no action needs to be taken. Yet a difference equal to or greater than SpotCheckInterval indicates that the correspondent node must replace the existing nonce in the nonce cache by a new random 64-bit value. Second, the correspondent node generates a fresh token for the mobile node according to the formula described in section 4.3.2. It then creates a new entry in the spot check token list, thereby possibly replacing the currently

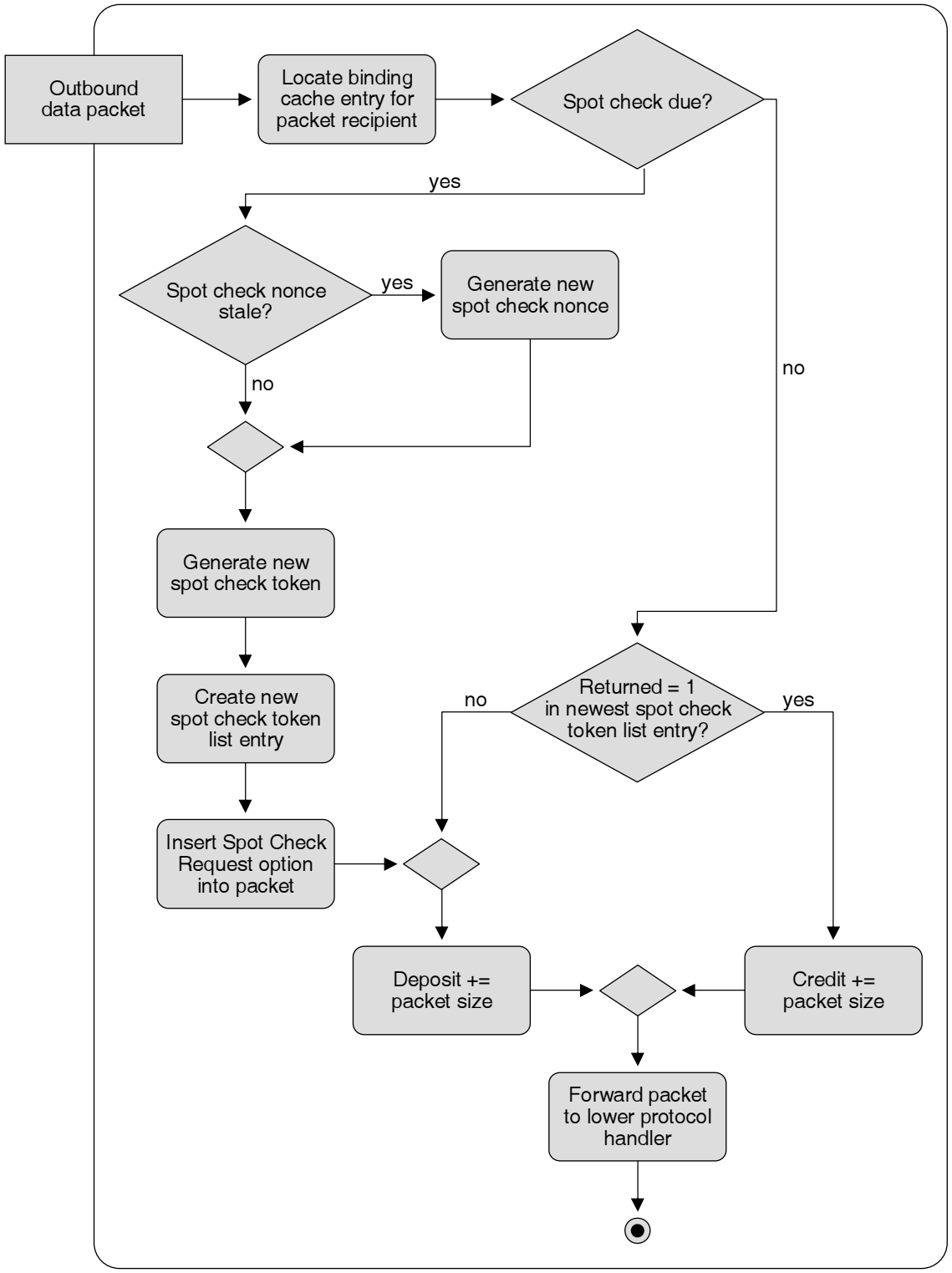



Figure 4.14: Correspondent node operation for packet transmission

oldest entry in the list, and populates the Spot Check Token field with the generated token and the Time Stamp field with the current time. In order to be able to access the entry efficiently, the entry is assigned a token index which is kept in the Spot Check Token Index field. The entry's Deposit field is initialized to the size of the outgoing packet in bytes. This accounts for the mobile node's expected effort for

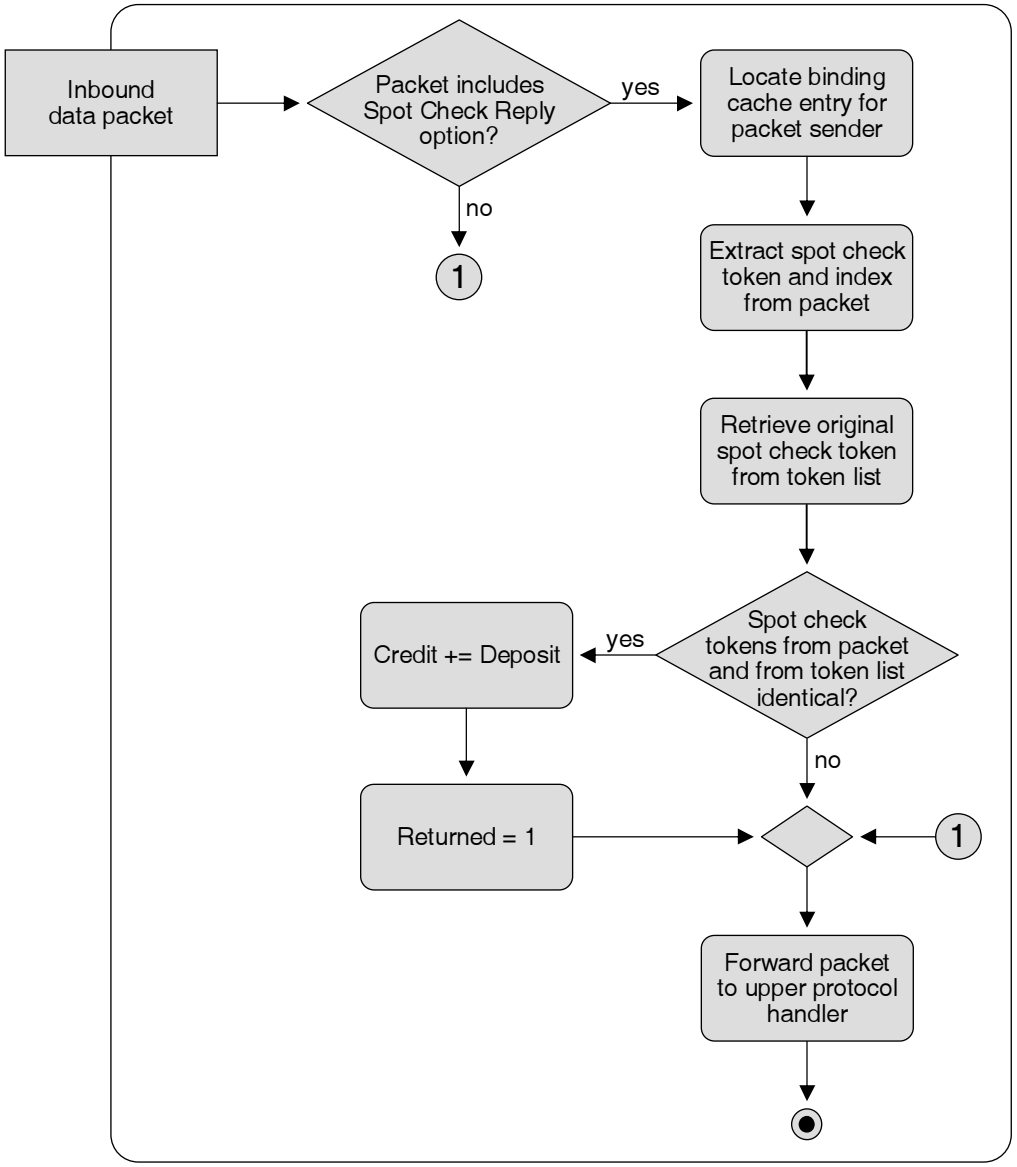



Figure 4.15: Correspondent node operation for packet reception

receiving the packet. Again, the deposit will be turned into credit when the token is received back from the mobile node. The Returned flag in the new entry is cleared. Finally, the correspondent node inserts into the packet an IPv6 Destination Options extension header with a Spot Check Request option containing the generated token, and forwards the packet to the link layer for transmission.

Since the correspondent node must determine for each outgoing packet whether or not a spot check is due, it is important to arrive at this decision efficiently. The mobility protocol must accomplish a binding cache lookup anyway in order to obtain the mobile node's preferred IP address. The procedure described above requires an additional integer comparison of two time stamps—the current time and the value stored in the Time Stamp field in the newest token list entry. This should be acceptable even on resource-constrained correspondent nodes.

The procedure for handling inbound packets that include an IPv6 Destination Options extension header with a Spot Check Reply option is shown in figure 4.15. The correspondent node first locates the binding cache entry maintained for the mobile node that originates the received packet and extracts the spot check token as well as its index from the Spot Check Reply option. Based on the token index, the correspondent node retrieves the original token from the token list and compares it to the token extracted from the received packet. If the two tokens match, the effort that the mobile node has spent since the initiation of that spot check and the initiation of the next spot check can be turned into credit. For this, the correspondent node adds the value of the Deposit field in the token list entry to the credit counter in the binding cache entry and sets the Returned flag in the token list entry. The packet is then forwarded either to the handler of the next option in the Destination Options extension header, if any, or to the upper-layer protocol. Incoming packets without an included Spot Check Reply option do not receive spotcheck-specific treatment.

### 4.3.7 Mobile Node Operation

As with the correspondent node operation, spotcheck-related activities at the mobile node are also triggered by either the reception of a packet that contains an IPv6 Destination Options extension header with an included Spot Check Request option, or the arrival of an outgoing packet from an upper-layer protocol. Figure 4.16 shows an activity diagram for the procedure anchored at the reception of a packet with a Spot Check Request option from a correspondent node. The mobile node first locates the binding update list entry that it maintains for the correspondent node. It then creates a new entry in the spot check token buffer, extracts the spot check token and its associated index from the received packet, and copies these values to the new buffer entry's Spot Check Token and Spot Check Token Index fields, respectively. The Time Stamp field in the entry is set to the current time. Finally, the mobile node forwards the packet to the upper protocol layer.

Figure 4.17 shows the steps that the mobile node follows when it has a packet to send to the correspondent node. The mobile node first locates the respective binding update list entry and purges the spot check token buffer of any stale entries. A token buffer entry is stale if and only if the difference between the current time and the value in the entry's Time Stamp field is greater than MAX_SPOT_CHECK_DURATION. If the spot check token buffer is empty at this point—be it due to the removal of stale entries, or because it did not contain any entries in the first place—, the outgoing packet is forwarded to the lower protocol layer as is. In case one or more tokens remain in the buffer, the mobile node inserts into the packet an IPv6 Destination Options extension header with a Spot Check Reply option, to which it copies all tokens from the buffer along with the respective indices. The mobile node finally flushes the entries from the token buffer and forwards the packet to the lower-layer protocol.

## 4.4 Summary

Credit-Based Authorization was designed as a mechanism that protects concurrent reachability verification against misuse for redirection-based flooding attacks or endured redirection-based reflection attacks. As these attacks attempt to trick a correspondent node into sending a large amount of data to an unverified on-link IP

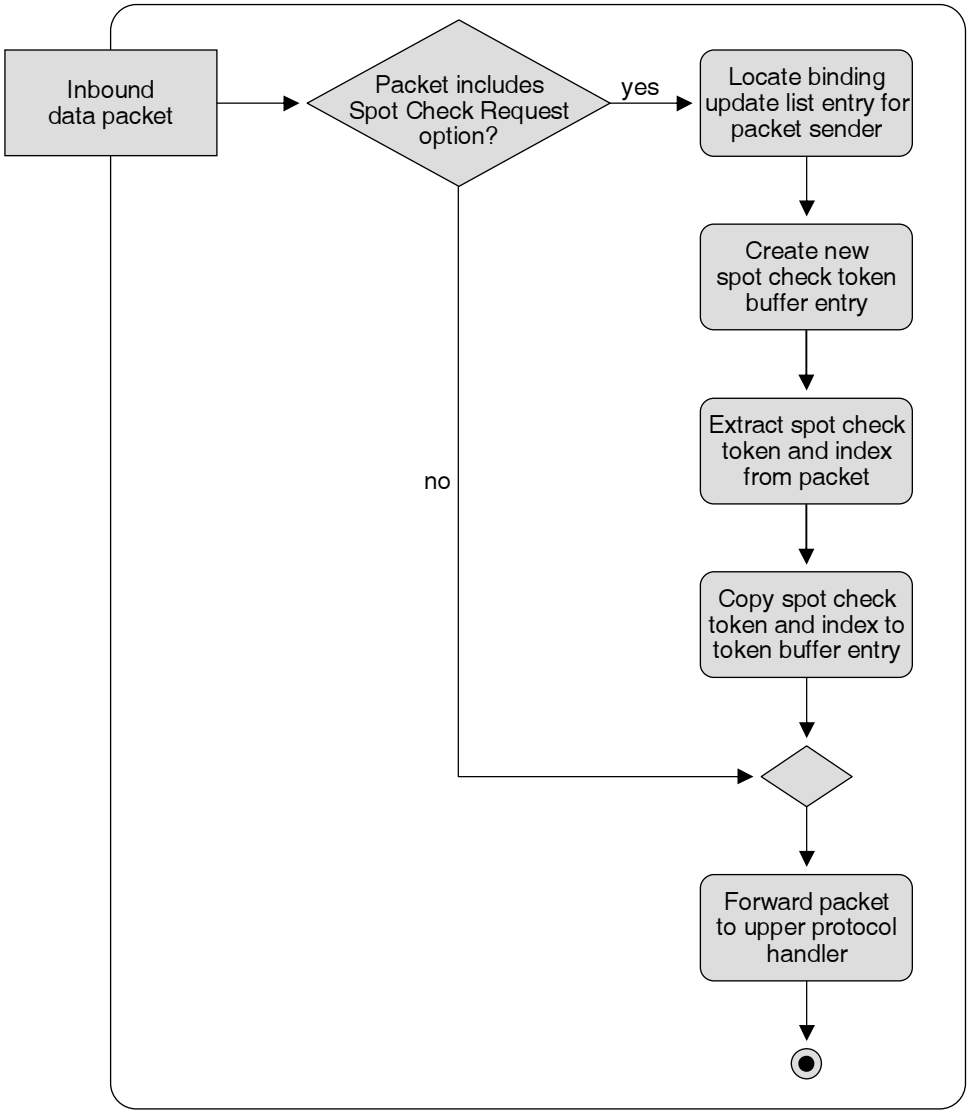



Figure 4.16: Mobile node operation for packet reception

address, Credit-Based Authorization counters the attacks by limiting that amount of data. The limit is determined dynamically such that the effort that a correspondent node can spend on sending payload packets to a mobile node's unverified IP address is at most as high as the effort that the correspondent node has recently seen the mobile node spend in communicating with the correspondent node. With amplification being the single most important property of redirection-based flooding, Credit-Based Authorization thus effectively defeats this type of flooding attack. The limit is furthermore reevaluated only when all of the mobile node's IP addresses are verified. Payload packets hence cannot be redirected to an unverified IP address on an endured basis, which precludes endured redirection-based reflection attacks.

A node's effort is practically measured in terms of sent or received payload packet bytes, depending on the mode of Credit-Based Authorization: In Inbound mode, the mobile node earns one credit for each payload packet byte it sends, and one acquired credit can be turned into one payload packet byte sent by the correspondent node to an unverified IP address. In Outbound mode, the correspondent node gives the

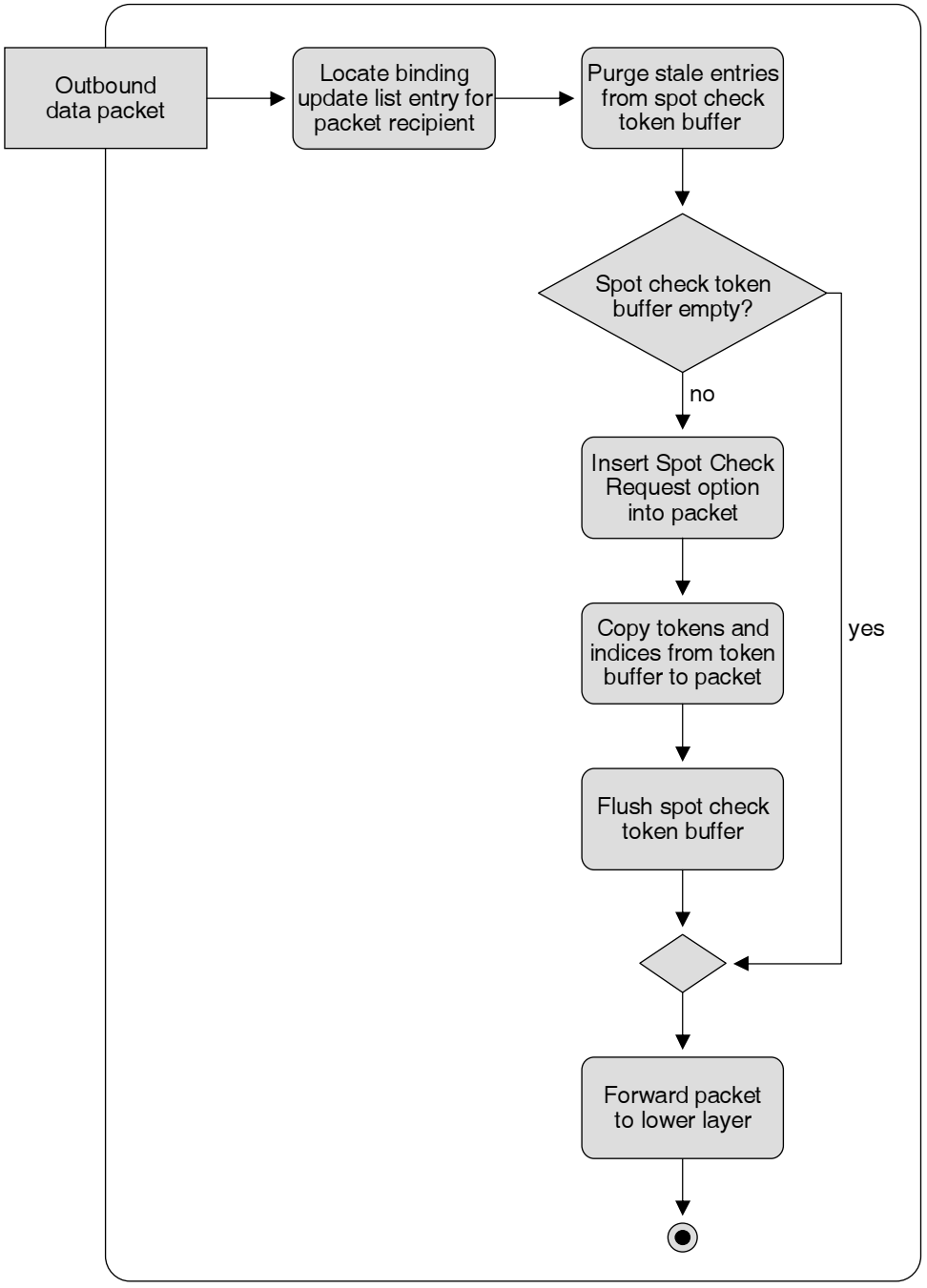



Figure 4.17: Mobile node operation for packet transmission

mobile node one credit for each received payload packet byte, and the credit can be redeemed as in Inbound mode. The actual delivery of payload packets to the mobile node is spot-checked periodically by the correspondent node through the exchange of an unpredictable token. Tokens are piggybacked onto payload packets to limit the extra protocol overhead they cause. The two Credit-Based Authorization modes reflect the trade-off between simplicity and suitability to different application types. Inbound mode is simple and easy to implement, yet the need for a mobile node to obtain credit by sending payload packets may lead to a shortage in credit during handover if application traffic patterns are asymmetric. Outbound mode works with

any application type, but it is more complex to implement and requires in-band signaling for spot-checking.

The credit that a mobile node acquires ages gradually over time so that it represents only the mobile node's *recent* effort. This prevents an attacker from slowly accumulating a high amount of credit.

# 5. Early Binding Updates

For a correspondent node to execute Credit-Based Authorization and send a mobile node payload packets early on after a change in IP connectivity, it must accept a binding update from the mobile node on a tentative basis before obtaining insurance that the mobile node is reachable at the claimed, new on-link IP address. Mobility protocols which execute binding update and reachability verification in this order allow for easy integrability of Credit-Based Authorization. For example, correspondent nodes with support for the mobility extensions of the Host Identity Protocol probe a mobile node's new on-link IP address after the binding update. Credit-Based Authorization allows them to send payload packets to the new IP address already before reachability verification completes. On the other hand, the requirement for an early binding update and a subsequent reachability verification calls for changes to those mobility protocols that integrate the reachability verification with the binding update. This is the case for Mobile IPv6, where the mobile node must first obtain home and care-of keygen tokens via the return routability procedure before it can initiate a binding update.

Given the anticipated importance of Mobile IPv6 in the next-generation Internet, the compatibility of Credit-Based Authorization and Mobile IPv6 would be of true benefit. This chapter is hence dedicated to the integration of Credit-Based Authorization and Mobile IPv6. It introduces *Early Binding Updates for Mobile IPv6*, an optional and fully backward-compatible extension to Mobile IPv6, which separates the processes of binding update and reachability verification. Correspondent nodes register a new care-of address and send payload packets thereto in parallel with reachability verification. Early Binding Updates also facilitate proactive mobility management for those scenarios where mobile nodes can foresee a change in IP connectivity. This chapter proposes a framework that allows a mobile node to obtain the set of subnet prefixes in use on a prospective new access link and configure a new care-of address before actually handing over to that access link. As with Credit-Based Authorization itself, the objectives for Early Binding Updates again derive directly from the criteria that were used in the assessment of candidate mechanisms for IP address verification in section 3.3.1.

## 5.1 Technical Approach

Early Binding Updates are composed of a number of constituent Mobile IPv6 optimizations that together enable a mobile node to quickly continue bidirectional communications after a change in IP connectivity. Some of the optimizations tune protocol behavior within the boundaries of the base specification. A Mobile IPv6 implementation that supports these optimizations is still standard-compliant. The remaining optimizations require modifications to the specification, but are designed to be backwards-compatible. Since Early Binding Updates include a concurrent care-of address test, they also incorporate Credit-Based Authorization.

### 5.1.1 Standard-Compliant Optimizations

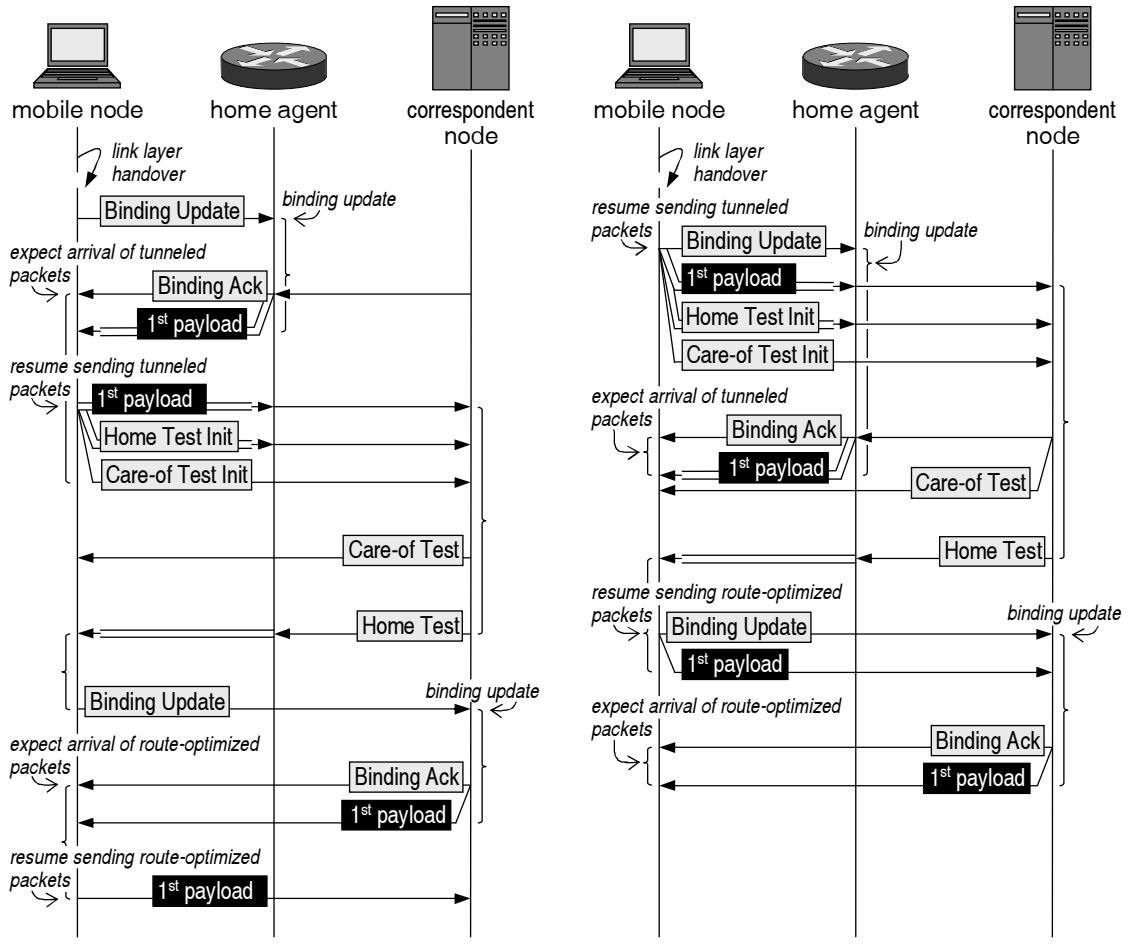
Standard Mobile IPv6 leaves mobile nodes liberties with respect to scheduling signaling and payload packets. Mobile nodes can leverage this freedom to reduce the handover delay of a correspondent registration without special support on the correspondent node side. They obtain the highest benefit through support of the following three standard-compliant optimizations:

1. *Parallel home registration and reverse tunneling* — A mobile node that communicates with a correspondent node by means of bidirectional tunneling may resume payload packet transmission after a handover once it has sent a Binding Update message to the home agent. It does not necessarily have to wait until it receives a Binding Acknowledgment message from the home agent. Such parallelization reduces the handover delay for payload packets from the mobile node by one round-trip time between the mobile node and the home agent.

2. *Parallel home registration and return routability procedure* — A mobile node that uses route optimization may initiate the return routability procedure right after it has sent the Binding Update message to the home agent. The home registration and the return routability procedure can then proceed in parallel. This reduces the handover delay for traffic in either direction by a round-trip time between the mobile node and the home agent because the latency of the home registration is subsumed by that of the return routability procedure.

3. *Parallel correspondent registration and route optimization* — A mobile node that uses route optimization may resume the transmission of payload packets to a correspondent node as soon as it has sent a Binding Update message to the correspondent node. It does not necessarily need to wait for a Binding Acknowledgment message from the correspondent node if it requests one by setting the Acknowledge flag in the Binding Update message. The handover delay for the payload packets that the mobile node sends can thus be reduced by the latency of the correspondent registration, which amounts to one round-trip time between the mobile node and the correspondent node.

The above Mobile IPv6 optimizations are mutually independent and hence allow for a variety of standard-compliant implementations. To reasonably represent the diversity within the scope of this thesis, considerations will be focused to two protocol




(a) Conservative Mobile IPv6 (b) Optimistic Mobile IPv6

Figure 5.1: Conservative Mobile IPv6 vs. optimistic Mobile IPv6

variants: *Optimistic Mobile IPv6* includes all of the standard-compliant optimizations defined above, and *conservative Mobile IPv6* includes none of them. Analogously, an *optimistic mobile node* uses optimistic Mobile IPv6, and a *conservative mobile node* sticks to conservative Mobile IPv6.

Figure 5.1 juxtaposes home and correspondent registrations of conservative Mobile IPv6 with those of optimistic Mobile IPv6, and it indicates the times at which payload packets can again be sent or received via a particular path. A conservative mobile node resumes sending reverse-tunneled payload packets when it receives a Binding Acknowledgment message from the home agent, whereas an optimistic mobile node begins reverse tunneling already after dispatching the Binding Update message to the home agent. The home agent updates its binding and the remote endpoint of the tunnel to the mobile node's new care-of address upon the reception of the Binding Update message. Payload packets from correspondent nodes, intercepted at the mobile node's home address, are tunneled to the new care-of address as of then. The operation of the home agent is independent of whether the mobile node is conservative or optimistic. In either case, the mobile node may receive encapsulated payload packets roughly at the time the home agent's Binding Acknowledgment message is received.

Route-optimized payload packets are not transmitted by a conservative mobile node until it receives a Binding Acknowledgment message from the correspondent node, while an optimistic mobile node starts sending route-optimized payload packets directly after it has dispatched the Binding Update message for the correspondent node. The correspondent node updates its binding to the mobile node's new care-of address upon the reception of the Binding Update message and redirects subsequent payload packets to the mobile node as of then. Since the operation of the correspondent node is independent of the Mobile IPv6 variant on the mobile node, the mobile node can in any case expect to receive route-optimized payload packets as of the reception of the correspondent node's Binding Acknowledgment message, provided that it has set the Acknowledge flag in the Binding Update message.

Overall, optimistic Mobile IPv6 fully reestablishes communications via bidirectional tunneling one round-trip time between the mobile node and the home agent earlier than conservative Mobile IPv6 does. Route-optimized traffic is restored two round-trip times sooner with optimistic Mobile IPv6 than with conservative Mobile IPv6: one round-trip time between the mobile node and the home agent plus another round-trip time between the mobile node and the correspondent node.

Unfortunately, neither of the two most popular open-source Mobile IPv6 implementations currently fully supports optimistic Mobile IPv6: Kame-Shisa [119], the implementation for FreeBSD, provides only conservative Mobile IPv6. Mobile IPv6 for Linux (MIPL) [95] allows a mobile node to initiate the home registration and the care-of address test in parallel, and is therefore not strictly conservative. But it does not go as far as permitting the entire return routability procedure to concur with the home registration. The mobile node must still wait for a Binding Acknowledgment message from the home agent until it can send a Home Test Init message. This approach in most cases does not improve handover delays because the latency of the home address test typically subsumes the latency of the care-of address test anyway. The behavior of MIPL is of benefit only when a previously acquired, still valid home keygen token allows the mobile node to skip the home address test.

### 5.1.2 Optimizations Requiring Changes to the Standard

The optimizations possible within the boundaries of the Mobile IPv6 RFC allow an optimistic mobile node to obtain significantly higher performance than a conservative mobile node in the general case. Yet, even with optimistic Mobile IPv6, it still takes a total of three round-trip times—one between the mobile node and the home agent, one between the home agent and the correspondent node, and another one between the mobile node and the correspondent node—until both end nodes have resumed bidirectional, route-optimized communications. The minimum handover delay for reactive end-to-end mobility management, however, is zero for payload packets which the mobile node sends, and one round-trip time between the mobile node and the correspondent node until the mobile node has informed the correspondent node about its new care-of address and the first route-optimized payload packet has propagated to the new care-of address. Besides, optimistic Mobile IPv6 does not permit proactive mobility management. Minimum handover delay and, optionally, support for proactive mobility management can be facilitated only through modifications to the Mobile IPv6 specification. The following is a list of four such optimizations, which in conjunction with the optimizations for optimistic Mobile IPv6 constitute Early Binding Updates:

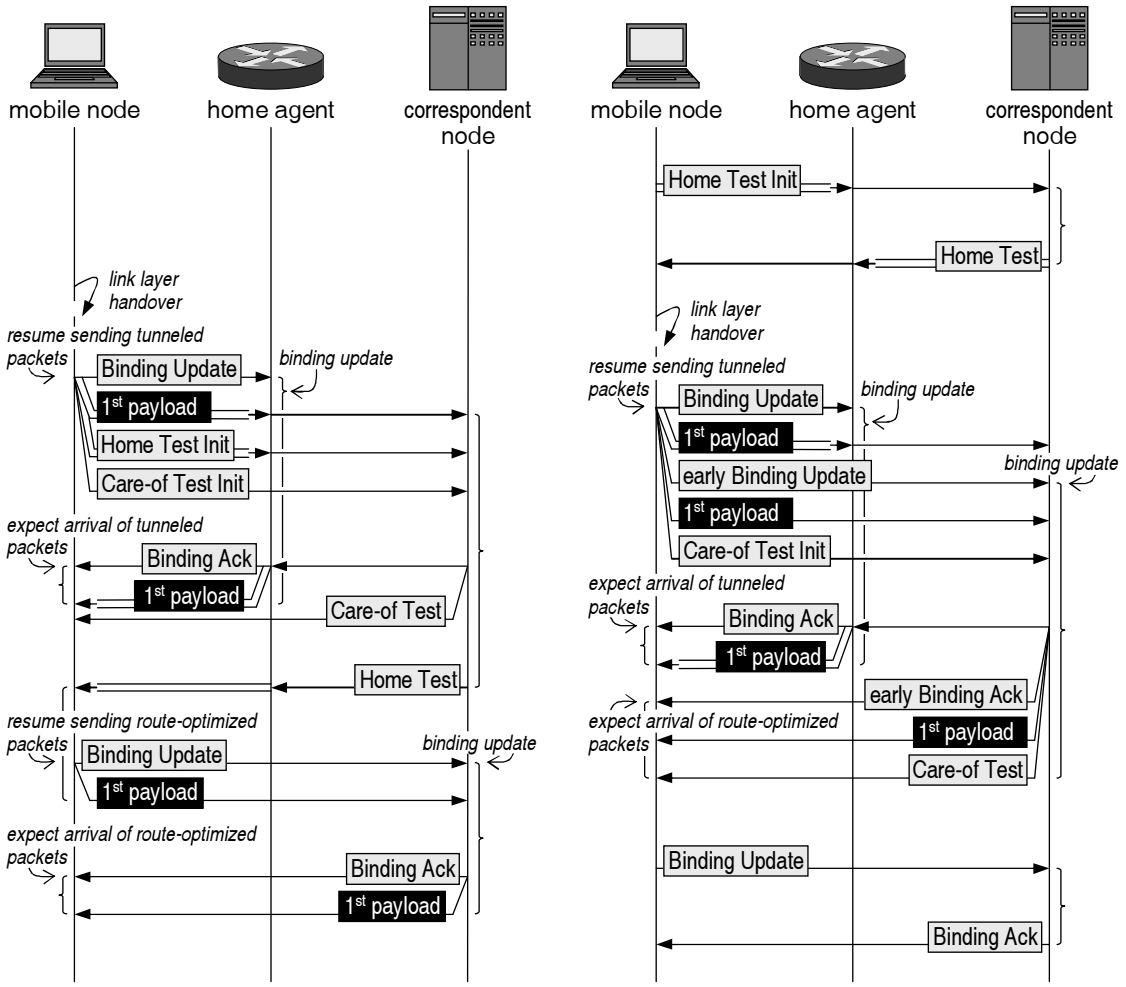
1. *Proactive home address test* — A mobile node performs a proactive home address test to acquire a home keygen token for a future handover ahead. Proactive home address tests save a potentially costly message exchange via the home agent during the critical handover period. The mobile node invokes proactive home address tests on a just-in-time basis if its link layer can announce a forthcoming handover to the IP layer. In the absence of adequate link layer functionality, the mobile node pursues a proactive home address test whenever the most recently obtained home keygen token is about to expire.

2. *Tentative binding* — The mobile node registers a *tentative binding* between its home address and a new care-of address as soon as it has generated the new care-of address. The mobile node does not provide evidence of its reachability at the new care-of address for this *tentative binding update*. The correspondent node accepts an early Binding Update message, which the mobile node authenticates only with a proactively acquired home keygen token. (Recall from section 2.3.3 that a standard Binding Update message is authenticated with home and care-of keygen tokens.) This facilitates a subsequent, concurrent care-of address test. In reactive mobility management, the tentative binding update happens from the new access link right after the mobile node has configured the new care-of address. In proactive mobility management, the tentative binding update happens already from the old access link, after the mobile node has anticipated the link layer handover and configured a new care-of address.

3. *Concurrent care-of address test* — Once a correspondent node has registered a tentative binding, it redirects payload packets for the mobile node to the new care-of address, although it has not yet verified at that time whether the mobile node is indeed reachable at the new care-of address. The care-of address test proceeds concurrently with regular communications. In conjunction with a proactive home address test, the return routability procedure can so be completely removed from the critical handover period. The correspondent node utilizes Credit-Based Authorization to prevent misuse of the concurrent care-of address tests for redirection-based flooding attacks.

4. *Parallel home and correspondent registrations* — A mobile node may register a tentative binding despite a pending home registration. This in particular applies to tentative bindings, which should happen in parallel with the corresponding home registration.

Figure 5.2 visualizes the differences between optimistic Mobile IPv6 and Mobile IPv6 with Early Binding Updates. Optimistic Mobile IPv6 already provides optimal handover delays for the home registration, so it does not differ from Mobile IPv6 with Early Binding Updates in this respect. The mobile node can in both cases resume sending reverse-tunneled payload packets once it has dispatched the Binding Update message for the home agent. And the home agent redirects tunneled payload packets for the mobile node to the new care-of address when it receives the Binding Update message, so that the first tunneled payload packet may arrive at the new care-of address just after the home agent's acknowledgment. However, while an optimistic mobile node must wait for both the home registration and the return routability procedure to complete before it can initiate the correspondent registration, a mobile




(a) Optimistic Mobile IPv6 (b) Mobile IPv6 with Early Binding Updates

Figure 5.2: Optimistic Mobile IPv6 vs. Mobile IPv6 with Early Binding Updates

node using Early Binding Updates can register a tentative binding already in parallel with the home registration. The mobile node can then send the first route-optimized payload packet to the correspondent node right after it has sent the early Binding Update message. The correspondent node tentatively updates its binding for the mobile node upon the reception of the early Binding Update message, and it redirects payload packets to the mobile node's new care-of address from then on. Route-optimized communications are reestablished, when the mobile node receives the first route-optimized payload packet from the correspondent node at the new care-of address. This is likely to happen one round-trip time, measured between the mobile node and the correspondent node, after the transmission of the early Binding Update message, which is the minimum handover delay for end-to-end mobility management. The Care-of Test Init and Care-of Test messages are exchanged after the mobile node has dispatched the early Binding Update message, hence while regular communications are already being resumed.

A mobile node may perform proactive home address tests also when it stays on its home link in order to provision for an optimized future handover onto a foreign link. The proactively obtained home keygen token must in this case be remembered

without a binding at the respective correspondent node. However, mobile nodes normally store a home keygen token as part of a binding update list entry, which they are not required to keep without an active correspondent registration according to the Mobile IPv6 specification. Mobile IPv6 implementations hence usually remove the binding update list from memory once the mobile node connects to its home link. To support proactive home address tests on the home link, an implementation would have to retain existing and create new binding update list entries while the mobile node is at home, just as it does when the mobile node roams away from home.

The Mobile IPv6 specification recommends that a mobile node should always be able to communicate via its home address while a binding is cached at a correspondent node. The specification accordingly stipulates that the lifetime requested in a Binding Update message for a correspondent node should always be less than or equal to the remaining lifetime of the current binding at the home agent. A conservative or optimistic mobile node learns about the binding lifetime at the home agent with the home agent's Binding Acknowledgment message. However, a mobile node using Early Binding Updates pursues a tentative binding update at the correspondent node in parallel with the home registration, so it does not know how long the binding lifetime at the home agent is at the time it sends an early Binding Update message. The home registration may even be rejected by the home agent for administrative reasons. In order to bound the duration that a binding may exist at the correspondent node in the absence of an equivalent binding at the home agent, Early Binding Updates limit the lifetime for tentative bindings to 10 s. The tentative binding thus times out quickly in case the home registration is unsuccessful. If the Binding Update message for the home agent needs to be retransmitted due to packet loss, the tentative binding can be refreshed periodically in the meantime. Finally, the mobile node defers the transmission of the standard Binding Update message for the correspondent node until it has received the Binding Acknowledgment message from the home agent. The binding lifetime requested from the correspondent node can then be bound by the binding lifetime at the home agent.

## 5.2 Reactive Mobility Management

This section specifies the operation of a Mobile IPv6 variant that supports the previously introduced optimizations for Early Binding Updates and Credit-Based Authorization. Since the link and IP layer requirements of proactive mobility management go well beyond those of reactive mobility management, only reactive mobility management is considered here, and a description of proactive mobility management is left to section 5.3.

### 5.2.1 Mobility Management Architecture

Reactive mobility management can conceptually be combined with any mechanism for router discovery, movement detection, and IP address auto-configuration; the selection of a specific set of protocols does not affect Mobile IPv6 signaling. The mobility management architecture and protocol operation specified in the following are nonetheless limited to the particular combination of the DNA Protocol and Optimistic Duplicate Address Detection for the sake of manageability and clearness. The DNA Protocol requires the mobile node's IP layer to be notified when the underlying link layer attachment changes so that it can promptly initiate handover-related

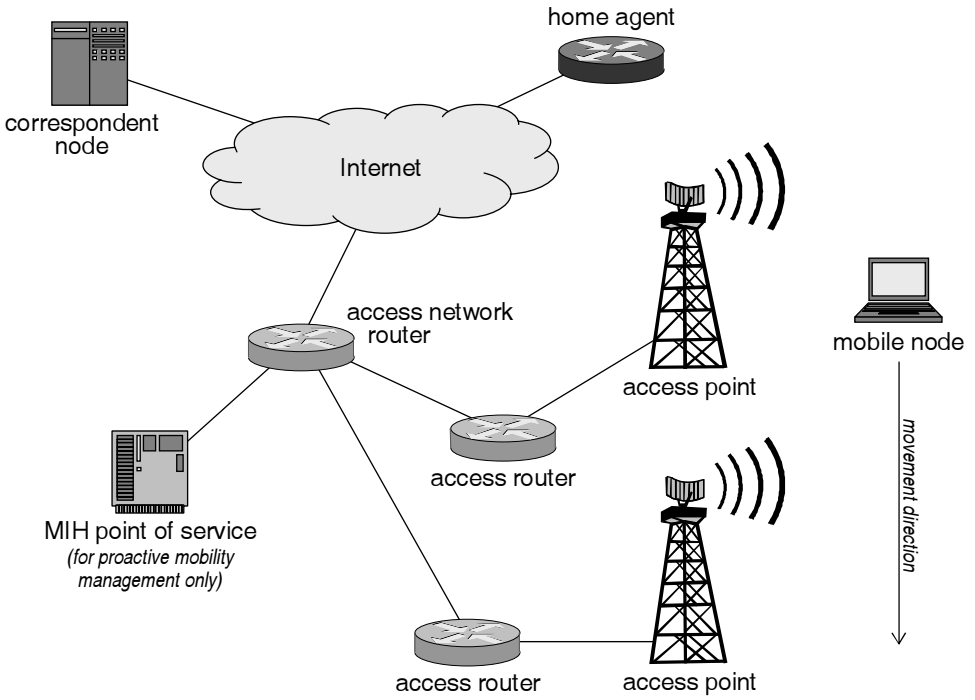



Figure 5.3: Network entities in reactive and proactive mobility management

activities. The architecture set forth in the following assumes that this notification is provided by Media Independent Handover Services. It is straightforward to integrate reactive mobility management with alternative mechanisms for router discovery, movement detection, and IP address auto-configuration, however.

Figure 5.3 depicts the network entities involved in the architecture for reactive mobility management in an example scenario. All of these are standard or optimized IPv6 or Mobile IPv6 entities. The MIH point of service shown in the bottom left corner of the figure is not part of the reactive mobility management architecture. It is needed only for proactive mobility management and should hence be ignored until later in this thesis.

Figure 5.4 details the functional components on the mobile node's network stack. It illuminates how the components interact with each other as well as with IPv6 entities in the access network and Mobile IPv6 entities elsewhere in the Internet. An MIH function serves as an abstraction shim layer between the link layer and the IP layer. The reactive mobility management architecture uses the MIH function simply as a relay for event notifications about handovers to a different access point from the link layer to the IP layer. One or multiple network interfaces, potentially for different access technologies, connect to the MIH function on the link layer side. The sole user of the MIH function at the IP layer is the *movement detection function*, or *MD function*. The MD function evaluates the events from the link layer, performs movement detection, and coordinates the handover-related activities of Neighbor Discovery, Stateless Address Autoconfiguration, and Multicast Listener Discovery accordingly.

No standardized protocol exists for the interaction between the MD function and Mobile IPv6. There is also lack of a widely accepted interface through which the

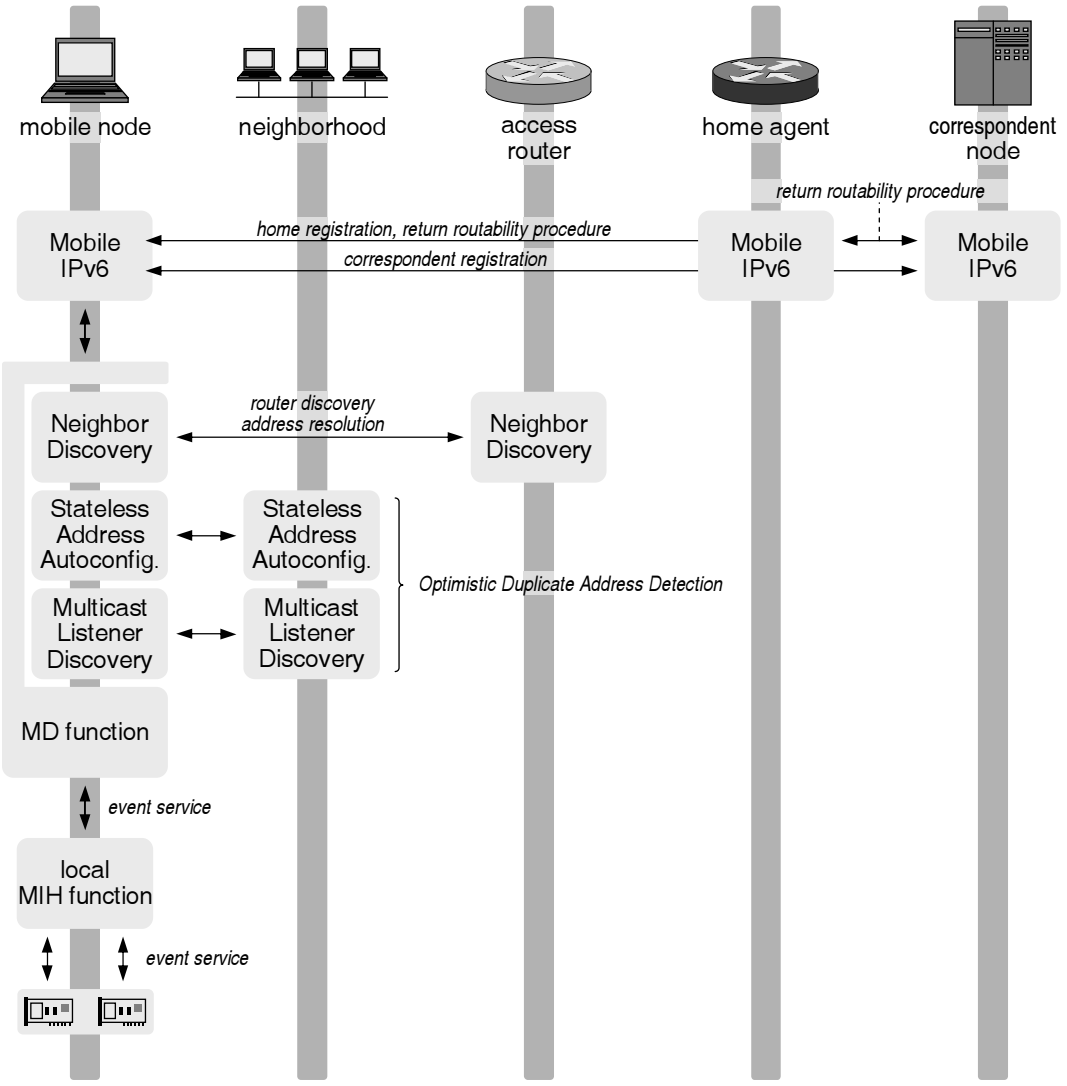



Figure 5.4: Functional network stack components in reactive mobility management

MD function could initiate activities of Neighbor Discovery, Stateless Address Autoconfiguration, and Multicast Listener Discovery. All of this intra-IP-layer signaling typically encompasses a set of proprietary messages or function calls specific to the network stack software.

### 5.2.2 Protocol Operation

Figure 5.5 depicts Mobile IPv6 and other IP layer signaling exchanged between the mobile node, its access router, home agent, and correspondent node when reactive mobility management is realized through Mobile IPv6 with Early Binding Updates. The figure also shows the implementation-specific signaling messages exchanged between the MD function and Mobile IPv6 internal to the mobile node. The circled numbers in the figure mark important steps in the handover procedure that are referenced in the text. A mobile node may in general communicate with multiple correspondent nodes in parallel, but for the purpose of clarity, the ensuing explanation is limited to a single correspondent node.

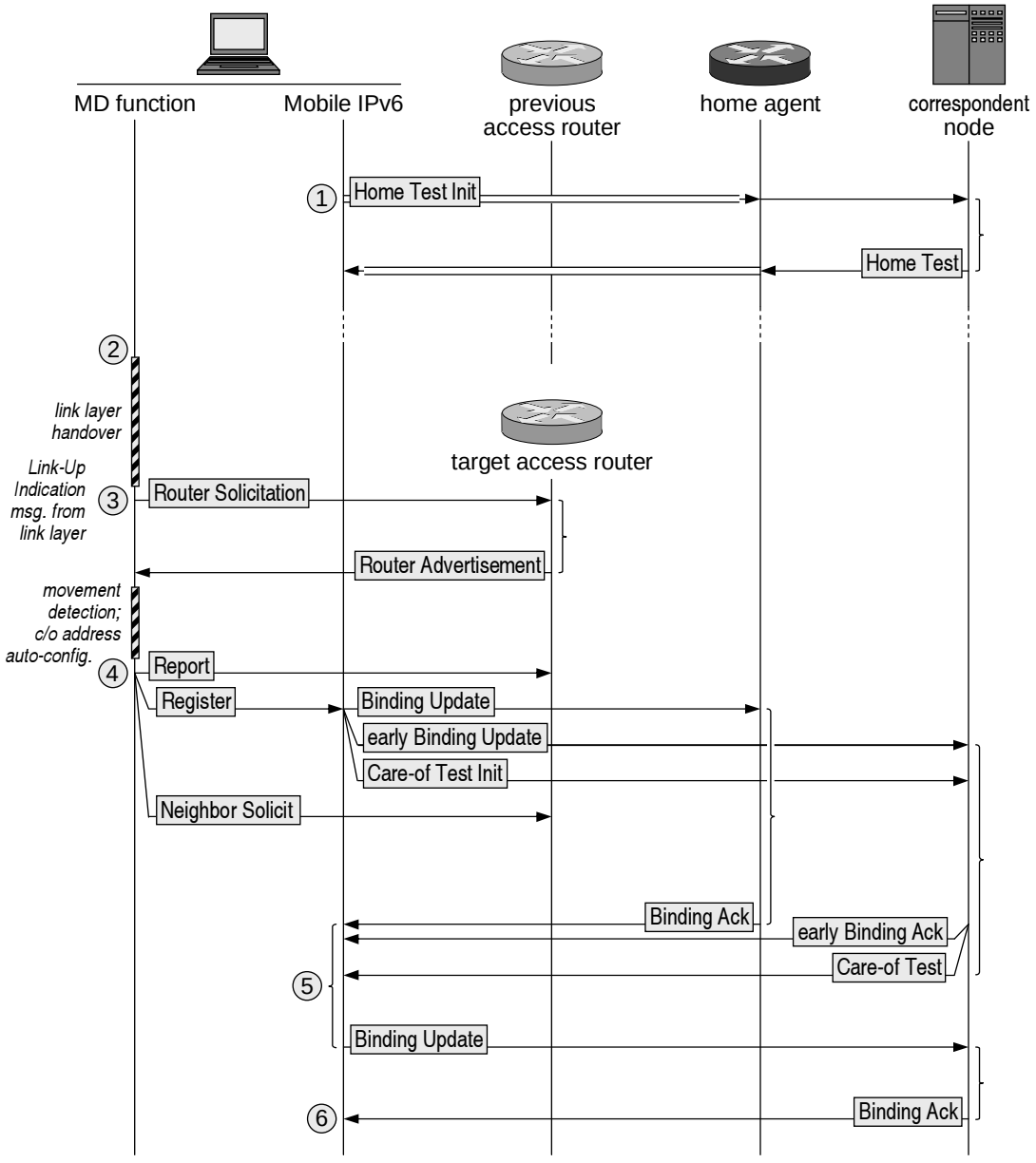



Figure 5.5: Protocol operation for reactive mobility management

### Proactive Home Address Test

The Mobile IPv6 instance on the mobile node sends a Home Test Init message prior to handover in order to proactively elicit a Home Test message from the correspondent node with a fresh home keygen token. The mobile node's link layer may provide a trigger announcing imminent handover so that the mobile node can initiate home address tests right in time. Alternatively, a proactive home address test may be initiated periodically whenever the home keygen token previously acquired from the correspondent node is about to expire. The validity of tokens is limited to 3.5 minutes to protect against time-shift attacks [94], so the interval between successive home address tests should be a little less. The home address test, beginning at (1) in the figure, delivers the home keygen token that is subsequently used during the handover. The Home Test Init and Home Test messages have the same syntax, and are processed in the same way, as specified in the Mobile IPv6 base specification, except that they are exchanged prior to handover and possibly on a periodic basis.

### Link Layer Handover

The handover phase begins with the link layer handover from the old access point to the target access point (2). Reactive mobility management does not require the link layer handover to be triggered by the IP layer or to be synchronized with any IP layer activities. The link layer handover time may hence be autonomously determined by the link layer itself. Once the connectivity with the target access point has been established, the link layer sends a Link Up Indication message to the MD function.

### Router Discovery

The Link Up Indication message from the link layer prompts the MD function to transmit a Router Solicitation message for router discovery (3). Neighboring access routers implement the DNA Protocol. So one of them sends a Router Advertisement message immediately, and additional routers send their advertisements each incrementally displaced by 20 ms. The MD function selects one of these candidate access routers as a target access router for handover. This is typically the access router from which the first Router Advertisement message is received.

### Movement Detection

The DNA Protocol enables reliable movement detection through the exchange of a Router Solicitation message and a single Router Advertisement message. The MD function hence performs movement detection once the first Router Advertisement message has been received. If the MD function thereby finds that IP connectivity did not change during the handover to the target access point, it skips any further IP layer handover activities. The mobile node can then continue to communicate via its current care-of address without Mobile IPv6 home and correspondent registrations. On the other hand, a new care-of address must be generated and registered with the home agent and the correspondent node in case IP connectivity has changed during the handover to the target access point.[1]

[1]There is a small chance that messages sent for IP address auto-configuration and Mobile IPv6 registrations collide with later Router Advertisement messages that other access routers may transmit. This chance is negligible, though, given the relatively generous spacing of successive Router Advertisement messages.

### IP Address Auto-Configuration

If the received Router Advertisement message indicates a change in IP connectivity, the MD function selects one or multiple of the new subnet prefixes, and generates new IP addresses from those based on Stateless Address Autoconfiguration and Optimistic Duplicate Address Detection. The new IP addresses are put into Optimistic state and are available for immediate use as a care-of address in Mobile IPv6 home and correspondent registrations, even though their uniqueness has not yet been determined. The mobile node must subscribe to the solicited-node multicast addresses that correspond to the new care-of addresses in order to conduct Optimistic Duplicate Address Detection, so the MD function transmits one or multiple Multicast Listener Discovery Report messages, depending on how many solicited-node multicast addresses must be subscribed to. Care-of addresses that match in the lower 24 bits map to the same solicited-node multicast address. A single Report message may hence be sufficient for multiple IP addresses, for example, if the interface identifiers in the IP addresses are all derived from the link layer address of the same network interface. The MD function then selects one of the new IP addresses as a new care-of address and triggers Mobile IPv6 home and correspondent registrations by sending a Register message to Mobile IPv6 (4). The Register message contains the new care-of address and the old care-of address that is to be replaced.

The normal procedure for a node to verify the uniqueness of a new IP address is to perform standard or Optimistic Duplicate Address Detection right after the new IP address has been generated and after the corresponding solicited-node multicast address has been subscribed to. A mobile node following the common behavior would thus send the Report and Neighbor Solicitation messages for the new care-of address before it initiates Mobile IPv6 home and correspondent registrations. The disadvantage of this signaling order is that any defending Neighbor Advertisement message, sent in response to one of the Neighbor Solicitation messages, is at risk of colliding with the Mobile IPv6 messages that the mobile node transmits at around the same time. The consequence of such a collision would be detrimental as the duplicated IP address would continue to be used by both the original owner and the mobile node. One of the nodes may then be unable to communicate with correspondent nodes off the local link, depending on which link layer address access routers associate the duplicate IP address with. The mobile node is likely to be able to communicate first because the access router memorizes the mobile node's link layer address during router discovery. However, Neighbor Advertisement messages sent for the duplicate IP address during later neighbor unreachability detection usually have the Override flag set, so they might cause the access router to replace the mobile node's link layer address with the original IP address owner's.

To reduce the collision probability during Optimistic Duplicate Address Detection signaling, the mobile node defers the uniqueness verification of any new IP address until the Mobile IPv6 home and correspondent registrations have been initiated. Mobile IPv6 messages return only after a round-trip time to the topologically closest home agent or correspondent node. This pause should be sufficient for any IP address duplicates to be discovered. The Optimistic Duplicate Address Detection specification facilitates the deferral as it allows a new IP address to be used before the transmission of a Neighbor Advertisement message commences the uniqueness verification. Multicast Listener Discovery Report messages are transmitted as soon

as the new IP addresses have been generated, but they do not solicit a response that may lead to collisions.

### Mobile IPv6 Registrations

Mobile IPv6 initiates the home and correspondent registrations necessary to update existing bindings to the new care-of address as it receives the Register message from the MD function (4). Early Binding Updates do not modify the procedure for the home registration, except that it is parallelized with the correspondent registration. It consists of the standard Binding Update and Binding Acknowledgment messages exchanged between the mobile node and the home agent, as shown in figure 5.5. The correspondent registration is augmented to include an initial tentative binding update when a received Register message indicates that the binding at a correspondent node has become stale, but the mobile node lacks a valid care-of keygen token with which it could authenticate a standard Binding Update message for the correspondent node. A valid care-of keygen token is typically known only when the mobile node returns to an access link that it has already recently visited and it ends up auto-configuring a previous care-of address again. A tentative binding update is therefore needed in most cases. It is initiated with an early Binding Update message, as shown in figure 5.5. The early Binding Update message is directly followed by a Care-of Test Init message to begin the concurrent care-of address test right away. The transmission of payload packets through the reverse tunnel to the home agent may resume once the Binding Update message for the home agent has been dispatched, and route-optimized payload packets can be sent via the new care-of address right after the early Binding Update message.

The Binding Update message for the home agent has the same contents and semantics as in standard Mobile IPv6, and it is likewise protected through IPsec transport mode. The home agent updates the remote endpoint of the forward tunnel to the mobile node's new care-of address when it receives the message. Outgoing payload packets for the mobile node are thus delivered to the new care-of address. The home agent completes the home registration by sending the mobile node an IPsec-protected Binding Acknowledgment message.

The early Binding Update message for the correspondent node. It has the same syntax as a standard Binding Update message, but its message authentication code is calculated with a binding management key that is a one-way hash on a home keygen token only. This allows the mobile node to authenticate the early Binding Update message with the home keygen token most recently acquired through a proactive home address test, and to obtain a care-of keygen token through a concurrent care-of address test afterwards.

The care-of nonce index in an early Binding Update message is set to zero since the message authentication code as defined above does not incorporate a care-of keygen token. In this respect, the early Binding Update message is similar to a standard Binding Update message that the mobile node sends to a correspondent node for the purpose of deregistration when it returns to the home link after a period of roaming. The lack of a care-of keygen token received at the new care-of address further implies that an early Binding Update message does not verify the mobile node's reachability at the care-of address. The mobile node hence follows up with a proof of reachability when it sends to the correspondent node a standard Binding Update message at a

later stage during the correspondent registration. This will be explained below. The lifetime that the mobile node may request in an early Binding Update message is limited to 10 s.

When the correspondent node receives the early Binding Update message, it tentatively binds the mobile node's home address to the new care-of address, thereby redirecting outgoing payload packets for the mobile node to the new care-of address. The new care-of address is set to Unverified state due to the absence of a proof of the mobile node's reachability at the care-of address. The lifetime granted for the tentative binding must not exceed 10 s. It may theoretically be lower, yet the already low requested lifetime renders any further reduction unreasonable. If the Acknowledge flag is set in the early Binding Update message, the correspondent node sends an early Binding Acknowledgment message back to the mobile node. The message authentication code in the early Binding Acknowledgment message is calculated analogously to the message authentication code for the early Binding Update message; the binding management key is again a one-way hash on the home keygen token only. The correspondent node further generates a fresh care-of keygen token for the mobile node when it receives the Care-of Test Init message and sends this back to the mobile node with a Care-of Test message. The Care-of Test Init and Care-of Test messages have the same syntax, and are processed in the same way, as in base Mobile IPv6.

After a Binding Acknowledgment message indicating a successful home registration has been received from the home agent, and after a fresh care-of keygen token has been obtained from the correspondent node, the mobile node sets about replacing the tentative binding at the correspondent node with a standard binding (5). It thus sends a standard Binding Update message to the correspondent node. The message authentication code is now calculated as defined in the Mobile IPv6 RFC, namely with a binding management key that is a one-way hash on the concatenation of the care-of keygen token from the received Care-of Test message and the home keygen token from the most recently received Home Test message. The home and care-of nonce indices are set accordingly. The binding lifetime requested by the mobile node in the standard Binding Update message can be as high as 7 minutes, the maximum that the Mobile IPv6 RFC permits for correspondent registrations. However, the requested binding lifetime should not exceed the lifetime granted for the home registration, which the mobile node can read from the Binding Acknowledgment message received from the home agent.

The correspondent node verifies the validity of the message authentication code as specified in the Mobile IPv6 RFC when it receives the standard Binding Update message and, provided the message authentication code is correct, changes the state of the mobile node's care-of address from Unverified to Verified. The correspondent node extends the lifetime of the mobile node's binding from its tentative value to the value requested in the standard Binding Update message, although a limit of 7 minutes is applied. If the Acknowledge flag in the message is set, the correspondent node returns a standard Binding Acknowledgment message to the mobile node (6).

Should the mobile node set the Acknowledge flag in the early Binding Update message, but fail to receive a corresponding Binding Acknowledgment message within appropriate time, it may retransmit the early Binding Update message every 1 s up to 3 times until it either receives an acknowledgment for the early Binding Update

message or meets the preconditions to send a standard Binding Update message. The retransmission algorithm for early Binding Update messages deviates from the standard retransmission algorithm specified in the Mobile IPv6 RFC in that it performs three retransmissions without exponential back-off and then ceases to send any further early Binding Update messages. (The standard retransmission algorithm uses exponentially increasing intervals between successive retransmissions, but may continue indefinitely.) This shorter, yet more aggressive behavior aims to accelerate the recovery from the loss of an early Binding Update message without causing too much additional signaling overhead.

When the mobile node can be sure that the correspondent node has received its early Binding Update message, it may periodically refresh the tentative binding until it can send a standard Binding Update message. This may be necessary if packet loss and extensive buffering delays in the network increase the latency of the care-of address test. The refresh of the tentative binding prevents the correspondent node from deleting the binding prematurely and falling back to bidirectional tunneling via the mobile node's home address. Lack of a binding would also cause the correspondent node to drop any route-optimized payload packets that it subsequently receives from the mobile node. The mobile node resends the standard Binding Update message—be it in case of a missing acknowledgment or in order to refresh an existing binding—according to the rules of standard Mobile IPv6.

#### Optimistic Duplicate Address Detection

The MD function initiates Optimistic Duplicate Address Detection for the new care-of address once the initial standard and early Binding Update messages as well as any Care-of Test Init messages have been sent. It transmits one Neighbor Solicitation messages per new IP address in order to solicit a Neighbor Advertisement message from any neighboring nodes that happen to use the same IP address. An IP address is transferred from Optimistic state to Preferred state when no defending Neighbor Advertisement message has been received after a period of 1 s.

In the unlikely case that a new IP address turns out to be already in use by a node on the new link, the mobile node receives a Neighbor Advertisement message from the true owner of the IP address in response to the respective Neighbor Solicitation message. The mobile node must then invalidate any previous home and correspondent registrations for which the duplicate IP address was used. It generates a new IP address and registers this as a new care-of address as specified above. The mobile node may in particular again begin correspondent registrations with a tentative binding update since the change of the care-of address has invalidated any previously obtained care-of keygen tokens. Once the initial Mobile IPv6 messages have been dispatched, the mobile node invokes Optimistic Duplicate Address Detection signaling to verify uniqueness of the new care-of address. The handover phase ends when all pending home and correspondent registrations are complete and all new care-of addresses have been transferred from Optimistic state to Preferred state.

### 5.2.3 Initial Correspondent Registration

The transmission of an early Binding Update message early on during a correspondent registration requires a mobile node to know a valid home keygen token. The mobile node obtains the home keygen token by means of proactive home address

tests, which it performs for each correspondent node registered in its binding update list. The mobile node is thus prepared to quickly initiate tentative binding updates in the event of a handover. On the other hand, the mobile node generally does not have a valid home keygen token when it contacts a new correspondent node. An early Binding Update message hence cannot be sent when a new session is established from a visited access link. The mobile node must then execute a full return routability procedure and subsequently send a standard Binding Update message, as defined in the Mobile IPv6 RFC. The mobile node initiates proactive home address tests after this initial correspondent registration so that Early Binding Updates can be applied during all subsequent handovers.

### 5.2.4 Message Types

Mobile IPv6 messages include a number that identifies them. These *message type values* are controlled by IANA. Mobile IPv6 extensions that introduce new messages may request type values for them from IANA. Different request and assignment procedures exist. In the special case of Mobile IPv6, new message type values are assigned [63, 90] for upcoming standards track RFCs or with the approval of the Internet Engineering Steering Group, which comprises the IETF chair and area directors. Assigning messages individual type values is technically convenient because it enables easy identification in the nodes that process them. On the other hand, it may not always be possible to obtain new message type values from IANA, as is the case for non-standards-track RFCs and experimental protocol deployment. It is then advantageous if the Mobile IPv6 extension at hand can do without new message type values.

Early Binding Updates for Mobile IPv6 use two special messages—early Binding Update and early Binding Acknowledgment messages—, but they function with and without individual type values for these messages. Individual message type values enable a correspondent node to directly determine for a received Binding Update message how to verify the included message authentication code. The same holds for a mobile node that receives a Binding Acknowledgment message. The latter is particularly helpful when both an early and a standard Binding Acknowledgment message are outstanding because the messages may potentially return in any order. Such a situation arises when the mobile node receives a Care-of Test message from a particular correspondent node, and accordingly sends a standard Binding Update message, before the early Binding Acknowledgment message arrives from the same correspondent node.

Message processing is slightly more complex if no individual mobility header type values are available for early Binding Update and early Binding Acknowledgment messages. Mobile and correspondent nodes supporting Early Binding Updates must then reserve the special care-of nonce index of zero for indicating that a message was authenticated without a care-of keygen token. When a mobile node sends an early Binding Update message, it sets the care-of nonce index included in the message to zero. When it sends a standard Binding Update message, the care-of nonce index is set to the non-zero value obtained from the correspondent node in a Care-of Test message. The correspondent node reads the care-of nonce index from a received Binding Update message in order to determine if the message is early or standard, and puts the new care-of address into Unverified or Verified state accordingly. The

mobile node must use a different method to identify a received Binding Acknowledgment message as early or standard because acknowledgments do not include a nonce indices. The mobile node hence matches a received Binding Acknowledgment message with the corresponding Binding Update message based on the sequence number. The sequence numbers are the same if the messages belong together, and they differ otherwise.

### 5.2.5 Discovering Compatibility

When a mobile node contacts a correspondent node for the first time, it usually does not know whether the correspondent node supports Early Binding Updates. It discovers this during the initial or the first few correspondent registrations. If the correspondent node turns out to not support Early Binding Updates, the mobile node records this information in the binding update list entry maintained for the correspondent node.

Compatibility discovery for Early Binding Updates functions differently than how a mobile node detects a correspondent node's standard Mobile IPv6 capability. Legacy IPv6 nodes without support for standard Mobile IPv6 do not understand the Binding Update message and return an ICMPv6 Parameter Problem message with a code indicating that the value in the Next Header field of the received packet's IP header is unknown. A mobile node that gets such a message from a correspondent node in response to a Binding Update message knows that the correspondent node does not support Mobile IPv6.

The procedure for detecting support for Early Binding Updates depends on whether separate message type values are used for early and standard Binding Update messages. If this is so, a correspondent node without support for Early Binding Updates does not recognize the new message type value of an early Binding Update message and sends a Binding Error message with a status code that indicates this. A mobile node that receives such a Binding Error message in response to an early Binding Update message knows that the respective correspondent node supports standard Mobile IPv6, but not Early Binding Updates.

Compatibility discovery is more complex if standard and early Binding Update messages share the same message type value. Correspondent nodes without support for Early Binding Updates do not reserve the care-of nonce index value of zero, but may use this value for normal nonce indexing. So when such a correspondent node reads the zero care-of nonce index from a received early Binding Update message, its response depends on whether it currently has a nonce with this nonce index. If all current nonces of the correspondent node have non-zero indices, the correspondent node does not attempt to verify the message authentication code in the early Binding Update message, but sends a Binding Acknowledgment message with a status code notifying the mobile node that the nonce it attempts to use has expired. This message tells the mobile node that the correspondent node does not support Early Binding Updates.

If the index of one of the correspondent node's current nonces is zero, the correspondent node erroneously attempts to verify the message authentication code of the received early Binding Update message, using the nonce pointed to by the zero index as a care-of nonce. The verification necessarily fails because the authenticator in the early Binding Update message was computed with a different binding

management key. The correspondent node silently discards the apparently improperly authenticated message and does not send a negative Binding Acknowledgment message. This makes it difficult for the mobile node to determine whether the registration fell victim to packet loss or if the correspondent node does not support Early Binding Updates. The mobile node may in such a case follow the normal algorithm for retransmitting early Binding Update messages, and possibly try again upon a later handover if this turns out to be unsuccessful. This also gives the correspondent node time to replace some of its nonces and eventually remove the nonce with index zero. When a zero nonce index is no longer in use, the correspondent node sends a negative Binding Acknowledgment message upon the receipt of an early Binding Update message, thus helping the mobile node to detect that it does not support Early Binding Updates. The Mobile IPv6 RFC recommends a nonce cache size of eight nonces and a nonce generation interval of 30 s. It therefore takes 4 minutes at the most until a nonce with index zero gets replaced by a new value.

The probability that a correspondent node has a nonce with index zero, and hence misinterprets a received early Binding Update message, is generally low: Nonce indices are 16 bits long, so there is room for 65536 different values. With a nonce cache size of eight nonces, chances are $8/65536 = 0.000122$ that a nonce index value of zero is in use at the time of a given correspondent registration. Correspondent nodes should hence in most situations send a negative Binding Acknowledgment message upon receipt of an unrecognized early Binding Update message and thereby facilitate straightforward compatibility discovery on the mobile-node side.

## 5.3 Proactive Mobility Management

Optimized reactive mobility management can eliminate the handover delay for payload packets that a mobile node sends, but a minimum black-out period of one round-trip delay between the mobile node and the correspondent node remains for payload packets that the mobile node receives: It always takes a one-way delay for a mobile node to register a new care-of address at a correspondent node, plus another one-way delay for the first payload packet to arrive at the new care-of address. Proactive mobility management can further reduce this residual delay.

### 5.3.1 Additional Requirements

Proactive mobility management requires advanced functionality at the link layer and the IP layer of a mobile node that goes beyond the unidirectional notification service used by the DNA protocol in reactive mobility management. It works best where the access network in addition provides a mechanism for the mobile node to retrieve handover-related information about geographically nearby points of attachment to which the mobile node may wish to hand over. The prerequisites are as follows:

1. *Handover anticipation* — A mobile node must be able to anticipate a forthcoming handover to a different access point so that it can initiate preparatory handover activities, including proactive Mobile IPv6 home and correspondent registrations. This requires special functionality at the link layer.

2. *Candidate access point discovery* — In order to select a suitable target access point for handover, the mobile node must be enabled to scan for available

candidate access points and measure their signal strengths. The selection of a candidate access point as a handover target may also be determined by other, possibly user-defined parameters such as available bandwidth, access latency, connection cost, or access provider.

3. *Cross-layer interaction and synchronization* — The mobile node's link and IP layers must interact and synchronize their respective handover-related activities. At a minimum, the link layer must notify the IP layer when a handover to a different access point is imminent, and the IP layer must be able to trigger the link layer handover to that access point when it decides that it is time to move.

4. *Proactive care-of address generation* — The mobile node must be able to generate a new care-of address prior to handover and use it for proactive home and correspondent registrations from the old link, while it must at the same time be able to defer any signaling necessary to verify the uniqueness of the care-of address until it arrives at the new link. Classic Stateless Address Autoconfiguration does not support this.

5. *Off-link subnet prefix discovery* — For the mobile node to generate a care-of address for a new link before it actually attaches to that link, the mobile node must be able to retrieve from the old link the set of subnet prefixes in use on the new link. The standard Neighbor Discovery protocol requires the mobile node to be on the new link in order to retrieve the subnet prefixes.

6. *Temporarily accepting payload packets from old care-of address* — The mobile node continues to send payload packets from its old care-of address via the old link after it has proactively initiated home and correspondent registrations for a new care-of address, and only switches to the new link after receiving an acknowledgment. The home agent and correspondent node may hence receive up to a round-trip time's worth of payload packets from the old care-of address even after the new care-of address has been registered. Both nodes would normally drop these packets, as described in section 5.1.1, and would thus defeat the objective of proactive mobility management for low handover delay and packet loss. The behavior of the home agent and the correspondent node must therefore be changed so that the nodes continue to accept packets from the old care-of address for while after a home or correspondent registration.

A variety of techniques are conceivable to satisfy the above requirements. Handover anticipation, candidate access point discovery, as well as cross-layer interaction and synchronization between the mobile node's link and IP layers could technically be realized through a proprietary mechanism. On the other hand, a standardized, uniform interface through which the IP layer can interoperate with any type of link layer can ease the development of network stack software substantially, in particular since the diversity of link layer technologies is high and is expected to further increase. The IEEE 802.21 Media Independent Event and Command Services provide such an interface. A proposal [128] from the IRTF MobOpts research group defines an alternative mechanism. Recent discussions indicate that this will eventually be merged into IEEE 802.21, however.

Optimistic Duplicate Address Detection is a standardized optimization for Stateless Address Autoconfiguration which permits proactive care-of address generation, and thus makes Stateless Address Autoconfiguration applicable to proactive mobility management. This is viable given that IPv6 nodes are required [73] to support Stateless Address Autoconfiguration and Duplicate Address Detection. Duplicate Address Detection must be performed for all unicast, non-anycast IPv6 addresses regardless of whether they are obtained through Stateless Address Autoconfiguration, DHCPv6, or manual configuration [130]. (Implementations must nonetheless be configurable to disable Duplicate Address Detection for particular network interfaces.) Optimistic Duplicate Address Detection provides the required interoperability and was hence chosen as a basis for the specification of proactive mobility management in this document.

Optimistic Duplicate Address Detection was not suited for proactive mobility management from the beginning, however. An initial version of the protocol hindered proactive mobility management in that it required a mobile node to send MLD Report and Neighbor Solicitation messages on the new link before the new IP address could be put to use. The protocol further suffered from a random desynchronization delay of between zero and 1 s for the MLD Report message [134]. Both deficiencies could, however, be resolved after they had been identified and discussed in the IETF IPv6 working group [138]. The version of Optimistic Duplicate Address Detection that was eventually published as RFC allows a mobile node to use an optimistically auto-configured care-of address prior to sending any messages on the new link. It hence facilitates proactive mobility management.

Different mechanisms have been proposed for off-link subnet prefix discovery, based on proxy router advertisements [70], information proliferated by roaming mobile nodes [131, 117], or formatted IEEE 802.11 service set identifiers [127]. However, all of these techniques are either not widely deployed, or integrated into a particular protocol, or both, while Mobile IPv6 was designed to function without special network infrastructure unless it is available ubiquitously. This contradiction makes it infeasible to prerequire any of the techniques for Mobile-IPv6-based proactive mobility management. The need for a widely deployed, generically applicable infrastructure for off-link subnet prefix discovery is to be satisfied by the IEEE 802.21 Media Independent Information Service. This standard can be expected to eventually provide a reasonable basis for proactive mobility management.

Regardless which mechanism the mobile node implements for off-link subnet prefix discovery, any required support required on the network side may at times not be provided by an access network to which the mobile node attaches. This holds all the more given that the Media Independent Information Service is still under development and it will likely take some time until the mechanism becomes deployed to the extent that it can be fully relied upon. It is therefore reasonable to define a backup solution with which proactive mobility management becomes applicable also in environments where neither the Media Independent Information Service, nor any comparable mechanism is available. The mobile node may then fall back to either or both of the following two autonomic approaches:

1. *Preconfiguration* — The mobile node provides an interface through which it can be preconfigured with mappings between the link layer addresses of access

points that it is likely to attach to and the sets of subnet prefixes in use on the links to which those access points connect.

2. *Caching* — The mobile node maintains a least-recently-used cache to map the link layer addresses of visited access points onto the sets of subnet prefixes learned at those access points.

Prefix preconfiguration and caching perform well in scenarios where mobile nodes tend to revisit a rather stable set of links, for example, at home or office environments, campuses, conferences, and local shopping centers. But there is obviously no benefit whenever a mobile node encounters an unknown access point.

### 5.3.2 Mobility Management Architecture

The requirements for proactive mobility management expounded in section 5.3.1 call for auxiliary mechanisms in the mobility management architecture, both on the mobile node and in the access network. A variety of techniques have been identified as suitable. Any of those, and possibly other techniques as well, could in principal be used for proactive mobility management as long as the given requirements are satisfied. Nonetheless, the architecture set forth herein narrows the solution space down to a particular set of mechanisms for the benefit of reasonably containing the complexity of the architecture itself and of the forthcoming protocol specification. These mechanisms comprise the IEEE 802.21 Media Independent Event and Command Services for handover anticipation, candidate access point discovery, as well as cross-layer interaction and synchronization[2], Optimistic Duplicate Address Detection for proactive care-of address generation, and the IEEE 802.21 Media Independent Information Service for off-link subnet prefix discovery. However, it is straightforward to adapt the architecture and protocol specification in this document so as to interoperate with alternative mechanisms.

Figure 5.3 depicts the network entities involved in the architecture for proactive mobility management in an example scenario. While reactive mobility management is solely based on standard IPv6 and Mobile IPv6 network entities, the architecture for proactive mobility management requires an additional IEEE 802.21 MIH point of service located in the mobile node's access network. The MIH point of service is the Media Independent Information Services peer that the mobile node contacts during off-link subnet prefix discovery. It may be colocated with the mobile node's current access point or access router, but in general it can be anywhere in the access network.

Figure 5.6 details the functional components on the mobile node and how they interact with each other as well as with standard IPv6 and IEEE 802.21 entities in the access network and Mobile IPv6 entities elsewhere in the Internet. An MIH function constitutes the mobile node's pivotal interface to all IEEE 802.21 services. One or multiple network interfaces, potentially for different access technologies, connect to the MIH function on the link layer side. The sole user of the MIH function at the

[2]Handover anticipation and candidate access point discovery belong to the Media Independent Event and Command Services at the time of this writing, but recent developments [59] within the IEEE 802.21 working group indicate a willingness to move both processes to the Media Independent Information Service.

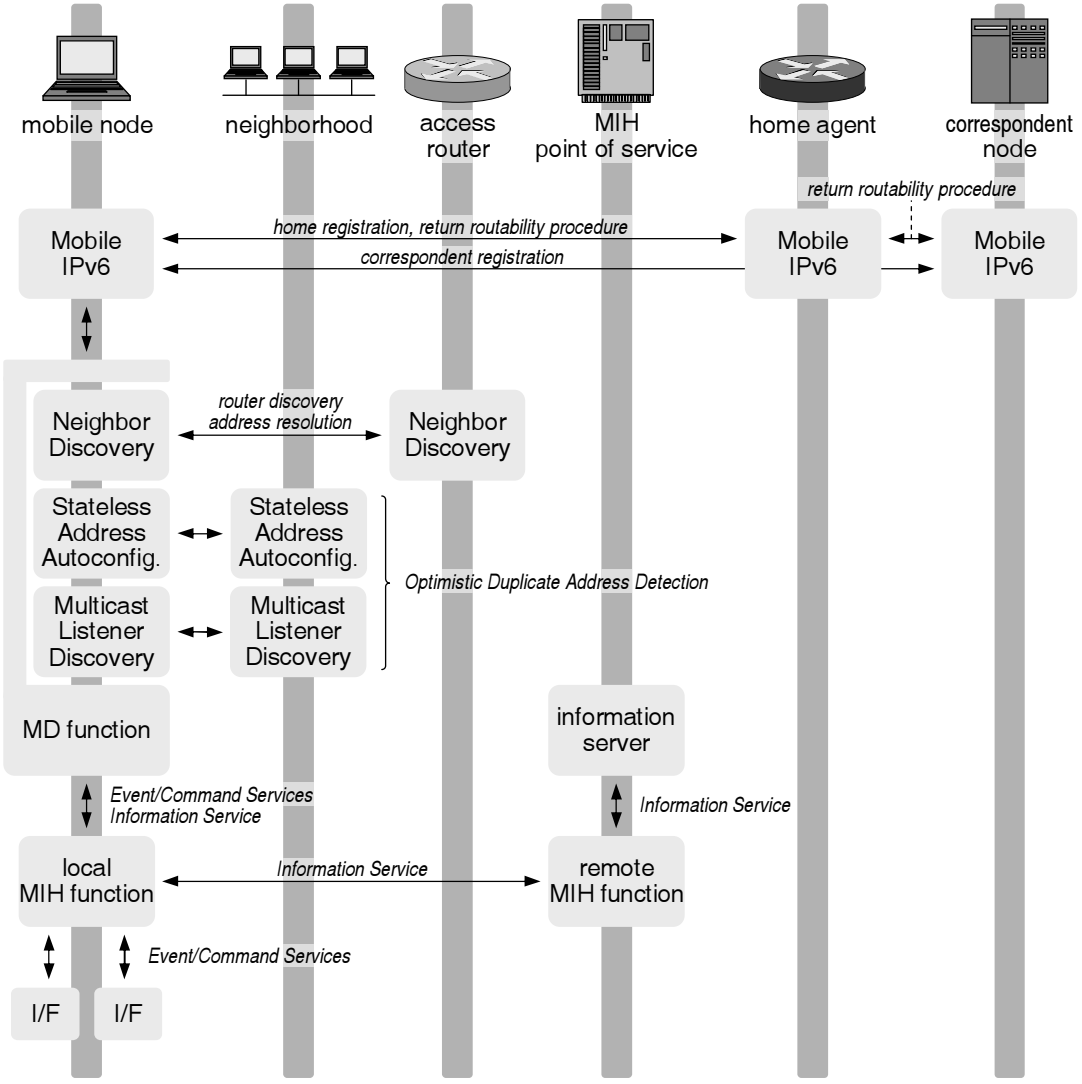



Figure 5.6: Network entities and architectural components involved in proactive mobility management

IP layer is the *movement detection function*, or *MD function*. The MD function conducts handover anticipation, candidate access point discovery, and subnet prefix discovery by help of the services of the MIH function, and it implements movement detection based on proactively retrieved subnet prefixes. The MD function further embodies the central control logic that coordinates the handover-related activities of the Neighbor Discovery, Stateless Address Autoconfiguration, and Multicast Listener Discovery protocols.

When a handover has been anticipated and a target access point has been selected, the MD function retrieves the subnet prefixes in use on the target access point's link through the local MIH function, which in turn fetches the subnet prefixes from a remote MIH function on the MIH point of service. The remote MIH function looks the requested subnet prefixes up in an information server. The information server may be realized as an additional process on the MIH point of service itself or as a separate entity elsewhere in the access network. For simplicity, it is colocated on the MIH point of service in figure 5.6. In case movement detection confirms

that the intended handover to the target access point brings about a change in IP connectivity, the MD function triggers the Stateless Address Autoconfiguration protocol to proactively generate one or multiple care-of addresses, and subsequently initiates Mobile IPv6 signaling through an interface to the Mobile IPv6 protocol. The MD function eventually initiates the actual link layer handover to the target access point through the Media Independent Command Service.

The MD function interfaces to the Neighbor Discovery protocol to advertise the mobile node's link layer address and initiate router discovery once the mobile node arrives on the new link. It also triggers Optimistic Duplicate Address Detection signaling in order to verify uniqueness of the new care-of addresses. There is no standardized protocol for the coordination between the MD function and the Stateless Address Autoconfiguration, Mobile IPv6, and Neighbor Discovery protocols. The coordination typically encompasses a set of proprietary messages or function calls specific to the network stack software.

The mobile node must generally discover a MIH point of service when it enters an access network for the first time, unless the information which MIH point of service to use is preconfigured into the mobile node. MIH point of service discovery may further be necessary after the mobile node has switched links within the same access network. Proactive mobility management requires the mobile node to contact the MIH point of service only when it is about to leave a particular access point, so the mobile node may discover the MIH point of service at any time while it resides at this access point. It would theoretically be sufficient for the mobile node to prompt the discovery in an ad-hoc manner after the MD function on the mobile node has arrived at the decision to move to a different access point. This approach would delay subnet prefix discovery, movement detection and, if needed, proactive care-of address generation and Mobile IPv6 signaling, however. As a consequence, the mobile node may at times lose connectivity to the serving access point prior to completing proactive handover preparations. To avoid such premature handovers, the mobile node should discover a MIH point of service well in advance, preferably already when it arrives at a target access point. The mobile node can then access the remote MIH function immediately when it again anticipates a handover to a different access point.

Media Independent Event and Command Services may also be used to realize network-controlled handovers. Handover anticipation and candidate access point discovery in the MD function on the mobile node is then replaced with a handover decision logic in the access network through which the MD function on the mobile node can be remote-controlled via IEEE 802.21 signaling with the mobile node's MIH function. The proactive mobility management developed in this document is limited to mobile-node-initiated handovers, however.

### 5.3.3 Protocol Operation

While reactive mobility management solely operates on the new link after a mobile node has moved, proactive mobility management operates from both the old and the new link. This naturally divides the proactive handover procedure in a preparatory *pre-handover phase*, which is initiated by and fully executed via the old link prior to changing the access point, and a follow-up *post-handover phase*, which begins once the mobile node arrives on the new link. The pre-handover phase includes

handover anticipation, the discovery of candidate access points and the election of one of those as a handover target, the discovery of the set of subnet prefixes in use at the target access point, movement detection and proactive IP address generation, as well as proactive Mobile IPv6 home and correspondent registrations. Uniqueness verification for the new IP addresses requires signaling local to the new link and is hence deferred until after the handover. The post-handover phase includes Optimistic Duplicate Address Detection signaling for the new care-of address as well as reactive Mobile IPv6 signaling for any home or correspondent registrations that could not complete during the pre-handover phase.

An illustration of the entire proactive handover procedure is, for the purpose of clarity, split across two figures. Figure 5.7 shows the IEEE 802.21 messages exchanged either within the mobile node—between the link layer, the local MIH function, and the MD function—, as well as between the local MIH function on the mobile node and a remote MIH function on an MIH point of service in the access network. The figure does not explicitly show the information server with which the MIH function on the MIH point of service interacts. The information server may run as a local process on the MIH point of service, or it may be realized as a separate network entity. Figure 5.8 depicts Mobile IPv6 and other IP layer signaling. It also shows the implementation-specific signaling messages between the MD function and Mobile IPv6 internal to the mobile node. The circled numbers in both figures denote important steps in the handover procedure and are referenced in the text that follows. They increase chronologically and cover all handover-related activities, which is why the numbers representing activities not shown in either of the figure are skipped. It is assumed that the mobile node discovers the MIH point of service well before the actual handover. The discovery is hence not shown in the figures. It is further assumed that the MD function has preconfigured the local MIH function with a set of threshold parameters appropriate for the link layer technologies in use and the expected movement patterns of the mobile node. The configuration is therefore not shown in the figures as well.

### Handover Anticipation

The pre-handover phase begins when the mobile node's MIH function decides to hand over to a different access point (1), be it because the signal strength at the current access point has dropped below the preconfigured threshold, or because an access point with higher available bandwidth, lower access latency, lower connection cost, or a preferred access provider has been detected. The handover reason in figure 5.7 is decreasing signal quality at the current access point, announced by a Link Going Down Indication message which the link layer sends to the local MIH function. The MIH function forwards the information as an MIH Link Going Down Indication message to the MD function.

### Candidate Access Point Discovery

The MD function initiates candidate access point discovery when it receives the MIH Link Going Down Indication message and sends an MIH Get Status Request message to the local MIH function (2). The local MIH function translates the MIH Get Status Request message into a sequence of Get Status Request messages for the different network interfaces, aggregates the responses from received Get Status Confirm messages, and relays the result back to the MD function within an MIH

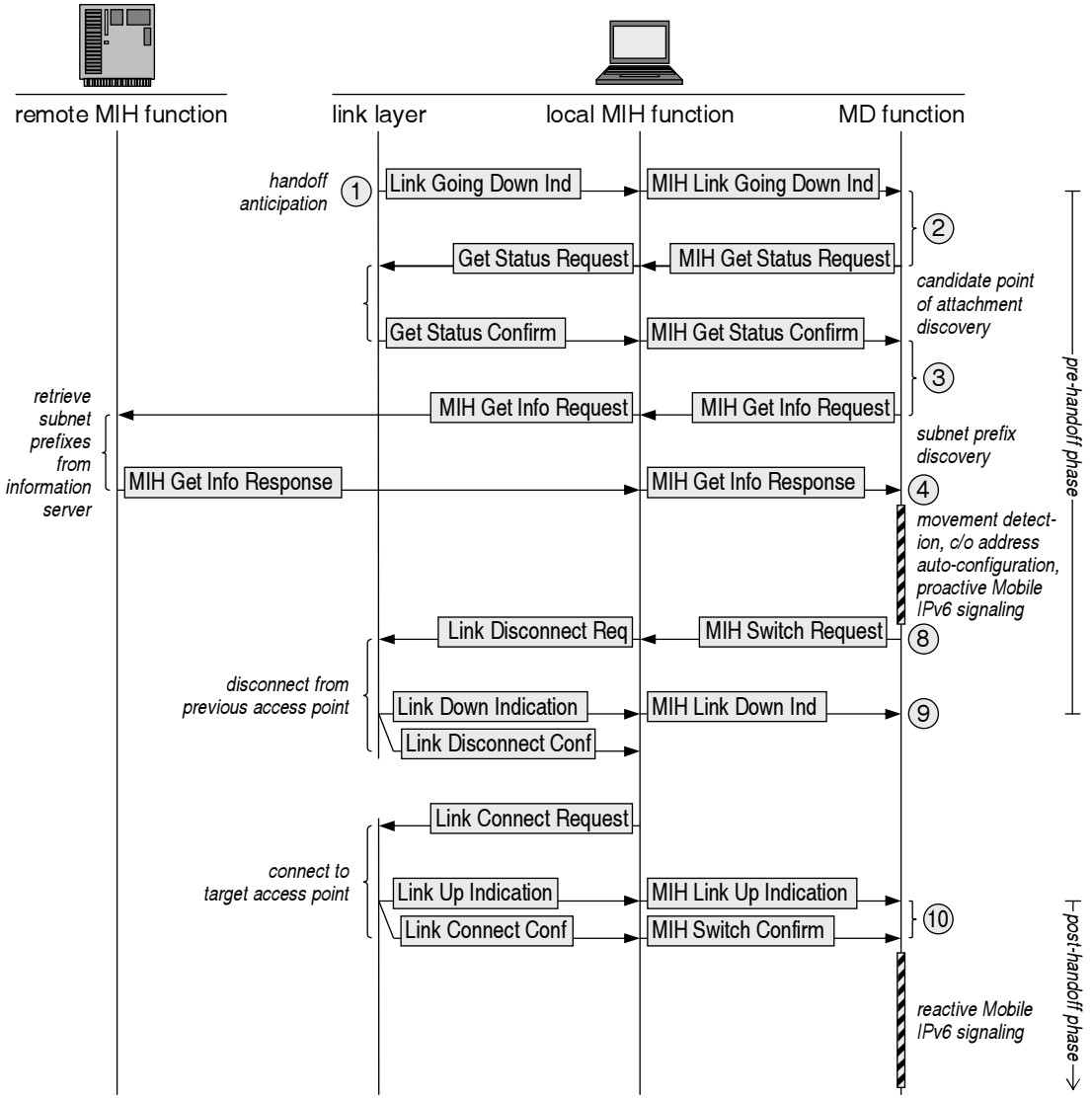



Figure 5.7: IEEE 802.21 signaling during proactive mobility management

Get Status Confirm message. The status information retrieved per candidate access point includes the current signal quality, the link speed, and the operational mode of the corresponding network interface, which could be active, shut-down, or in power saving mode. The MD function selects one of the discovered candidate access points as a handover target.

### Proactive Subnet Prefix Discovery

Once the target access point has been determined, the MD function proceeds to discover the set of subnet prefixes in use on the link to which the target access point connects (3). It first checks to see whether the subnet prefix information has been preconfigured for the link layer address of the target access point, or whether such a mapping has been cached from a previous handover. If neither is the case, the MD function acquires the subnet prefixes by means of the Media Independent Information Service as shown in figure 5.8. It sends an MIH Get Information Request message including a Subnet Information information element to the local MIH function, which forwards the message to the remote MIH function on the MIH point of service. The

remote MIH function retrieves the desired prefix information from the information server and sends the results back to the MIH function on the mobile node within an MIH Get Information Response message. The MIH function forwards the message to the MD function.

### Movement Detection

The MD function subsequently performs movement detection on the retrieved subnet prefixes (4). Proactive movement detection is based on the same algorithms that the DNA protocol defines for reactive movement detection: A handover between access points involves a change in IP connectivity if and only if none of the prefixes that were in use at the old access point are still valid at the target access point. However, to prevent the mobile node from erroneously considering a node with an IP address from one of the proactively obtained subnet prefixes to be a neighbor on the current, old link, the new prefixes are not added to the prefix list before the mobile node has actually attached to the target access point.

If the MD function finds that IP connectivity will not change during the handover to the target access point, it directly initiates the link layer handover (5), skipping any IP layer handover activities. The mobile node can then continue to communicate via its current care-of address without Mobile IPv6 home and correspondent registrations. Some of the retrieved prefixes may still be unknown to the mobile node, but it is sufficient to configure new IP addresses from those in reactive manner after the handover, according to the standard Stateless Address Autoconfiguration and Optimistic Duplicate Address Detection protocols. On the other hand, proactive care-of address generation and Mobile IPv6 home and correspondent registrations become necessary in case IP connectivity will change during the handover to the target access point.

### Proactive Care-of Address Generation

If movement detection determines that IP connectivity will change during the handover to the target access point, the MD function selects one or multiple of the new subnet prefixes and generates care-of addresses for each of those. The new care-of addresses are put into Optimistic state as they would with the standard, reactive Stateless Address Autoconfiguration and Optimistic Duplicate Address Detection protocols. However, the new care-of addresses are not configured on a network interface until the mobile node has actually switched over to the target access point. This prevents the mobile node from sending from the old link packets with an IP source address that topologically belongs to the new link. Optimistic Duplicate Address Detection permits a care-of address to be generated on the old link and proactively registered with the home agent and any correspondent nodes, while allowing any signaling necessary to verify the uniqueness of the care-of address to be deferred until the mobile node has moved to the new link. The MD function then sends one Register message per new care-of address to Mobile IPv6 in order to trigger the necessary home and correspondent registrations. Each Register message contains a new care-of address and the respective old care-of address that is to be replaced by the new one.

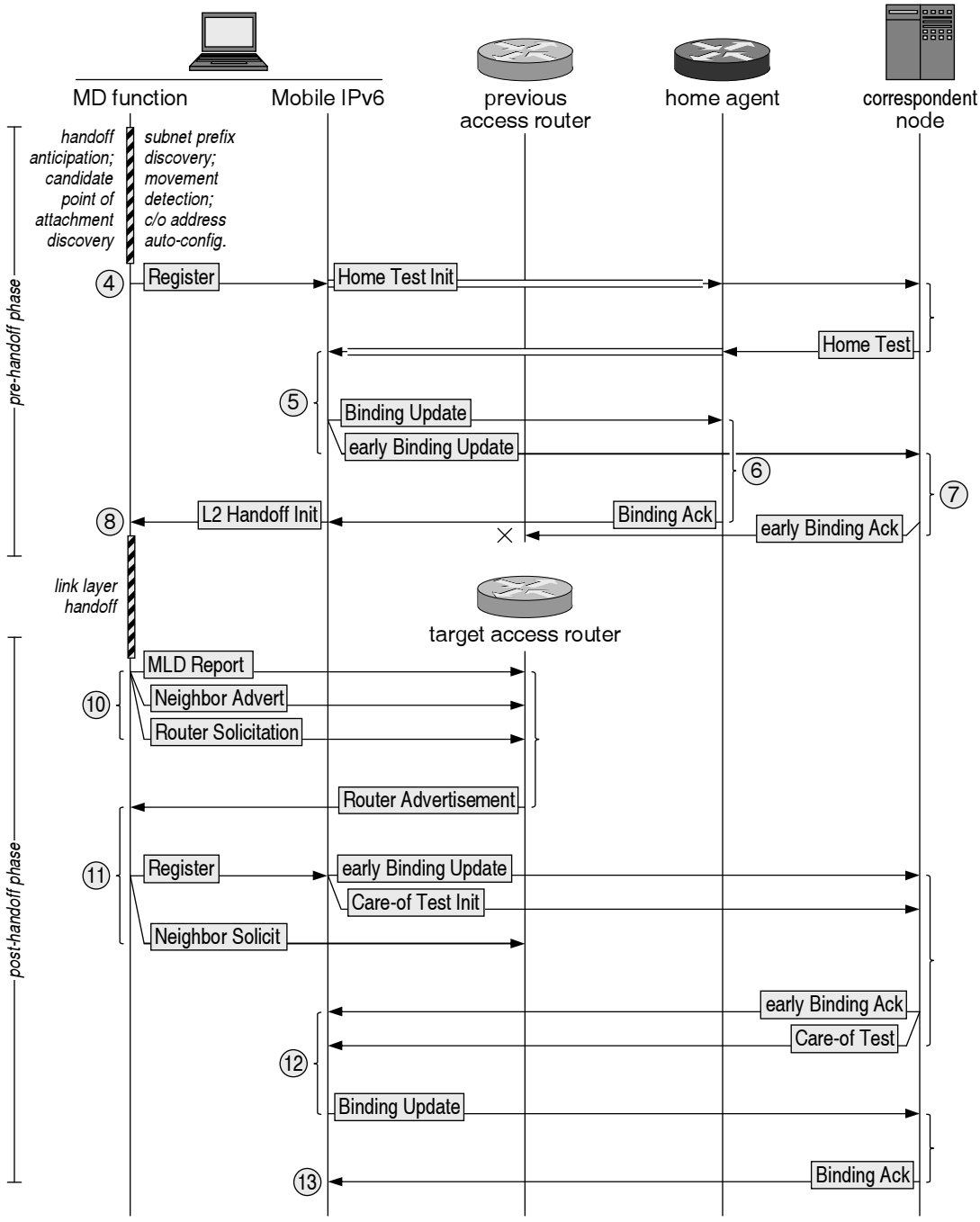



Figure 5.8: IP connectivity change signaling in proactive mobility management

### Proactive Mobile IPv6 Registrations

Mobile IPv6 conducts home and correspondent registrations as needed when it receives the Register messages from the MD function. Specifically, a binding at the home agent or a correspondent node must be updated if and only if one of the received Register messages replaces the care-of address in use for this binding. Figure 5.8 depicts the standard case where the same care-of address is used for the home agent and a single correspondent node. Mobile IPv6 first initiates a proactive home address test for the correspondent registration by sending a Home Test Init message.

The correspondent node generates a new home keygen token when it receives this message, and sends the token back to the mobile node within a Home Test message.

The home keygen token allows the mobile node to authenticate an early Binding Update message for the correspondent node. Mobile IPv6 then sends a Binding Update message to the home agent and an early Binding Update message to the correspondent node (5). These messages are still sent from the old link and hence from the mobile node's old care-of address. A correspondent node consequently cannot read the new care-of address from the Source Address field of the IP header, as it usually does, so Alternate Care-of Address mobility options [63] are added to the Binding Update messages to hold the new care-of address. An interest in an acknowledgment is indicated by setting the Acknowledge flag in both messages. The mobile node continues to send payload packets from its old care-of address until it actually moves to the new link.

The home agent registers the new care-of address when it receives the Binding Update message (6), but temporarily continues to also accept payload packets that the mobile node may send from the old care-of address before moving to the new link [104]. Likewise, the correspondent node registers the new care-of address upon receipt of the early Binding Update message (7), but continues to also accept payload packets from the old care-of address for a while.

Standard Mobile IPv6 rules cause the home agent and the correspondent node to acknowledge Binding Update and early Binding Update messages to the old care-of address, and to direct subsequent payload packets to the new care-of address. One of these acknowledgments will cause the mobile node to instruct its link layer to switch to the target access point. The mobile node implements its own policy as to which acknowledgment it uses for this purpose. Possible solutions are discussed in section 5.3.4. In figure 5.8, the mobile node initiates the link layer handover upon reception of the home agent's Binding Acknowledgment message (8). The early Binding Acknowledgment message from the correspondent node arrives at the old link a bit later and is therefore lost.

**Link Layer Handover**

Mobile IPv6 sends an L2 Handover Init message to the MD function when the Binding Acknowledgment message that determines the time of handover is received and thus turns control back over to the MD function. The MD function then sends an MIH Switch Request message to the local MIH function. The request contains the link layer address of the target access point and specifies whether the link layer should associate with the target access point after or before it disassociates from the old access point, resulting in a break-before-make handover or a make-before-break handover, respectively. The MIH function translates the MIH Switch Request message into a sequence of commands specific to the involved network interfaces: If the desired handover mode is break-before-make, a Link Disconnect Request message for the old access point is sent first, and a Link Connect Request message for the target access point is sent second. This is the order shown in figure 5.7. If the handover is make-before-break, the sequence of messages would be reverse. The link layer responds to the requests by sending Link Disconnect Confirm and Link Connect Confirm messages, respectively. The link layer also generates Link Down Indication and Link Up Indication messages, which the MIH function forwards to the

MD function as MIH Link Down Indication and MIH Link Up Indication messages, respectively.

The pre-handover phase ends when the MIH Link Down Indication message for the old access point is received by the MD function (9), and the post-handover phase begins when the MD function receives the MIH Link Up Indication message showing that the link layer has reassociated with the target access point (10). There is usually a small pause between the pre- and post-handover phases if the handover mode is break-before-make, due to the time it takes the link layer to tune into the frequency of the target access point, authenticate, and associate. If the handover mode is make-before-break, the two handover phases typically overlap shortly.

**Link Layer Address Announcement**

Incoming payload packets may already be queued at the new access router when the mobile node arrives on the new link, or they arrive shortly, as the home agent and the correspondent node should already be using the new care-of addresses. For the new access router to deliver these packets to the mobile node, it must resolve the mobile node's link layer address. The MD function accelerates the delivery in that it transmits an unsolicited Neighbor Advertisement message (10) when the reception of the MIH Link Up Indication message from the local MIH function tells that the mobile node has arrived on the new link. The unsolicited advertisement inserts the mobile node's link layer address directly into the neighbor caches of the new access router. This is an optimization that conforms to the standard Neighbor Discovery protocol.

The access router would in the absence of the unsolicited Neighbor Advertisement message transmit a Neighbor Solicitation message and thereby trigger the mobile node to respond with a solicited Neighbor Advertisement message. This behavior could be a source of increased handover delay and packet loss, however: The access router sends a Neighbor Solicitation message for the mobile node's new care-of address already when it receives the first payload packet for the mobile node. It may well be that the mobile node is still on the old link, or in the process of handing over to the new link, when this message is sent, so the mobile node necessarily misses the message. Rate limitations require the correspondent node to retransmit the Neighbor Solicitation message only every 1 s, even if payload packets for the mobile node arrive much faster. Consequently, if the first or first few solicitations were lost, the delivery of payload packets queued at the access router would be delayed for up to 1 s. Part of the payload packets may furthermore get lost because the buffer space that access routers reserve for packets awaiting link layer address resolution are limited in size and typically do not hold more than three packets.[3] While most operating systems provide an application programming interface through which the buffer size can be increased, large buffers can potentially be exploited for denial-of-service attacks, so it is generally wise for security reasons to keep them small. Link layer address announcement through an unsolicited Neighbor Advertisement message hence forms an important part of proactive mobility management.

The unsolicited Neighbor Advertisement message also has the purpose of advertising the mobile node's link layer IP address to any correspondent nodes that happen to

[3]The Linux operating system defaults to three packets, while the FreeBSD operating system goes with the minimum of one packet, which the IPv6 Neighbor Discovery RFC stipulates.

connect to the same link to which the mobile node has handed over. Such correspondent nodes do not relay payload packets for the mobile node to the access router, but attempt to transmit them directly to the mobile node. The correspondent nodes thus perform link layer address resolution like the mobile node's new access router, and they must comply to the same rate limitations for sending Neighbor Solicitation messages. The unsolicited Neighbor Advertisement message avoids increased handover delay and packet loss, which may result from the rate limitations, in that it inserts the mobile node's link layer address into the correspondent nodes' neighbor caches directly.

Since the mobile node sends the unsolicited Neighbor Advertisement message for a care-of address in Optimistic state, the Override flag in the message must be cleared. This does not prevent the access router and any correspondent nodes to accept the link layer address specified in the message, however.[4] They typically create a new neighbor cache entry in Incomplete state for the mobile node's care-of address when they begin link layer address resolution. The Override flag in unsolicited Neighbor Advertisement messages is ignored if it pertains to such a neighbor cache entry.

The unsolicited Neighbor Advertisement message may be retransmitted up to 2 times for increased reliability in the presence of packet loss. However, the minimum time between successive advertisements is defined by the RetransTimer variable that access routers announce in their Router Advertisement messages. The mobile node does not know the value of the RetransTimer variable effective on the new link until it has received at least one Router Advertisement message. It therefore must defer setting the timer for any retransmission of the unsolicited Neighbor Advertisement message until it has performed router discovery. Figure 5.8 shows only a single transmission of the advertisement for simplicity. Unsolicited Neighbor Advertisement messages are sent to the all-nodes multicast address. The mobile node hence does not need to know the new access router's IP or link layer address at the time it sends the messages, which enables it to send the first advertisement prior to router discovery.

### Router Discovery

The Media Independent Information Service enables a mobile node to proactively retrieve the set of subnet prefixes in use on the link to which an elected target access point connects, but the current IEEE 802.21 specification does not include a mechanism with which the mobile node can retrieve the IP or link layer addresses of an access router on the new link.[5] The MD function therefore initiates reactive router discovery by sending a Router Solicitation message once the mobile node's link layer address has been announced through the unsolicited Neighbor Advertisement message.

[4] The access router and the correspondent nodes would reject the unsolicited Neighbor Advertisement message only if they already have a neighbor cache entry for the same care-of address and a different link layer address. This can happen only if the mobile node's care-of address is a duplicate, in which case the mobile must anyway transition to a different care-of address as described next.

[5] The inability to learn about new access routers available at the target access point calls for an amendment to the current IEEE 802.21 specification. Mobile nodes are at present required to perform router discovery on the new link in reactive manner in order to update their default router list and send packets to the Internet. This implies additional handover delays.

On-link access routers that implement the DNA protocol calculate a delay after which they send a Router Advertisement message based on their own link-local IP address and the link-local IP addresses of the mobile node and other access routers. This behavior desynchronizes the transmissions of Router Advertisement messages from multiple routers and thus prevents collisions. The DNA protocol enables one of the routers to send its advertisement immediately; additional routers send their advertisements each incrementally displaced by 20 ms. The MD function initiates reactive Mobile IPv6 registrations by sending another set of Register messages to Mobile IPv6 after the first Router Advertisement message has been received. Each Register message includes one new care-of address along with the old care-of address that it replaces. The 20-ms interval between the first and the second advertisement should be sufficient for the initial Mobile IPv6 messages to be transmitted without getting exposed to a collision with subsequent Router Advertisement messages.

#### Reactive Mobile IPv6 Registrations

Some of the Binding Acknowledgment messages from the home agent and correspondent nodes are usually lost on the old link, so Mobile IPv6 cannot tell whether all registrations were successful. It hence retransmits standard and early Binding Update messages for any unacknowledged registrations when it receives the second set of Register messages from the MD function (11). In figure 5.8, the registration with the only correspondent node remains unacknowledged on the old link, so the mobile node resends the early Binding Update message from the new link. Reactive Mobile IPv6 registrations executed during the post-handover phase follow the same procedure as in reactive mobility management. They cannot occur until router discovery has been completed on the new link because the mobile node must know the link layer and IP addresses of an access router to which it can relay the Binding Update messages.

The mobile node may send payload packets from the new care-of addresses as soon as all standard and early Binding Update messages have been transmitted. The mobile node then also initiates a concurrent care-of address test for each correspondent node by sending a Care-of Test Init message. After a Care-of Test message with a fresh care-of keygen token has been received (12), the mobile node sends a standard Binding Update message to the respective correspondent node. It finally receives a standard Binding Acknowledgment message (13), provided that the Acknowledge flag in the Binding Update message was set.

#### Optimistic Duplicate Address Detection

The MD function initiates Stateless Address Autoconfiguration and Optimistic Duplicate Address Detection signaling after the standard and early Binding Update messages have been sent (11). It first transmits one or multiple MLD Report messages in order to subscribe to the solicited node multicast addresses corresponding to the new care-of addresses. A single MLD Report message is sufficient if the care-of addresses deviate only in the subnet prefix—which they likely do—because the solicited node multicast address is the same for all IP addresses that differ only in the upper 104 bits. The MD function then initiates the transmissions of one Neighbor Solicitation messages per new care-of address in order to verify the uniqueness of the care-of addresses according to the rules of Optimistic Duplicate Address Detection.

A care-of address is transferred from Optimistic state to Preferred state when no defending Neighbor Advertisement message has been received within a period of 1 s.

In the unlikely case that a proactively generated care-of address turns out to be already in use by a node on the new link, the mobile node receives a Neighbor Advertisement message from the true owner of the care-of address in response to one of the Neighbor Solicitation messages. The mobile node must then invalidate any previous home and correspondent registrations for which the duplicate care-of address was used. It generates a new care-of address and registers anew. All follow-up registrations proceed as has been specified for reactive mobility management. The mobile node subsequently invokes Optimistic Duplicate Address Detection signaling to verify uniqueness of the new care-of address. The post-handover phase ends when all pending home and correspondent registrations are complete and all care-of addresses have been transferred from Optimistic state to Preferred state.

### 5.3.4 Appointing Link Layer Handover Initiation

A mobile node that prepares a change in IP connectivity through the proactive transmission of standard and early Binding Update messages considers one of the returning Binding Acknowledgment messages as a trigger to initiate the link layer handover to the target access point. Which acknowledgment to select is up to the policy of the mobile node. The actual time of the link layer handover is therefore not uniquely determined when the mobile node updates multiple bindings and expects to receive multiple acknowledgments. The acknowledgments are likely to arrive at the mobile node at different times because the round-trip delays to the home agent and to the correspondent nodes usually differ. There is hence a span of time within which the link layer handover may be initiated. Even if the mobile node communicates with only a single correspondent node, it must still decide between the home agent's and the correspondent node's Binding Acknowledgment message upon which it initiates the link layer handover. The time for link layer handover initiation is well-defined only when the mobile node communicates exclusively by means of bidirectional tunneling through the home agent.

Yet the decision of when to initiate the link layer handover has a strong impact on handover delay and packet loss. If the mobile node moves too early, it may lose payload packets at the old care-of address and end up waiting for payload packets to arrive at the new care-of address. If the mobile node switches too late, it may have to communicate at high power levels via the old access point due to fading signal strength, while packets arriving in its absence at the new care-of address are bound to be lost. Figure 5.9 exemplifies this problem with a scenario where the mobile node registers with its home agent and two correspondent nodes, $A$ and $B$. The round-trip delay to correspondent node $A$ is 50 ms, and the round-trip delay to correspondent node $B$ is 100 ms. The 150-ms round-trip delay to the home agent is highest, as it may be the case when the mobile node roams away from home. The mobile node sends the Binding Update messages for all three peers at the same time. The respective Binding Acknowledgment messages arrive at the old care-of address after 50 ms, 100 ms, and 150 ms. The mobile node may initiate the link layer handover upon the reception of any of those.

An analysis of the IP layer handover delay and packet loss is simplified by the assumption that the link layer handover delay approaches zero, and that the round-trip

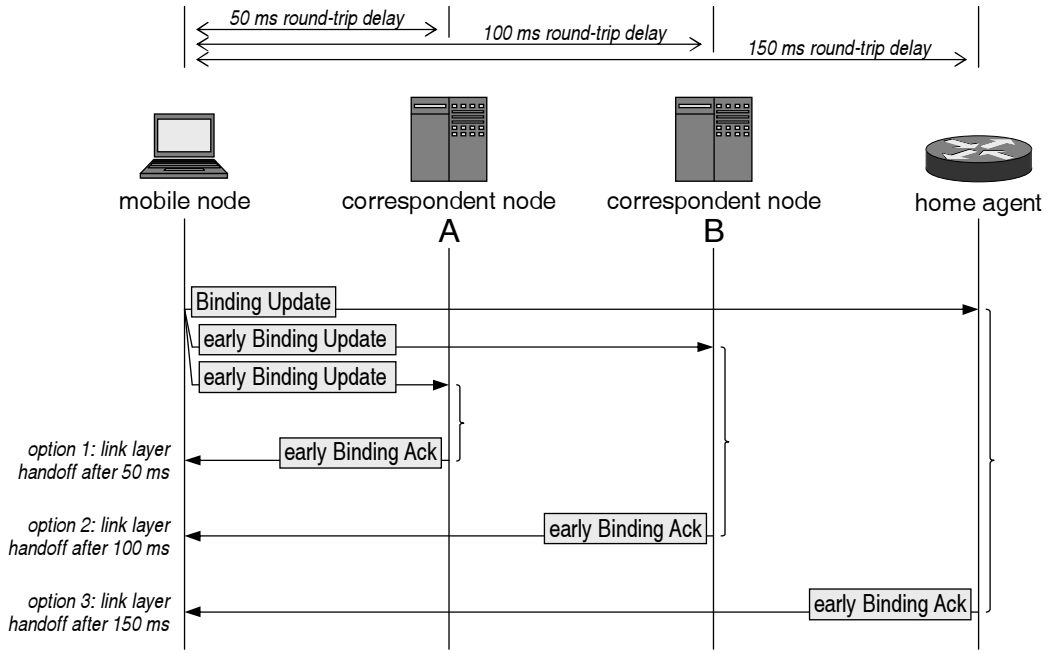



Figure 5.9: Possibilities of when to initiate the link layer handover

delays from the mobile node's old care-of address to correspondent nodes $A$ and $B$ equal the respective round-trip delays from the new care-of address to correspondent nodes $A$ and $B$. Then, if the mobile node decides to initiate the link layer handover upon receipt of correspondent node $A$'s Binding Acknowledgment message, it will lose 50 ms worth of payload packets from correspondent node $B$ at the old care-of address, and even 100 ms worth of payload packets that the home agent continues forwarding to the old care-of address. If the mobile node defers the link layer handover until it receives correspondent node $B$'s Binding Acknowledgment message, it is likely to miss 50 ms worth of packets that correspondent node $A$ sends to the new care-of address in the meantime. The new access router may be able to hold a limited number of these packets in its link layer address resolution buffer. But such buffers are usually small and hence unlikely to provide a feasible cushion (see also section 5.3.3).

In general, the IP layer handover delay $L_{IP}$ for a communication session with a particular correspondent node is a function of the link layer handover delay $L_{link}$, the time $D_{link}$ between the reception of the correspondent node's Binding Acknowledgment message at the old care-of address and the link layer handover initiation, as well as the difference between the round-trip delay $R_{new}$ from the new care-of address to the correspondent node and the round-trip delay $R_{old}$ from the old care-of address to the correspondent node. This function given by the following equation:

$$L_{IP} \quad = \quad D_{link} + L_{link} - (R_{new} - R_{old})$$

Figure 5.10 illustrates the impact that each of these four parameters has on the IP layer handover delay.

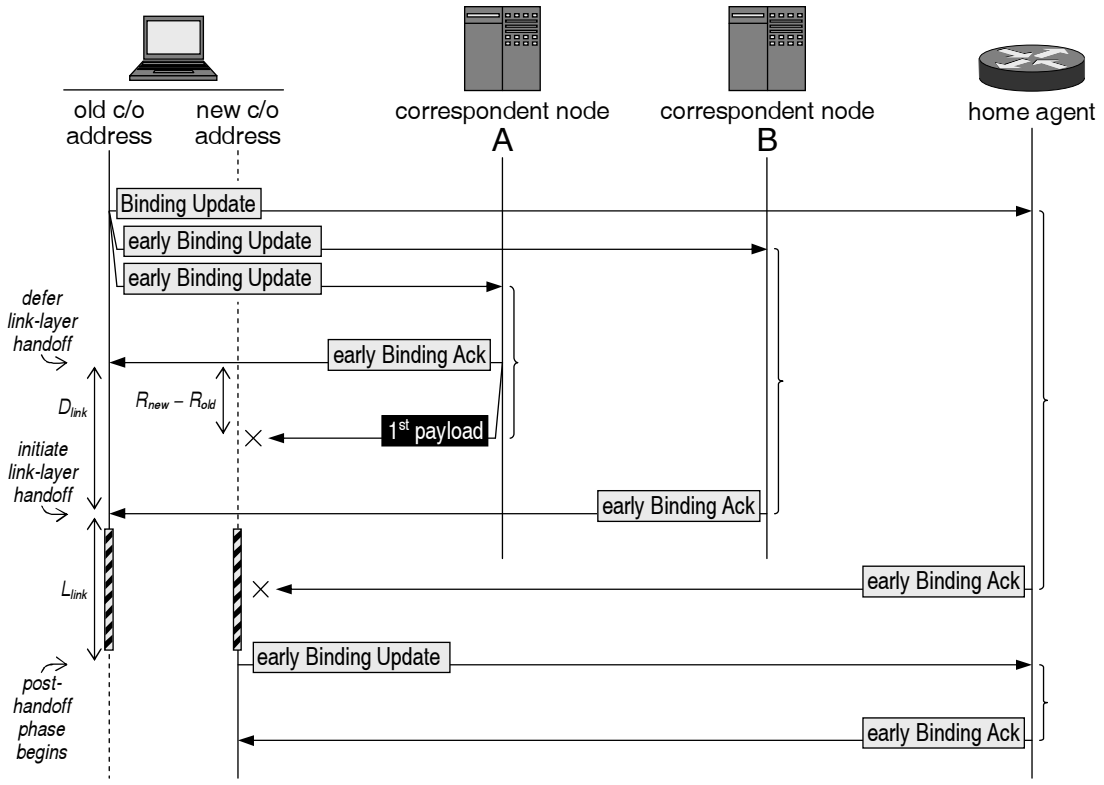



Figure 5.10: Impact of link layer handover initiation time

There are multiple approaches to this problem. A simple heuristic would be to initiate the link layer handover when the home registration is acknowledged. The Mobile IPv6 RFC prioritizes home registrations higher than correspondent registration, so this strategy falls in line with the base specification. Earlier reception of an early Binding Acknowledgment message from a correspondent node would be registered as such to avoid an unnecessary retransmission of the respective early Binding Update message on the new link, but further communications with that correspondent node would be deferred until the mobile node arrives on the new link. This technique is adequate when the mobile node does not frequently use route optimization, or when prioritized communications are tunneled via the home agent.

On the other hand, prioritizing home registrations may be a source of decreased handover performance when route optimization is in use. Given that a typical use case for route optimization is real-time applications, the heuristic is in fact likely to penalize highly delay-sensitive communications. Circumstances are worst in the Two Japanese in America scenario. Better handover performance can then be obtained if the link layer handover is tied to the arrival of an early Binding Acknowledgment message, provided that the mobile node uses route optimization with at least one correspondent node. A rudimentary implementation might prompt the link layer handover when the first early Binding Acknowledgment message arrives; a more sophisticated algorithm could choose an acknowledgment based on real-time properties of active applications.

A third strategy launches the link layer handover upon the first acknowledgment received, whether it originates with the home agent or with a correspondent node.

This approach seeks to combine the advantages of the two previous techniques: When no route optimization is used, the home agent's Binding Acknowledgment message is the only one, and the time for link layer handover is chosen well. The result is similar if the home agent is closer to the mobile node compared to the correspondent nodes, as it is likely to be the case in environments where home agents are dynamically assigned based on the logical position of the mobile node [62]. However, when a correspondent node's early Binding Acknowledgment message is received first, a possibly long wait for the home agent's acknowledgment is aborted in favor of route-optimized communications.

Dynamic techniques where the Binding Acknowledgment message that determines the time of link layer handover is selected based on application characteristics are certainly superior to algorithms that handle acknowledgments in the order they are received. Humans are limited in the number of concurrent activities they can handle or perceive. This may justify the assumption that at most one, possibly two real-time applications take place at a time. If insight in the nature of active applications is available, the mobile node could identify the peer for which to schedule link layer handover would maximize user satisfaction.

Similarly, a handover decision could be drawn from the ratio between the number of communication sessions for which the mobile node uses route optimization and the number of communication sessions for which it does not. If most data flows are routed via the home agent—be it because the bigger part of all correspondent nodes does not support route optimization or for other reasons—, it may be wise to give precedence to the home agent's Binding Acknowledgment message. On the other hand, an acknowledgment from a correspondent node would trigger the link layer handover when most communication sessions use route optimization. Mobile IPv6 implementations keep anyway state about correspondent nodes that do not support route optimization so as to reasonably limit attempts to establish a binding. This information could aid in calculating the ratio between route-optimized and bidirectionally tunneled communication sessions

The protocol specification in section 5.3.3 does not define a particular policy engine for a mobile node to use for scheduling link layer handovers in proactive mobility management. Which strategy is optimum depends on the number of correspondent nodes communicating with the mobile node, the topological properties of the involved routing paths, and the requirements of the applications in use. The performance evaluation in chapter 6 is based on three different strategies which reasonably represent the choices implementors have: Initiating the link layer handover upon receipt of the home agent's Binding Acknowledgment message, the first early Binding Acknowledgment message from a correspondent node, or the first acknowledgment in general, independent of whether it originates from the home agent or from a correspondent node. These modes of operation are called *Home Priority mode*, *Correspondent Priority mode*, and *Impatient mode*.

## 5.4 Summary

This section has demonstrated the integrability of concurrent reachability verification and Credit-Based Authorization into Mobile IPv6 route optimization. Mobile IPv6 belongs to those mobility protocols that do not inform a correspondent node

about a mobile node's new IP address before the mobile node's reachability at that IP address is proven. So for reachability verification to take place concurrently, the correspondent node must be enabled to learn a new IP address early on after a handover. This has been achieved with a set of standard-compliant optimizations and a set of optimizations that require changes to the Mobile IPv6 protocol specification. The implementation of all changes together yields the highest benefits for reactive mobility management, and it also facilitates proactive mobility management. The value of the standard-compliant optimizations alone is that they can be used as a guideline for implementors to produce Mobile IPv6 software for efficient, reactive mobility management, which still complies with today's standard.

As important as the Mobile IPv6 signaling itself is its interoperation with other handover-related activities on the IP layer. This interoperation was hence specified along with the modified Mobile IPv6 signaling. Existing optimizations of the protocols for router and prefix discovery, movement detection, and IP address autoconfiguration were exploited as much as possible for this. Moreover, to avoid needless signaling, it is important for a mobile node to discover a correspondent node's Early Binding Updates capability. A method for Early Binding Updates to be deactivated on the mobile-node side when a correspondent node turns out to not support it was hence devised.

Proactive mobility management requires a mobile node to anticipate and prepare a handover to a particular new link while still residing on its old link—two tasks that common network protocol stacks do not support. In this section, Mobile IPv6 was coupled with Media-Independent Handover Services to achieve these tasks. Another characteristic of proactive mobility management is that its performance depends strongly on the appointment of the link layer handover, yet the optimal link layer handover time is peculiar to the scenario. Three Mobile IPv6 modes for proactive mobility management were hence devised, to be applied selectively by a mobile node depending on current needs.

# 6. Evaluation

Credit-Based Authorization has been designed to protect against amplified redirection-based flooding attacks and sustained redirection-based reflection attacks. Since this protection is based on a limitation of the data that can be sent to a mobile node early on after a handover, the performance benefits that concurrent reachability verification and Credit-Based Authorization have on the communication sessions of legitimately behaving nodes remains to be evaluated. For this purpose, Early Binding Updates and Credit-Based Authorization have been implemented [17, 65, 113, 64] based on the Kame-Shisa Mobile IPv6 software, and an experimental testbed was built to measure the performance of Mobile IPv6 with Early Binding Updates relative to that of conservative and optimistic Mobile IPv6. This chapter demonstrates and evaluates measurements that have been obtained for UDP-based Internet telephony sessions as well as for TCP-based file transfers.

## 6.1 Testbed Setup

The objective for the setup and configuration of the experimental testbed was to reflect the properties of real deployment scenarios as faithfully as possible. This involved conceptual decisions while constructing the testbed as well as the selection of a number of configuration variables. Below is a description of the testbed; experimentation parameters are explained as far as they are relevant to the ensuing performance evaluation.

### 6.1.1 Mobile IPv6 Implementation

All testbed nodes run the FreeBSD operating system version 5.4. Mobile IPv6 functionality for this release is available separately with *Kame-Shisa*, a two-part implementation of RFC 3775 including a kernel patch for performance-critical packet processing as well as userspace daemons for control and signaling. The userspace daemon for the mobile node comprises two state machines. One state machine handles return routability signaling for all binding update list entries, the other is responsible for pending home or correspondent registrations. The userspace daemons for the home agent and the correspondent node function without state machines.

Mobile nodes can be configured with respect to whether they request a Binding Acknowledgment message for a correspondent registration. If an acknowledgment is requested, the mobile node waits for it before it starts route-optimizing payload packets from the care-of address being registered.

The original Kame-Shisa software provides only conservative Mobile IPv6, so the software has been modified to also support optimistic Mobile IPv6 as well as Early Binding Updates and Credit-Based Authorization. Early Binding Updates can be used for reactive and proactive mobility management, where proactive mobility management can be operated in Impatient mode, Home Priority mode, or Correspondent Priority mode. Mobile nodes running optimistic Mobile IPv6 can again be configured with respect to whether they request a Binding Acknowledgment message while registering a new care-of address with a correspondent node, in which case they wait for the acknowledgment before route-optimizing any payload packets from the new care-of address. Standard and early Binding Acknowledgment messages can also be requested in Mobile IPv6 with Early Binding Updates, yet the mobile node in this case begins route optimization already when it has sent the early Binding Update message. Binding Acknowledgment messages are always requested during proactive mobility management where they are used to trigger the link layer handover on the mobile-node side. Performance-wise, the way in which Binding Acknowledgment messages are handled is relevant only for conservative and optimistic Mobile IPv6. Where these protocol variants are represented in measurement diagrams, a suffix "+Ack" attached to the protocol name in the diagram legend is used to indicate that the mobile node requests and waits for a Binding Acknowledgment message during correspondent registrations.

The implementation of Credit-Based Authorization supports Inbound mode and Outbound mode, and Outbound mode may or may not be combined with care-of address spot checks. Credit aging uses the default parameters proposed in sections 4.2 and 4.3.

### 6.1.2 Topology

The experimental testbed consists of five nodes taking the roles of the mobile node, the correspondent node, and three access routers, as illustrated in figure 6.1. One of the access routers—the one shaded black in the figure—serves as the mobile node's home agent; its local link constitutes the mobile node's home link. The two gray-shaded access routers attach to foreign access links which the mobile node may visit when away from home. The exterior interfaces of the three access routers and the correspondent node connect to the "Internet".

#### Routing paths

Physically, the testbed nodes are connected by two Ethernet cables, one connecting the correspondent node and the external interfaces of the three access routers, and another one connecting the mobile node and the three access routers' internal interfaces. The mobile node's connectivity to either access router can be selectively en- and disabled through FreeBSD's IPFW2 link layer filter. The properties of global routes across the Internet are reproduced by FreeBSD's DummyNet bandwidth manager and delay emulator. This limits bandwidth to 1024 kbps and imitates end-to-end round-trip times of between 25 ms and 400 ms, depending on the experiment.

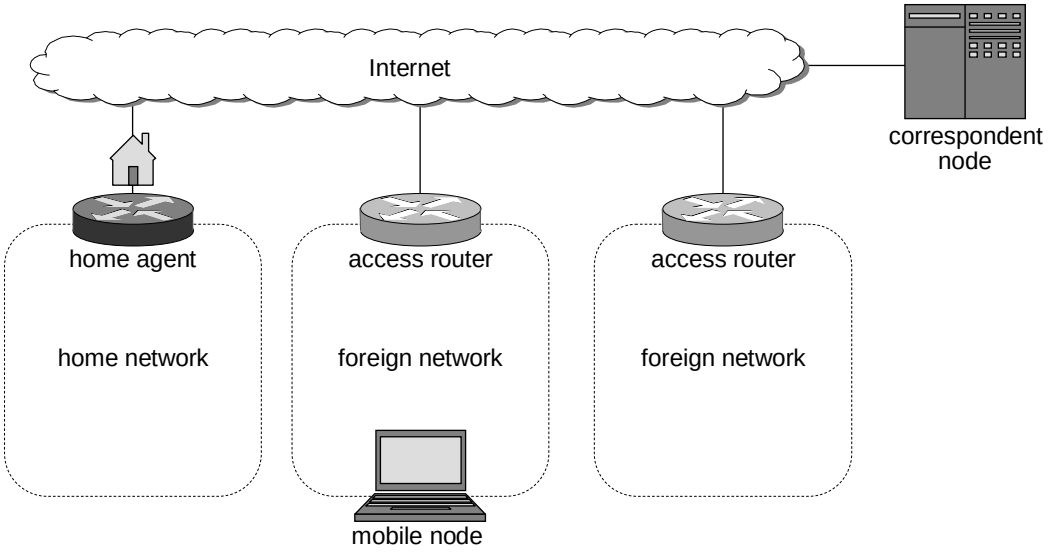



Figure 6.1: Testbed setup

Propagation delays on a given path are the same in both directions. DummyNet is not configured on the access links between the mobile node and an access router, so link-local packet propagation delays are negligible. The correspondent node and the three access routers are preconfigured with static neighbor cache entries of each other's external IP addresses so as to avoid IP address resolution over the Internet.

Most experiments use a common round-trip time, $x$, between all node pairs amongst the three access routers and the correspondent node as depicted in figure 6.2(a). This topology is referred to as the *symmetric network topology* throughout the following performance evaluation. The study of proactive mobility management is further based on the *asymmetric network topology* shown in figure 6.2(b). The round-trip time $x$ on paths between the correspondent node and either access router is here independent of the round-trip time $y$ on paths between the home agent and either access router. The round-trip time between the home agent and the correspondent node is always the maximum of $x$ and $y$.

### Movement pattern

The mobile node's experiment itinerary begins on a foreign access link after home and correspondent registrations have been completed. An experiment starts with the communication session between the mobile node and the correspondent node. The mobile node then pursues five handovers between the two foreign access links. The average pause time between successive handovers is 30 s. The pause time is randomized by $\pm 15$ s to avoid synchronization of a series of handovers with TCP's saw-tooth-like congestion window dimensioning.

### Link layer

Much contemporary, experimental work on IP mobility focuses on a particular data link and medium access technology, frequently adopting the IEEE 802.11 standard. Results from such experiments have the convenient property that they shed light on performance achievable in a certain real-life environment. On the other hand, it

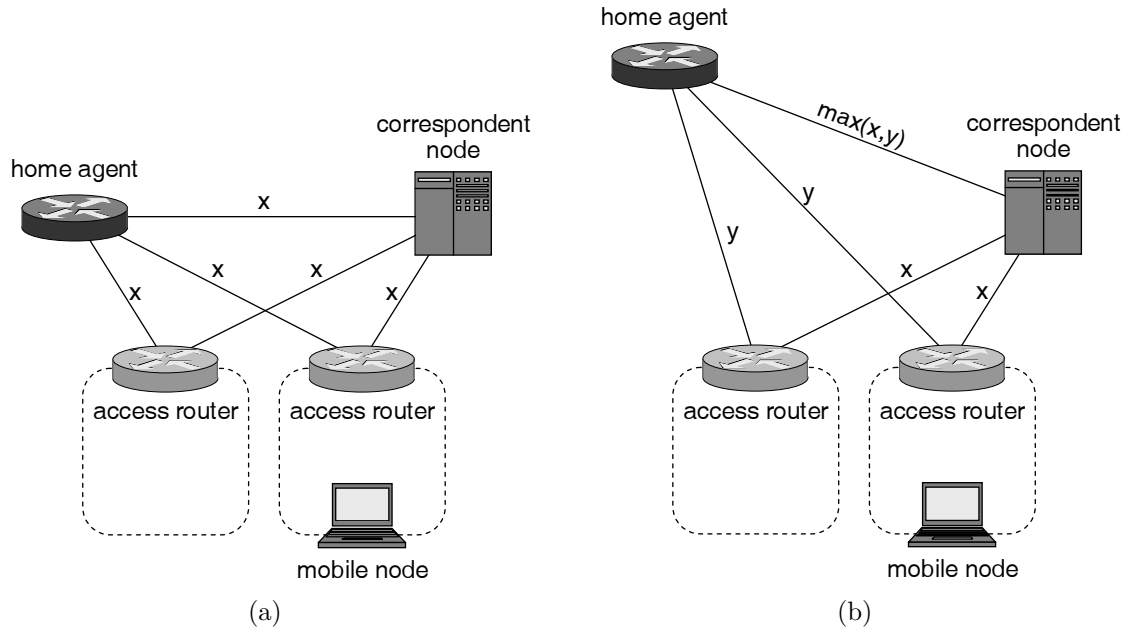



Figure 6.2: Symmetric (a) and asymmetric (b) network topologies

is generally infeasible to convey the results to different technologies. And although IEEE 802.11 prevails today, it is questionable whether this standard can accommodate the rigid requirements of delay-sensitive applications [80][133]. Sufficient performance may be achievable only through further technological improvements [42]. Research also shows that link layer characteristics may vary strongly even for a *specific* technology, given different cell loads or user application and mobility patterns [83]. While focusing experiments on a certain technology is a must for research on cross-layer interactions, it might unintendedly narrow down the results' representativity otherwise. In this study, a deliberate decision was therefore made to abstract from the properties of a specific wireless link layer technology in favor of an all-wired testbed where handovers are realized based on link layer packet filters.

### 6.1.3 TCP Buffers

The amount of data that a TCP sender can transmit without receiving an acknowledgment is, besides flow and congestion control, also limited by TCP's send and receive buffers. Since the amount of outstanding data at the time of a handover is closely related to handover-related packet loss, the capacity of these buffers can have a significant impact on the handover performance. Wise buffer dimensioning is therefore a prerequisite for meaningful experimentation results. A send buffer typically contains one window worth of unacknowledged data during periods without packet loss, while the receive buffer is mostly empty provided that the receiving application fetches arriving data eagerly. Packet loss leads to a gap in the received data and hence requires the TCP receiver to buffer up to one window worth of data until a retransmitted segment closes the gap. At the same time, the send buffer should hold the one window of data in which packet loss was suffered plus another window of new data injected while the lost segment is being resent and acknowledged. Since the minimum of both buffers limits the amount of outstanding data, a common rule

of thumb for manually configuring the TCP buffers is to set both buffers to twice the bandwidth-delay product of the transmission path [116].[1]

The bandwidth-delay product of the transmission path is seldom known in advance, however. So unless TCP buffers scale dynamically [116] with the bandwidth-delay product observed during a particular session, they need to be configured to a fixed value which may prove suboptimal for some TCP sessions. [78] provides statistics of common receive buffer sizes based on the receive windows that TCP clients advertise. The study shows that by far most clients use receive buffers of either 8 KB, 16 KB, or 64 KB, with 32 KB being the median receive buffer size. Based on an earlier analysis [4] where the median was as low as 8 KB, the authors conclude that there is a trend towards larger receive buffers. The experiments conducted in this study use TCP send and receive buffers of 64 KB as a trade-off between accordance to the aforementioned statistics, and the higher buffer size of 100 KB that would be required to hold twice the bandwidth-delay product on a path with a 400-ms round-trip time and a 1024-kbps bandwidth. It should be noted that interactive applications may intentionally configure receive buffers smaller than 64 KB in an attempt to reduce buffering delays and increase responsiveness. Non-interactive file transfers benefit from larger receive buffers, however, rendering the chosen size of 64 KB appropriate.

### 6.1.4 Applications

The performance of the different Mobile IPv6 variants has been evaluated with two inherently dissimilar communication applications—UDP-based Internet telephony and TCP-based file transfers. Internet telephony is an interactive real-time application with stringent requirements for low propagation latencies and jitter. Excessive packet loss, which may occur when one end node changes IP connectivity, becomes noticeable as annoying disruptions and thus curtails the quality of a conversation. TCP-based file transfers are more adaptive to long or variable propagation latencies, but may suffer substantial throughput degradations when a high number of packets is lost due to a handover of an end node.

A common misconception is that the non-real-time character of TCP-based file transfers would render any mobility-related performance optimizations redundant. This is frequently untrue as the ensuing evaluation shows. In fact, the negative impact of mobility on TCP performance can be strong enough to be perceptible to the user, and the proposed Mobile IPv6 optimizations can yield substantial improvement. The importance of TCP in today's Internet therefore mandates a thorough analysis of the protocol's behavior in combination with different Mobile IPv6 variants.

#### Internet telephony

Internet telephony traffic is modeled on the bidirectional 64-kbps constant-bit-rate data stream generated by a G.711 PCM codec [52]. The data stream is split into chunks of 10 ms length, each holding 80 G.711 frames of 0.125 ms length. A chunk is prepended by IPv6, UDP, and Real-time Transport Protocol [114] headers to form a packet of 164 bytes length, including the IPv6 Destination Options and Routing

[1] Reserving send buffer space amounting to two times the transmission path's bandwidth-delay product might be more than necessary because the sender's congestion window is reduced to half its previous size at the time the loss is detected. The rule of thumb does not consider this reduction. A send buffer of slightly more than 1.5 times the bandwidth-delay product should hence be sufficient.

extension headers required for Mobile IPv6 route optimization. The size increases for packets that contain a Spot Check option. Packets that the mobile node sends already include a Destination Options extension header, thus only the Spot Check option itself must be added. This increases the packet size to 188 B, assuming that only a single spot check token is returned to the correspondent node at a time. Packets that the correspondent node sends require an additional Destination Options extension header, so in this case the packet size increases to 212 B.

While not unrealistic, the chosen rate of one packet per 10 ms may appear aggressive compared to the widespread use [52] of only one packet per 20 ms. However, as the granularity of handover performance measurements is limited by the inter-packet arrival time, the higher rate has the practical advantage that the measurements are twice as precise as they would be with the lower rate. No silence compression was performed in the experiments for the same reason. Internet telephony applications may incorporate voice activity detectors to suppress or reduce the transmission of payload packets during phases of silence in conversation.

#### File transfers

The performance of file transfers in a mobile environment has been measured based on different TCP variants so as to reflect the heterogeneity in deployment that the evolution of TCP has incurred. Recent investigations [78] indicate that a growing majority of Web servers (about two thirds) and clients (nine tenths) support TCP SACK, while a shrinking population of Web servers and clients is restricted to TCP Tahoe, TCP Reno, or TCP NewReno. Independent of these base variants is the application of the Limited Transmit algorithm, which some of the deployed implementations execute. The diversity of deployed TCP variants calls for an analysis of multiple variants in the performance evaluation that follows. TCP SACK was used in most experiments due to its dominating role on today's Internet landscape; additional experiments compare the handover performance of TCP SACK with that of TCP Reno or TCP NewReno. The Limited Transmit algorithm was disabled by default and selectively enabled to evaluate its impact.[2]

### 6.1.5 IPv6 Auto-Configuration

Router discovery, movement detection, and IP address auto-configuration are substantial factors in handover performance. A large amount of effort has hence gone into developing optimized protocols for these tasks. This effort still continues [39], and it signifies that mechanisms of different performance are likely to coexist during a transition phase in the medium term. The set of IPv6 auto-configuration protocols for use in the testbed hence needs to be carefully selected.

[2] An earlier publication [140] claimed that the impact of different TCP variants is mostly negligible. This holds in most cases where mobility is managed reactively, which was the focus of [140]. The typically high packet loss during reactive mobility management causes TCP to run into a retransmission timeout in most cases and thus repair the loss in Slow Start mode. Different TCP variants behave mostly the same in such an event. On the other hand, proactive mobility management may reduce handover-related packet loss to a level where TCP attempts to recover in Fast Recovery mode. The selection of a specific TCP variant is then of more substantial impact because the variants differ mostly in the way they enter and behave in Fast Recovery mode. More sophisticated router discovery, movement detection mechanisms, and IP address auto-configuration mechanisms further contribute to reducing packet loss and hence increase the likelihood that TCP enters Fast Recovery mode upon loss detection.

#### Router discovery and movement detection

Three distinct router discovery and movement detection protocols were modeled and installed on the access routers in the testbed to account for the expected heterogeneity in deployed protocols:

1. Standard router discovery and movement detection, where access routers multicast Router Advertisement messages in intervals of between 30 ms and 70 ms, and three advertisements are required for the mobile node to reliably detect a change in IP connectivity.

2. Complete Prefix List, where sophisticated logic on the mobile node facilitates reliable router discovery and movement detection based on only a single Router Advertisement message in the majority of cases, despite standard protocols on the access router.

3. DNA protocol, which enables the mobile node to solicit an immediate Router Advertisement message that allows it to review its current IP configuration.

The DNA protocol was configured as a default in the conducted experiments since it promises to eventually prevail amongst router discovery and movement detection techniques. Other modes were used only where this is explicitly stated.

#### IP address auto-configuration

After a mobile node has performed router discovery upon the receipt of the first Router Advertisement message subsequent to a handover, the mobile node proceeds to configure a global IP address for each new subnet prefix which the advertisement contains. The standard Stateless Address Autoconfiguration protocol [129, 130], which most stationary IPv6 nodes use for this purpose, was recently supplemented by Optimistic Duplicate Address Detection [87], which allows stateless IP address auto-configuration without incurring any delays at the IP layer. Since Optimistic Duplicate Address Detection is expected to be the protocol of choice of mobile nodes, it was used throughout all conducted experiments.

### 6.1.6 Router Buffers

Routers use *forward buffers* to cache arriving packets which they cannot forward immediately due to limitations in the outbound bandwidth. The buffers provide a cushion for short-term throughput peaks of bursty traffic, reducing packet loss at the cost of buffering delays. Access routers are in addition endowed with *IP address resolution buffers* for packets that wait for the IP destination address to be resolved into the recipient node's link layer address. The routers in the testbed, all of which are access routers, further incorporate a *propagation buffer* to store the packets that are artificially delayed according to the transmission path's round-trip time. Figure 6.3 illustrates the interaction between the three buffers. The figure depicts an access router on a foreign link, yet the buffers in the home agent are the same.

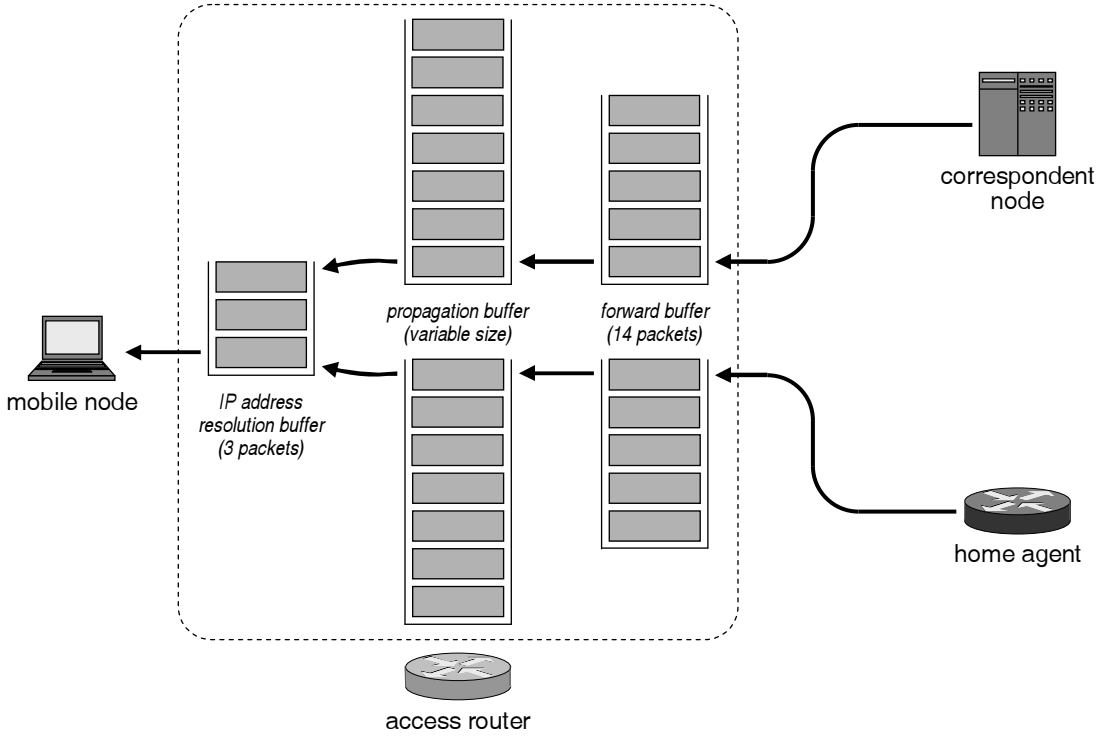



Figure 6.3: Testbed router buffers

### Forward buffers

Conventional wisdom on dimensioning a router's forward buffer states that TCP connections work best if the buffering capacity of the bottleneck router on the transmission path is sufficient to hold packets worth the transmission path's bandwidth-delay product. As the round-trip time is opaque from a router's perspective, it is typically assumed to be 250 ms. This leads to an optimal buffer space of 250 ms times the link speed according to this rule of thumb.[3] The downside of this amount of buffering space is that it may also introduce extra delays of up to 250 ms. This effect is of minor importance when the traffic to be forwarded across a specific next-hop link is less than the link's capacity, as it is the case for the conducted Internet telephony experiments. On the other hand, the saw-tooth-like throughput progression of TCP sessions periodically fills router buffers, causing buffering delays that are proportional to the current buffer load.

While delays due to forward buffering are a salient property of real Internet communications, in the testbed, they have the undesirable property of increasing the *actual round-trip time* of a path to beyond the *base round-trip time* configured on the path as described in section 6.1.2. Most of the conducted experiments are parameterized by the base round-trip time, and the measurements are plotted accordingly. Additional delays due to forward buffering therefore make the experimentation results less intuitive to understand. To reduce this effect, the forward buffers in the testbed routers were configured to half the capacity which the aforementioned rule of thumb prescribes. This leads to a maximum forward buffering delay of 125 ms possible for

[3]The true optimum capacity of router buffers is subject to ongoing research [142, 108, 44, 51].

TCP-based file transfers. Given the bandwidth of 1024 kbps, the buffer space is hence sufficient to hold a number of

$$\left\lfloor \frac{\text{bandwidth} \cdot \text{delay}}{\text{packet size}} \right\rfloor = \left\lfloor \frac{128\,000\,\text{Bps} \cdot 125\,\text{ms}}{1108\,\text{B}} \right\rfloor = \lfloor 14.440 \rfloor = 14$$

full-sized TCP segments. It should be noted that the additional buffer space that is truncated in this formula is large enough to accommodate the increased packet size which the presence of a Spot Check option in some of the packets may imply.

The differentiation between the base and the actual round-trip time requires a concretization of what a path's *bandwidth-delay product* is. The term is hence defined in this study as the product of the path's bandwidth times its base round-trip time. Accordingly, the bandwidth-delay product is the amount of data which the path can durably transport once per base round-trip time. The existence of forward buffers on the path may allow a sender to temporarily inject new data at a higher rate onto the path without packet loss.

#### Propagation buffers

The propagation buffer scales with the bandwidth-delay product of the respective transmission path. No manual configuration is required.

The simplified topology of the testbed differs from that of a real-life scenario in that the routes between the mobile node on one side and the home agent or correspondent node on the other side both pass through the same single access router. This setup would introduce unrealistic interferences between packet forwarding on the two transmission paths if an access router had just one forward buffer, or the bandwidth of 1024 kbps was shared by the packets on both paths. However, in order to reproduce the characteristics of truly separate transmission paths, the three access routers are endowed with separate forward and propagation buffers for each adjacent transmission path, and packets in one buffer do not delay any packets in a different buffer. So from the perspective of the end nodes, it seems as if each transmission path had its own bottleneck router.

#### IP address resolution buffers

A small IP address resolution buffer may cause increased loss of packets when the access router does not know the recipient mobile node's link layer address and hence cannot immediately deliver the packets that arrive for the mobile node. This is typically not an issue with reactive mobility management because IP address resolution then takes place well before any payload packets for the mobile node arrive at the access router. However, proactive mobility management may cause the delivery of payload packets to a target access router already before the mobile node attaches to the target link. Insufficient capacity of the IP address resolution buffer may then be responsible for these packets being lost. The Neighbor Discovery RFC prescribes a minimum IP address resolution buffer size of one packet, which is the value that FreeBSD defaults to. Since the IP address resolution buffer of the Linux operating system is large enough for three packets, and since this increased buffer size can improve the packet delivery ratio during proactive mobility management, the IP address resolution buffer size was increased to three packets also in the FreeBSD-based testbed.

### 6.1.7 Packet Loss

Packet loss in the testbed may happen due to overflows in forward buffers or IP address resolution buffers. With TCP file transfers, such *handover-unrelated packet loss* has been found to occur at rates of between 0.1% and 0.3% in the experiments. The loss rate is higher in topologies with small round-trip times because the frequency of TCP filling the bandwidth-delay product on the transmission path, and hence the interval between successive forward buffer overflows, is then smaller. The observed packet loss rates are commensurate with findings in [34] for paths between North America, Europe, East Asia, and Oceania. They also can be handled by TCP's fast retransmit and recovery mechanisms quite well [100], and so should occur unnoticable to the user in most cases. Handover-unrelated packet losses can have a significant impact on handover performance as they influence the size of the TCP sender's congestion window at the time of the handover.

Loss of Mobile IPv6 messages can cause handover signaling to fail and significantly degrade handover performance. This may occur when such a message cannot be enqueued into an access router's forward buffer due to the buffer being occupied by TCP segments. Lost Mobile IPv6 messages are eventually retransmitted, but the cost in terms of handover delay and packet loss can be tremendous. The efficiency of Mobile IPv6's retransmission mechanisms, as well as its interaction with Neighbor Discovery signaling on the mobile node's access link, is still a subject for further research. Due to the complexity involved, it is clearly out of scope of this study. The ensuing performance evaluation hence focuses on handovers during which loss of Mobile IPv6 messages does not happen. In an effort to avoid scanning experimentation logs for Mobile IPv6 retransmissions and removing those logs that match, a packet classifier was added to the DummyNet code which implements the forward buffer in access routers. This limits packet loss to payload packets, avoiding the distractive influence that the loss of signaling packets would have on experiment results. Theoretically, signaling packets may still fall victim to overflows in an IP address resolution buffer, but this did not happen in the conducted experiments.

The sending rate of an Internet telephony application never exceeds the 1024 kbps of bandwidth available on a transmission path, so handover-unrelated packet losses do not occur in Internet telephony experiments. This does not reduce the realisticness of these experiments, however, because the application does not adapt its throughput in the event of congestion, and the amount of outstanding data at the time of a handover would be the same even in the presence of preceding packet loss.

## 6.2 Internet Telephony

Interactive real-time applications have gained in importance tremendously during the recent past, and this trend is expected to continue [81]. Yet the stringent requirements that these applications have in terms of delay and packet loss make it difficult to support them in mobile environments as even short interruptions during handover can lead to perceptible degradations in speech quality. Route optimization in standard Mobile IPv6 reduces packet propagation delays to a minimum and thus accommodates real-time applications while a mobile node stays at one point of attachment without moving. But extended handover delays and handover-related packet losses have been found to adversely impact these same applications when the

mobile node changes IP connectivity. This section evaluates the handover performance benefits that can be obtained for interactive, real-time applications through the use of Early Binding Updates and Credit Based Authorization, and it puts the result in relation to the performance of standard Mobile IPv6. The measurements were conducted with IP telephony. Given the similar needs of interactive real-time applications in general, it is expected that experiments with other such applications will yield comparable results.

### 6.2.1 Packet Loss

Changes in IP connectivity are typically responsible for the loss of payload packets that a correspondent node sends to a mobile node, and possibly also for the loss of payload packets that the mobile node sends to the correspondent node. In reactive mobility management, lost packets from the correspondent node are all sent to the mobile node's old care-of address, but fail to be delivered because the mobile node is no longer reachable at the old point of attachment. In proactive mobility management, loss may also affect packets that the correspondent node directs to the mobile node's new care-of address, and that fail to be delivered because the mobile node is not yet present at the new point of attachment. Payload packets that the mobile node sends may be dropped locally in the mobile node's network stack, be it during the link layer handover, during subsequent IPv6 auto-configuration, or due to a conservative mobile node's policy not to send any payload packets until the home registration is complete.

Most codecs used in Internet telephony applications incorporate algorithms to conceal the loss of a small amount of payload data, which is typically equivalent to one or two payload packets [52, 61] depending on the codec and the packet size. The goal of such *loss concealment* is to mask the properties of lossy environments such as wireless access links, where transmissions may sporadically fail due to collision, interference, or obstruction. On the other hand, loss concealment algorithms are poor in repairing loss *bursts*, and they fall short of hiding the erasure of multiple payload packets that typically occurs when a mobile node changes IP connectivity. Handover-related packet loss consequently translates into moments of blackout in speech, causing annoying disruption during conversation. The Internet telephony application [137] used in these experiments produces a constant-bit-rate payload data stream which does not undergo silent suppression, so the number of payload packets lost during handover is proportional to the length of the blackout that results from it.

*Handover-related packet loss*, or simply *packet loss*, is here defined as the number of payload packets that the correspondent node sends to the mobile node and that cannot be delivered due to the mobile node's move between two points of attachment. The loss of packets in this traffic direction is generally higher than the loss affecting the packets that the mobile node sends. The correspondent node can redirect its packets to the new care-of address only when it receives a standard or early Binding Update message from the mobile node, and the transmission of this message may require an a-priori home registration and return routability procedure depending on the Mobile IPv6 variant. The length of the packet loss phase in reactive mobility management is hence in the order of round-trip times. The mobile node, on the other hand, knows about its current care-of address being stale already when movement

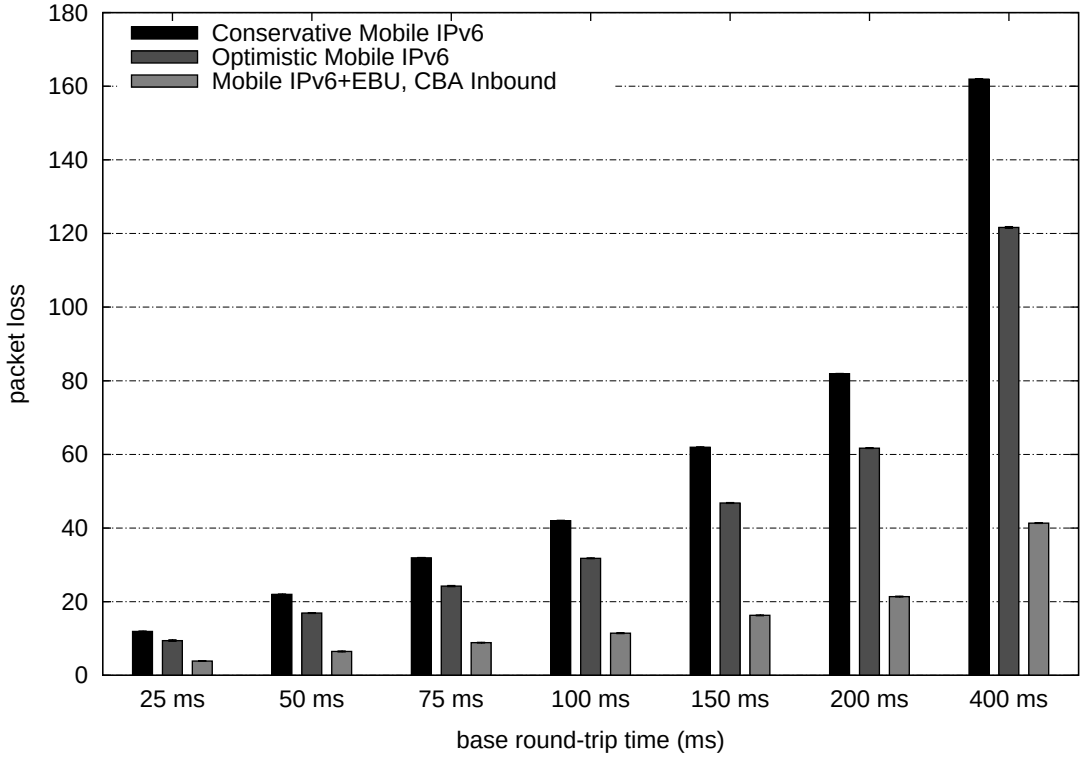



Figure 6.4: Mean packet loss for Internet telephony and reactive mobility management

detection completes. The immediate availability of a new care-of address, which Optimistic Duplicate Address Detection permits, then enables the mobile node to send subsequent packets with a topologically correct IP source address. The loss of packets that the correspondent node sends is consequently at least as high as the loss of packets that the mobile node sends. The chosen packet loss metric is based on the former traffic direction and hence accommodates Internet telephony's natural dependency on bidirectional packet delivery.

Furthermore, some Mobile IPv6 variants require the mobile node to temporarily send its payload packets via the home agent after a handover, so as to ensure the delivery of these packets despite an out-of-date binding at the correspondent node. Reverse-tunneled packets experience increased propagation latencies and are hence at risk of not being delivered in time to the receiving Internet telephony application. They may get dropped eventually even though the correspondent node receives them. The chosen packet loss metric accommodates this need for timeliness. The measured packet loss in the direction from the correspondent node to the mobile node is always at least as much as the packet loss in the opposite direction, so the metric covers any period of reverse-tunneling and potentially late packet delivery. The metric also implies that it makes no difference whether the mobile node begins route-optimizing its payload packets for the correspondent node once it has sent the Binding Update message to the correspondent node, or whether it continues reverse-tunneling until it receives the responding Binding Acknowledgment message. The period of reverse-tunneling is covered by the metric in any case.

Figure 6.4 juxtaposes the average handover-related packet losses measured in experiments with conservative Mobile IPv6, optimistic Mobile IPv6, and Mobile IPv6 with Early Binding Updates. The experiments were repeated for symmetric network toplogies with base round-trip times of between 25 ms and 400 ms. Link layer handover delays are zero, and the DNA protocol eliminates most of the IP layer router discovery and movement detection delays. Since Optimistic Duplicate Address Detection further provides the mobile node with a new care-of address immediately afterwards, the handover delay introduced at the Mobile IPv6 level dominates the total handover delay that is visible to the application. The packet loss results were obtained from 100 handovers recorded in 20 experiments per Mobile IPv6 variant and base round-trip time.

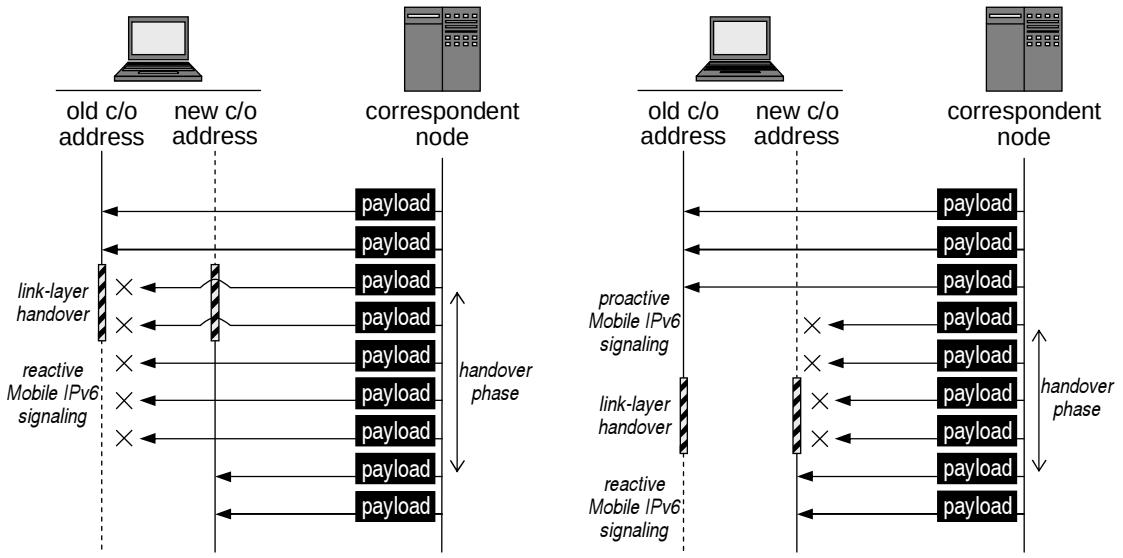
The measurements clearly reflect the different signaling latencies claimed by the observed Mobile IPv6 variants: Conservative Mobile IPv6 requires 3.5 round-trips to update the mobile node's binding at the correspondent node, and an additional one-way time elapses until the correspondent node's first payload packet reaches the mobile node at the new care-of address. The loss of packets that the correspondent node sends to the mobile node is hence equivalent to four round-trip times, yielding increasingly long speech blackouts as the network paths become longer. (The actual round-trip time of a path can be assumed to be equal to that path's base round-trip time in the experiments with Internet telephony due to the absence of notable forward buffering delays. The applications' sending rate is here well below the 1024 kbps of bandwidth available, so forward buffering delays cannot accrue.) The mobile node reverse-tunnels payload packets for the correspondent node via its home agent from the time movement detection completes up to when it receives the correspondent node's Binding Acknowledgment message.

Optimistic Mobile IPv6 reduces the blackout in speech transmitted from the correspondent node to the mobile node down to three round-trip times by parallelizing the home registration with the return routability procedure. The reduction in packet loss that results from this demonstrates the benefit that conservative Mobile IPv6 implementations can obtain by adopting optimizations that are compliant to RFC 3775. Mobile IPv6 with Early Binding Updates updates a binding within a single one-way time. Adding the propagation latency of the first payload packet delivered to the new care-of address yields a loss equivalent to one round-trip time between the mobile node and the correspondent node, and hence optimal performance for reactive end-to-end mobility management.

The relationship between the three Mobile IPv6 variants stretches across all of the observed round-trip times. The number of lost packets, $loss_1$, $loss_2$, $loss_3$, for conservative Mobile IPv6, optimistic Mobile IPv6, and Mobile IPv6 with Early Binding Updates, respectively, can be mathematically expressed as

$$loss_1 \approx \frac{4\,x}{I_{UDP}}, \quad loss_2 \approx \frac{3\,x}{I_{UDP}}, \quad loss_3 \approx \frac{x}{I_{UDP}}$$

where $x$ is the base round-trip time used in the symmetric network topology, and $I_{UDP}$ is the inter-arrival time of the Internet telephony packets, which equals 10 ms in these experiments. Since $I_{UDP}$ is constant and forward-buffering delays were negligible, packet loss measurements vary only very little for a given value of $x$. This effectively renders the 95% confidence intervals printed in figure 6.4 are invisible.




(a) Reactive mobility management (b) Proactive mobility management

Figure 6.5: Composition of handover phase in Internet telephony

### 6.2.2 Handover Delay

Changes in IP connectivity on the mobile-node side may disrupt Internet telephony sessions through packet loss or temporary unidirectional packet delivery. Undisturbed conversations are then delayed until bidirectional packet delivery has been restored. These circumstances suggest the definition of the *handover phase* as the period between the correspondent node's transmission of the payload packet that immediately follows the last payload packet successfully received by the mobile node at the old care-of address up to the point at which the correspondent node transmits the first payload packet that the mobile node again successfully receives at the new care-of address. Figure 6.5 illustrates this definition.

The *handover delay* is defined as the length of the handover phase and hence approximates the time span during which speech transmission from the correspondent node to the mobile node intermits. The silence observed by the correspondent node itself may be shorter if reverse-tunneled payload packets from the mobile node reach the correspondent node in time, or if the mobile node resumes route optimization at the time it sends the Binding Update message to the correspondent node rather than waiting for the reception of a Binding Acknowledgment message. The silence at both peers is equally long if the mobile node continues reverse-tunneling payload packets up to the reception of the correspondent node's Binding Acknowledgment message, and the reverse-tunneled packets are dropped by the receiving application as stale. So as in the case of handover-related packet loss, the handover delay is longer in the traffic direction from the correspondent node to the mobile node than it is in the reverse direction. The interactiveness of Internet telephony applications hence justifies the definition of the handover phase and latency from the perspective of the correspondent node. For a handover without any packet losses, the definitions imply that the beginning and the end of the handover phase coincide with the transmission of the first payload packet directed to the new care-of address, and the handover delay reduces to zero.

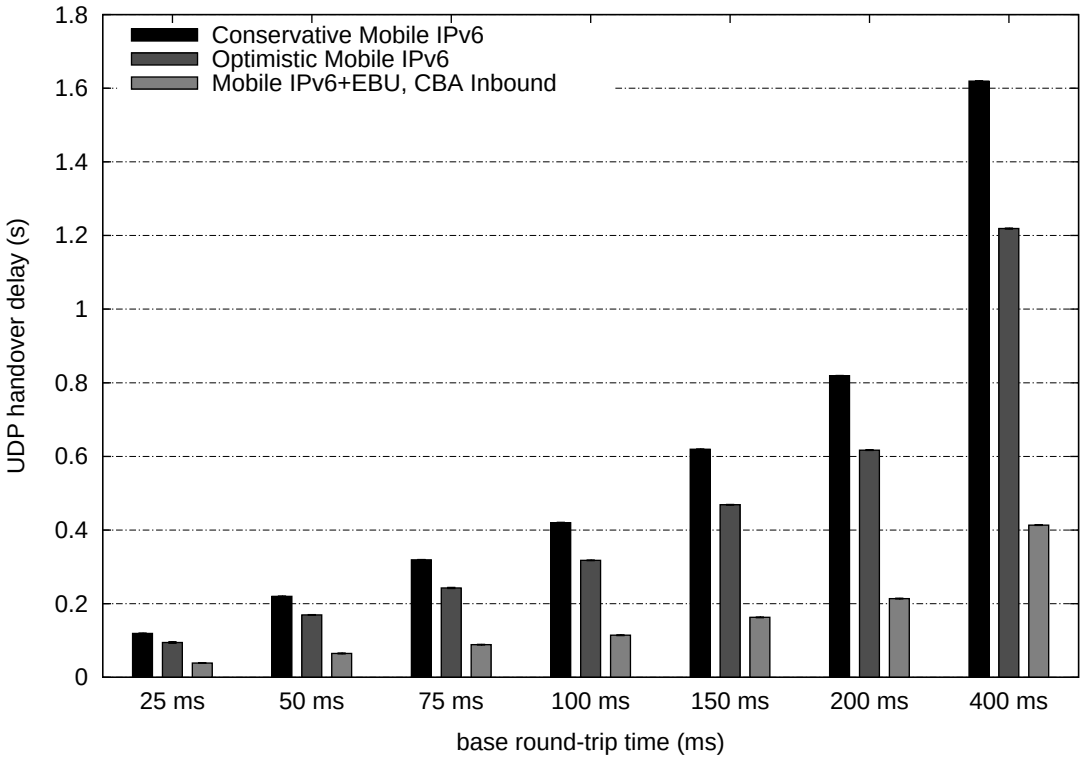



Figure 6.6: Mean handover delay for Internet telephony and reactive mobility management

Figure 6.6 shows the average handover delays measured in the experiments that were introduced in section 6.2.1. Handover delays match the signaling latencies consumed by the respective Mobile IPv6 variants, and they are proportional to packet loss due to the constant bit rate at which the observed Internet telephony applications send. So if $latency_1$, $latency_2$, and $latency_3$ are the handover delays of conservative Mobile IPv6, optimistic Mobile IPv6, and Mobile IPv6 with Early Binding Updates, respectively, then these can be mathematically expressed as

$$latency_1 \approx 4\,x, \quad latency_2 \approx 3\,x, \quad latency_3 \approx x$$

where $x$ is the base round-trip time used in the symmetric network topology. The optimizations of optimistic Mobile IPv6 and Mobile IPv6 with Early Binding Updates yield an increasing benefit as round-trip times grow across the network topologies shown. With a handover delay of one round-trip time, the performance of Early Binding Updates is optimal for reactive end-to-end mobility management.

The small variation in measured packet loss implies a similarly small variation in measured handover delays. The 95% confidence intervals printed in figure 6.6 thus become invisible.

### 6.2.3 Credit Availability

One of the benefits of Early Binding Updates and Credit-Based Authorization is that the techniques enable mobile and correspondent nodes to continue communications

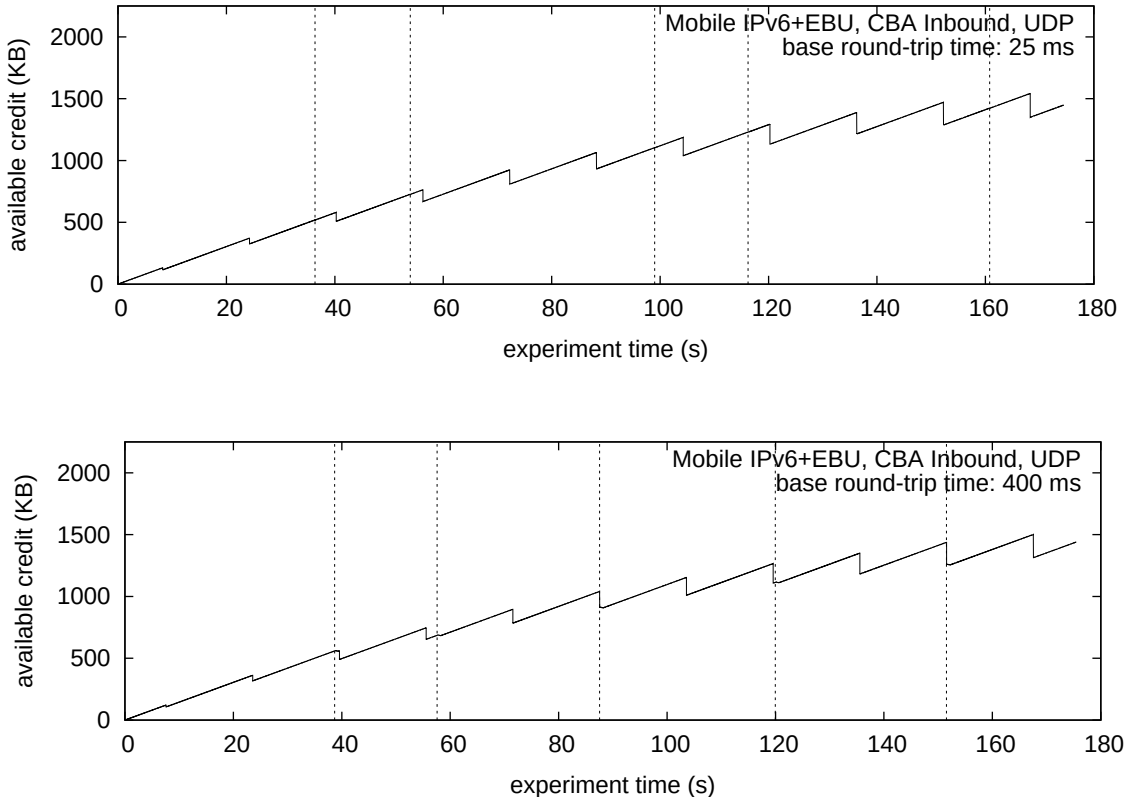



Figure 6.7: Credit availability in CBA Inbound mode

early on after a handover while the mobile node's new care-of address is still in Unverified state. This feature, however, hinges on the mobile node's available credit not running out until the new care-of address moves to Verified state. Lack of credit may be due to a long reachability verification latency. It may also happen if the mobile node changes IP connectivity more frequently than would allow it to refill the consumed credit during the time between successive handovers. Finally, credit could be insufficient if credit aging eliminates the credit faster than it can be turned into data sent to an unverified care-of address.

Figure 6.7 shows the amount of credit the mobile node earns and consumes over time in two Internet telephony experiments where Credit Based Authorization is used in Inbound mode. The experiments were conducted in symmetric network topologies. The base round-trip time on all paths is 25 ms in the top chart and 400 ms in the bottom chart. Credit aging leads to a drop in the available credit by 1/8 every 16 s, giving rise to the zigzag curves in the figure. Yet despite the periodic reductions, the credit continually grows until it reaches a point at which the new credit earned within a period of one crediting interval equals 1/8 of the total credit, and is hence eliminated by aging at the end of the crediting interval. The upper credit limit is thereby determined by the mobile node's sending rate. This is constant, so credit acquisition is independent of the base round-trip time.

The vertical lines in both charts of the figure mark the times at which the mobile node pursues a handover. Payload packets that the correspondent node sends to a care-of address in Unverified state at around this time each consume credit worth

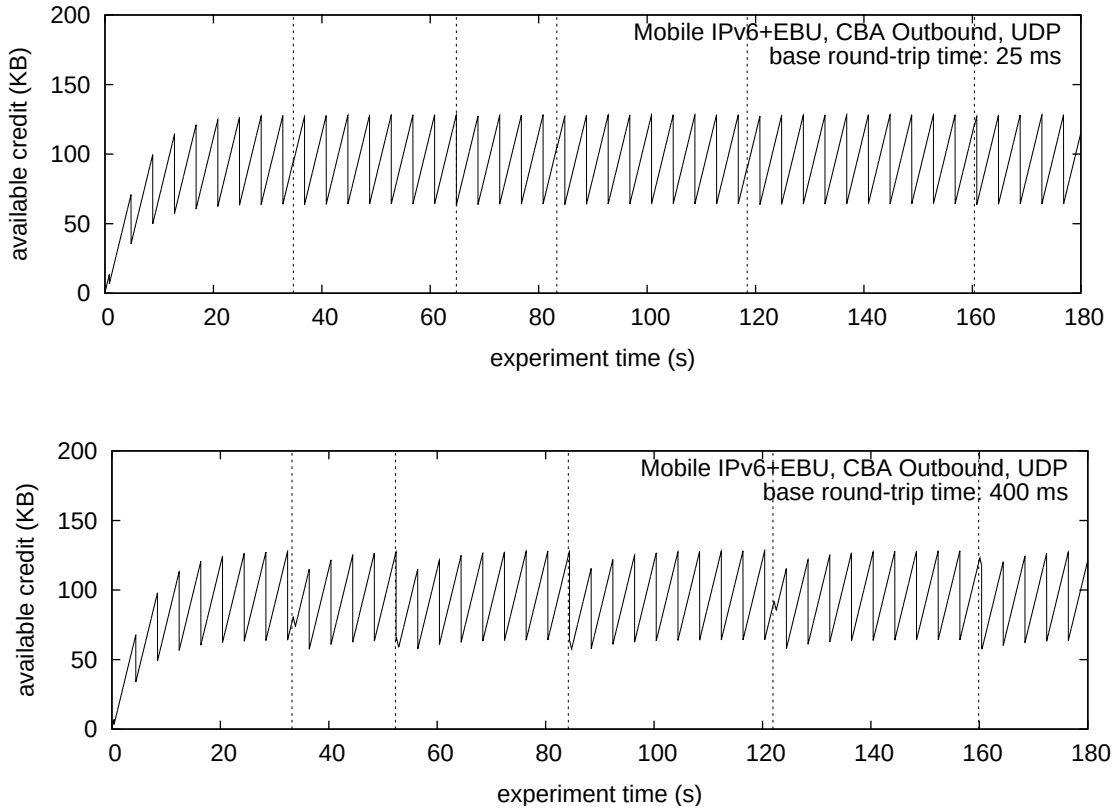



Figure 6.8: Credit availability in CBA Outbound mode without spot check support

164 B. The correspondent node sends at a constant rate, so credit consumption increases with the round-trip time on the new transmission path. However, the charts indicate that the consumed credit is negligible compared to the amount of credit available even if the base round-trip time between the mobile node and the correspondent node is 400 ms. This is due to the fact that the credit aging function has been designed to accommodate the asymmetric traffic properties of TCP, where credit collection proceeds at a much lower rate than credit consumption does during handover. However, the traffic patterns of UDP-based Internet telephony applications are symmetric, so the mobile node ends up gathering a multiple of the credit that it may actually require. This high availability of credit does not limit the security that Credit-Based Authorization provides against redirection-based flooding attacks, however. After all, the mobile node still cannot earn more credit than is equivalent to the data it sends, and due to aging, this credit can only be kept for a certain amount of time.

Credit aging is more rigid in Outbound mode, and the amount of credit available is consequently lower. Figure 6.8 demonstrates this with two charts from experiments with Outbound mode. Like in figure 6.7, these experiments were obtained from a symmetric network topology with base round-trip times of 25 ms and 400 ms in the top and bottom charts, respectively. Evidently, the lower credit availability in Outbound mode does not lead to a credit shortage during handover, avoiding negative performance implications.

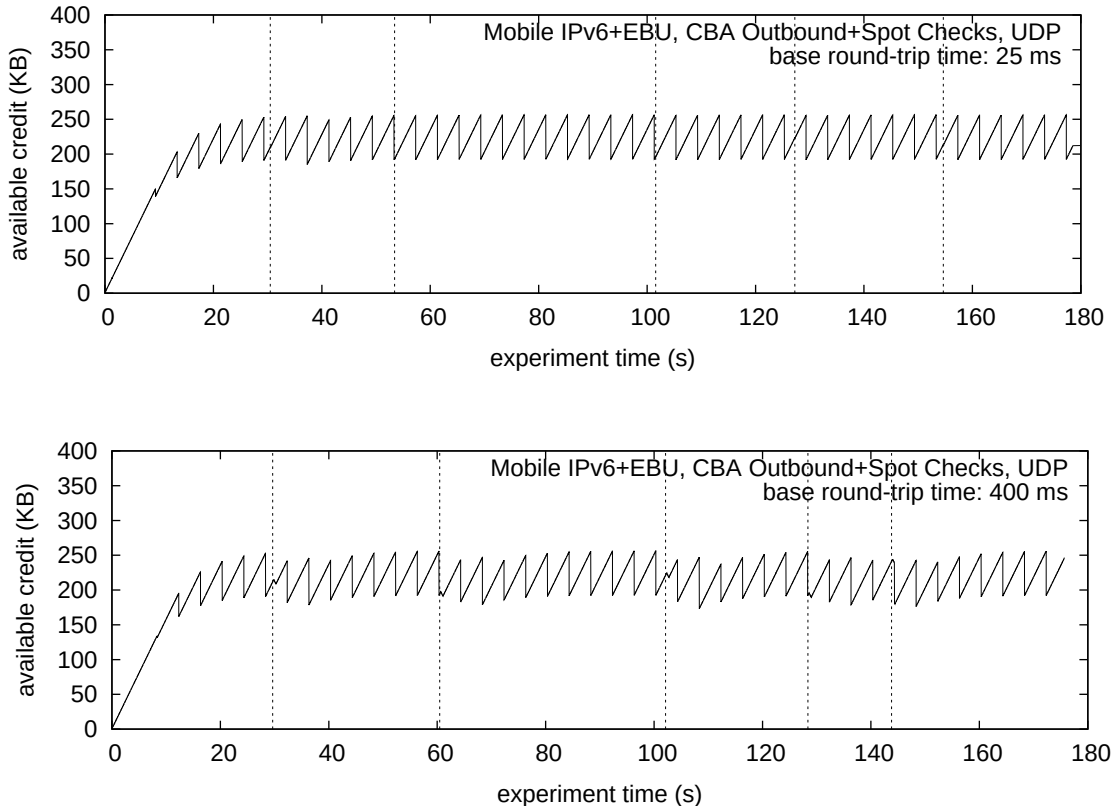



Figure 6.9: Credit availability in CBA Outbound mode with spot check support

The use of spot checks in conjunction with Credit-Based Authorization Outbound mode is conceptually of little importance with respect to credit availability. The Internet telephony applications used in the experiments provide a steady flow of packets onto which the mobile node can piggyback any spot check tokens received from the correspondent node. Deposit on the correspondent node side is therefore in most cases turned into credit eventually. One exception holds for sporadic, handover-unrelated packet loss. When this affects a packet including a Spot Check option, newly assigned credit is temporarily less. Overall however, handover-unrelated packet loss has a minor impact given that it only occurs at rates of between 0.1% and 0.3%. The mobile node's available credit is hence on average at most 0.3% less if spot checks are used. The only more substantial difference which the use of spot checks makes regards the payload packets that the correspondent node sends to the mobile node's old care-of address while the mobile node is handing over to a new point of attachment, assuming that the old care-of is in Verified state at that time. Unless spot checks are used, these packets are fully turned into credit even though they are not received by the mobile node. Spot checks can unveil such packet loss, thus reducing the credit available during the handover. The number of packets that the correspondent node sends to the old care-of address in vain, and hence the amount of credit these packets are worth, depends on the latency of the mobile node's link layer handover, any delays for IPv6 auto-configuration on the new link, as well as the actual round-trip time on the new transmission path accounting for the propagation of the mobile node's early Binding Update message and the correspondent node's first payload packet directed to the new care-of address. The top

and bottom charts in figure 6.9 represent scenarios with the same parameters as before with the exception that the correspondent node now applies Outbound mode with spot checks in order to estimate the delivery ratio of the payload packets it sends. The charts notably differ from the previous ones in that credit aging now is no longer visible in the form of periodic drops in the available credit. This is because the spot check implementation that was used for these experiments differs from the specification in section 4.3 in that the correspondent node keeps credit as deposit for an entire aging interval, even after the corresponding spot check tokens have been successfully returned. The deposit is then aged right before it is turned into credit—a behavior that is based on an earlier specification which was revised in the meantime. The charts in figure 6.9 hence *underestimate* the mobile node's available credit slightly, because the assignment of new credit may happen up to one crediting interval later than according to the specification in section 4.3. Given that no credit shortage occurs in the conducted experiments, it can be concluded that none would happen with an implementation that turns deposit into credit directly when a spot check token has been successfully returned.

## 6.3 Internet Telephony with Proactive Mobility Management

The preceding sections corroborate the improvements in handover performance that optimizations for Mobile IPv6 can have if mobility is managed reactively. Beyond this, the possibility to defer reachability verification for new care-of addresses, which Early Binding Updates and Credit-Based Authorization permit, enables mobile nodes that can anticipate changes in IP connectivity to handle mobility in a proactive manner. This facilitates additional performance improvements.

### 6.3.1 Packet Loss

The theoretic elaboration in section 5.3 shows that proactive mobility management based on Mobile IPv6 with Early Binding Updates can eliminate packet loss provided that a suitable network topology and sufficiently small handover delays at the link layer or outside Mobile IPv6 at the IP layer enable the mobile node to leave the old point of attachment after it has received the last payload packet destined to the old care-of address, and to arrive at the new point of attachment early enough to receive the first payload packet sent to the new care-of address. This claim has been verified based on Internet telephony experiments with proactive mobility management in Home Priority mode, Correspondent Priority mode, and Impatient mode, both in symmetric and asymmetric network topologies. The DNA protocol is used for router discovery and movement detection, and link layer handover delays are zero.

In the symmetric network topologies, the common base round-trip time on all paths ranges from 25 ms to 400 ms. The expected elimination of packet losses turned out to occur for all measured handovers without a single exception. This unambiguousness and reproducibility is mostly due to the regular sending patterns of the monitored Internet telephony applications. Another contributing factor is the absence of round-trip time variations, which forward buffering delays on the transmission path may induce: Furthermore, since the actual round-trip times are on all path the same,

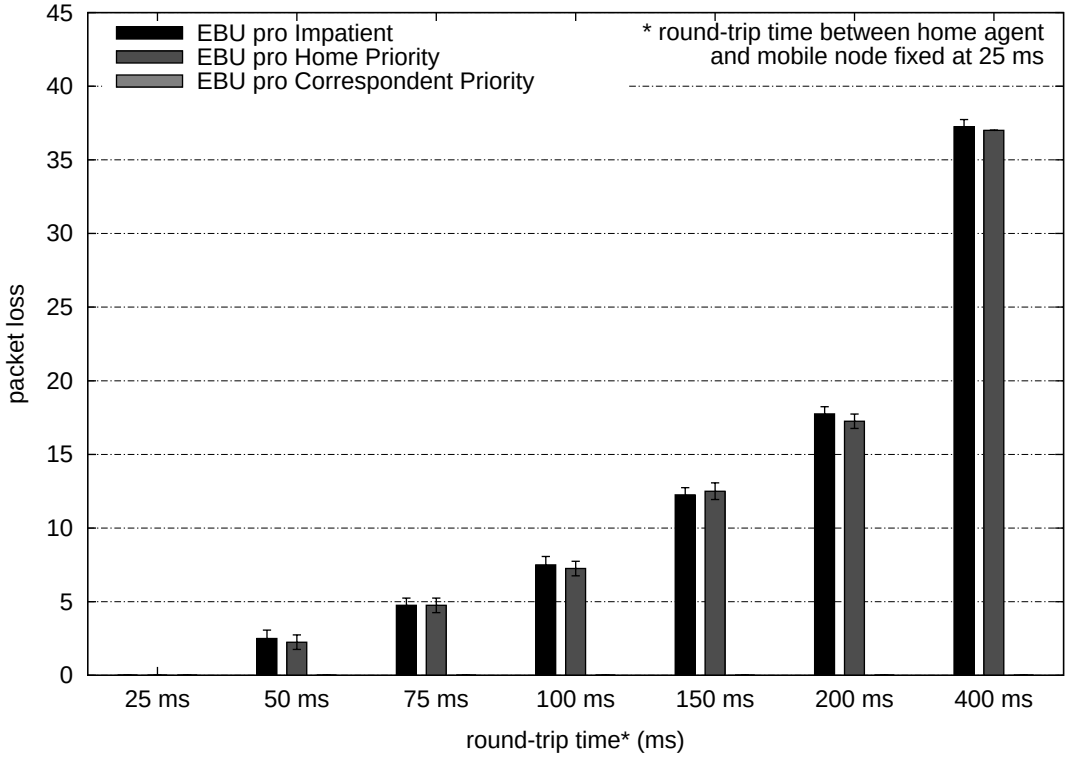



Figure 6.10: Mean packet loss for Internet telephony and proactive mobility management

the home agent's and the correspondent node's Binding Acknowledgment messages always arrive at the mobile node at about the same time, which means that Home Priority mode, Correspondent Priority mode, and Impatient mode end up performing equally. The initiation of the link layer handover thus follows the delivery of the last payload packet at the old point of attachment in all of the three modes, escaping any loss at the old point of attachment. At the same time, loss of early packets at the new point of attachment is unlikely as well because packet delivery at the old and new points of attachment does not overlap. Moreover, the new access router's IP address resolution buffer could salvage up to three packets that arrive at the new point of attachment before the mobile node does. Given that packet loss was zero without exception in each of the three modes of proactive mobility management, the results are not displayed in a separate diagram here.

The effect that an asymmetric network topology has on handover-related packet loss has been studied based on two further sets of experiments. Figure 6.10 shows packet loss averages and 95% confidence intervals from topologies with a constant round-trip time of 25 ms between the home agent and any other node, and varying round-trip times of between 25 ms and 400 ms between either access router in the visited networks and the correspondent node. Consistent with the foregoing analysis for symmetric network topologies, the datum on the left-hand side of the diagram shows that packet loss is zero in the special case where the base round-trip times is 25 ms on all paths. Packet loss in Correspondent Priority mode continues to be minimum also as the network topologies become more and more asymmetric because the time at which the mobile node pursues the link layer handover is in this

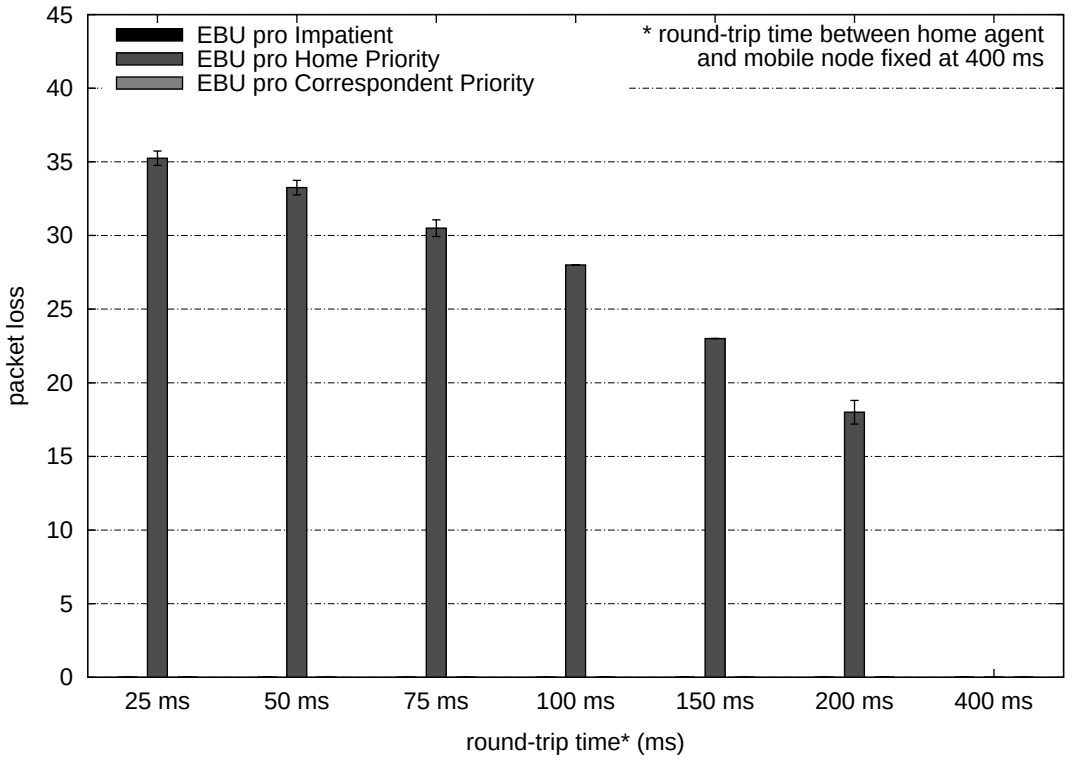



Figure 6.11: Mean packet loss for Internet telephony and proactive mobility management

mode always synchronized with the redirection of the correspondent node's payload packets from the old care-of address to the new one. The low round-trip time on the paths to the home agent hence has no influence. However, the increasing asymmetry causes packet loss to grow where Home Priority mode is used because the mobile node then changes links before the last payload packet destined to the old care-of address has been delived. If $x$ is the base round-trip time between the mobile node and the correspondent node, $y$ is the shorter or equal base round-trip time on the paths incident to the home agent, and $I_{UDP}$ is the inter-arrival time of the Internet telephony packets, then the number of lost packets generally amounts to

$$loss_{p1} \approx \frac{x-y}{I_{UDP}} = \frac{x-25\,\mathrm{ms}}{10\,\mathrm{ms}}$$

The results for Impatient mode are similar to those of Home Priority mode because the home registration latency is always less than or equal to the correspondent registration latency in these experiments, rendering Home Priority mode equivalent to Impatient mode.

Figure 6.11 summarizes the average packet losses and 95% confidence intervals in experiments where the home registration is longer than the correspondent registration. The round-trip between the home agent and any other node now takes a constant 400 ms, while the round-trip time between the mobile node and the correspondent node varies as before from 25 ms to 400 ms. Again, the results for Correspondent Priority mode are not affected by the new asymmetry. But Home Priority mode requires the mobile node to stay longer than optimum at the old point of attachment

so that packets already arrive at the new point of attachment in the mobile node's absence. The IP address resolution buffer in the target access router can salvage up to three of these packets, averting packet loss if the mobile node arrives on the new link before the fourth packet is delivered. The number of packets lost then equals

$$loss_{p2} \approx \max\left(0,\ \frac{y-x}{I_{UDP}} - 3\right) = \max\left(0,\ \frac{400\,\text{ms} - x}{10\,\text{ms}} - 3\right)$$

where $x$ is again the round-trip time between the mobile node and the correspondent node, and $y$ is the round-trip time on all paths incident to the home agent. Packet loss at the new point of attachment accordingly becomes inevitable if the round-trip time difference, $y - x$, reaches $3\,I_{UDP}$. Since the correspondent node's Binding Acknowledgment message arrives before the home agent's in these experiments, Impatient mode becomes equivalent to Correspondent Priority mode.

### 6.3.2 Handover Delay

The definition of handover delay in section 6.2.2 implies a delay of zero for handovers without packet loss. This is reasonable since such handovers are invisible from an application's perspective.[4] Handovers that do not cause packet loss may happen if either the mobile node runs in Correspondent Priority mode and thus initiates the link layer handover at a time that is optimal with respect to the redirection of the correspondent node's payload packets from the old care-of address to the new one, or if the network topology is symmetric and the mode of proactive mobility management does not make a difference. These scenarios have been discussed in section 6.3.1, and zero packet loss has been found to occur predictably across all of them. The handover delays in the same scenarios are consequently zero as well, and hence not plotted in a separate diagram here.

On the other hand, asymmetric network topologies do cause handover delays in case the mobile node operates in Home Priority mode or, if the home registration latency is shorter than the correspondent registration latency, in Impatient mode. Figure 6.12 summarizes the average handover delays and 95% confidence intervals for those experiments from section 6.3.1 where the base round-trip time between the home agent and any other node is fixed at 25 ms, and the base round-trip time between the mobile node and the correspondent node varies from 25 ms to 400 ms. The left-hand side datum of the figure relates to a network topology where the paths between all pairs of nodes are equally long, so the handover delay is here still zero for all modes of proactive mobility management. The handover delay continues to be zero also for more asymmetric network topologies if the mobile node operates in Correspondent Priority mode. However, loss of payload packets at the old point of attachment leads to handover delays in Home Priority mode as the difference in round-trip times grows. The same holds for Impatient mode, which in this case is equivalent to Home Priority mode due to the small home registration latency.

Overall, it can be seen that handover delays are proportional to packet loss. The regular sending pattern of the correspondent node and the absence of forward buffer-

[4] Buffering of packets in a new access router during IP address resolution may introduce some delays even for handovers without packet loss. This delay is not captured by the definition of handover delay used in this thesis.

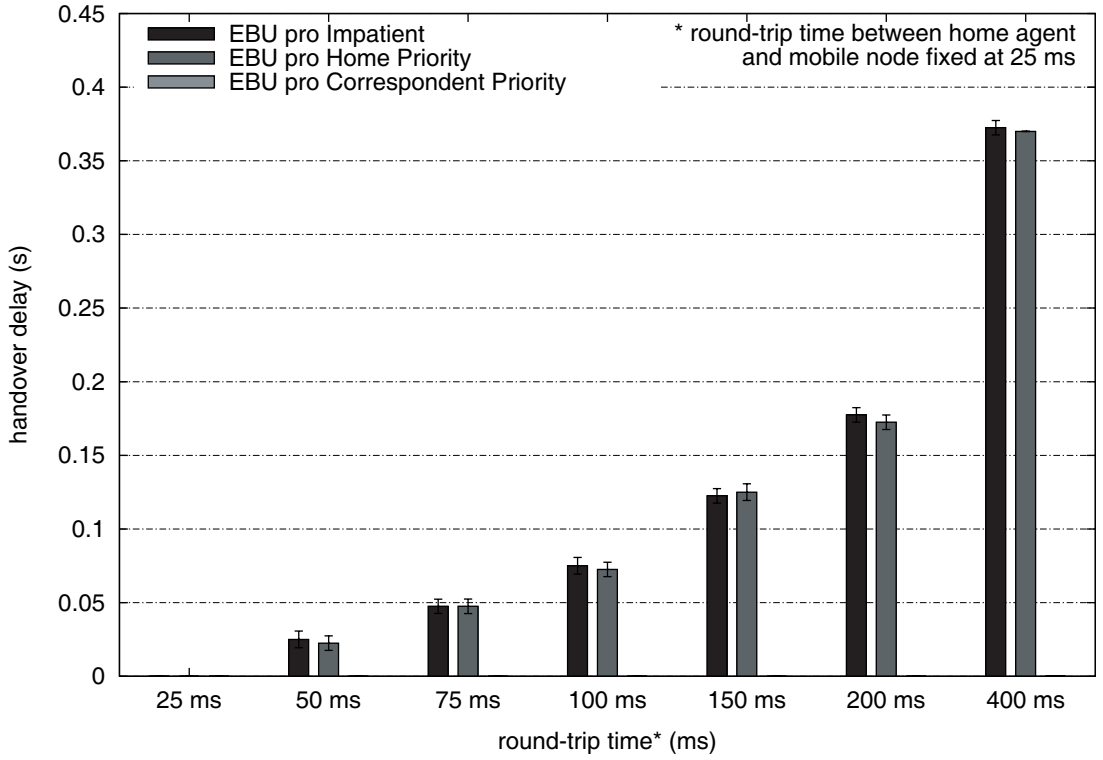



Figure 6.12: Mean handover delay for Internet telephony and proactive mobility management

ing delays on the transmission path give rise to the following theoretic derivation of the handover delay as a function of the number of packets lost during handover:

$$handover\ delay = I_{UDP} \cdot loss$$

Figure 6.13 shows the average handover delays and 95% confidence intervals for experiments where the round-trip time on the paths incident to the home agent is fixed at 400 ms and the common round-trip time on all other paths again varies between 25 ms and 400 ms. The measurements for Correspondent Priority mode reveal the absence of handover delays across all topologies as before. The same holds for the measurements for Impatient mode, since the correspondent registration latency is never longer than the latency of a home registration. Handover delays do occur if the mobile node operates in Home Priority mode. They are highest in the scenario represented by the left-most datum in the figure due to the large difference between the home and correspondent registration latencies, and they shrink as the difference becomes less across the scenarios further to the right. The handover delays of all three modes of proactive mobility management are zero on the right-most datum in the figure since the latencies of the home and correspondent registrations are the same in this case.

### 6.3.3 Credit Availability

Credit consumption during a handover is higher in proactive mobility management than it is in reactive mobility management due to a longer time during which the

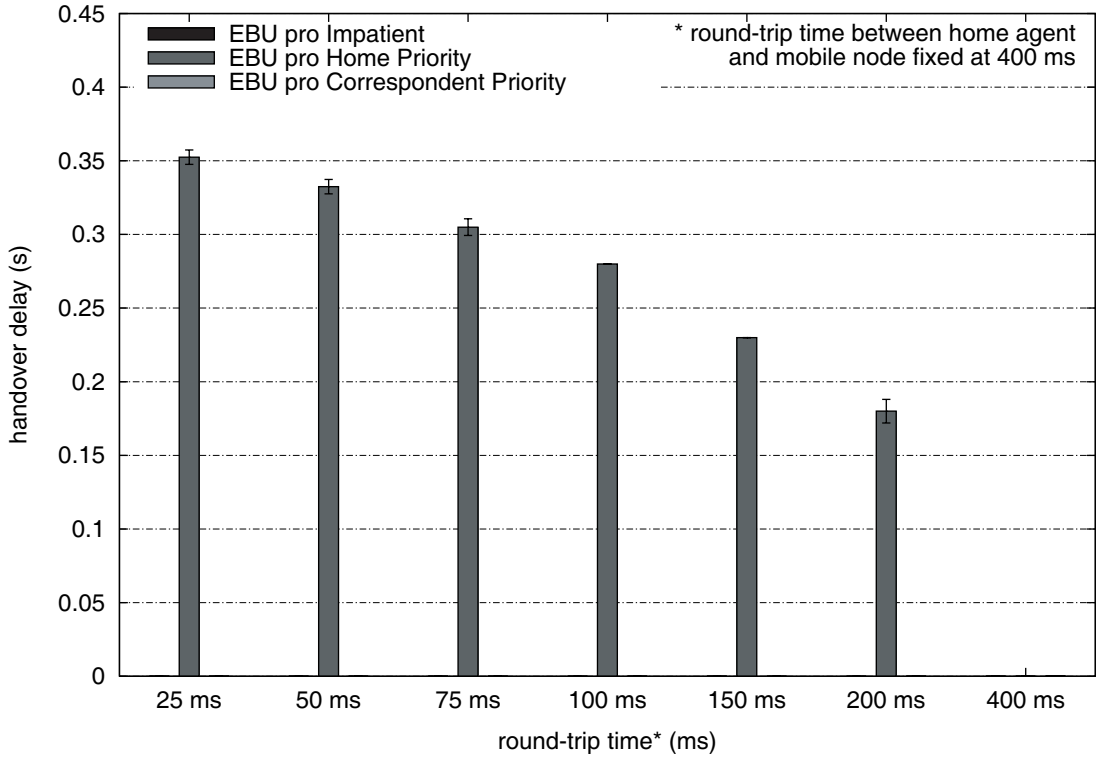



Figure 6.13: Mean handover delay for Internet telephony and proactive mobility management

mobile node' new care-of address is in Unverified state. In reactive mobility management, a new care-of address is typically in Unverified state for not more than one round-trip time between the mobile node and the correspondent node because reachability verification is already initiated in parallel with the binding update. On the other hand, the reachability of a care-of address that is proactively registered prior to handover can be verified only after the handover, so the time during which the care-of address remains in Unverified state is necessarily longer. The phase may be further prolonged by additional handover delays at the link layer or the IP layer other than those of Mobile IPv6 itself. Credit consumption is proportional to the length of this phase given that the Internet telephony application at the correspondent node sends at a constant rate without disruption.

The amount of credit consumed in proactive mobility management is particularly high when both, the mobile node's stay at the old point of attachment after sending the Binding Update messages, and the verification of the mobile node's reachability at the new point of attachment, takes long. Amongst the experiments evaluated so far, maximum credit consumption can consequently be expected from those where round-trip times of 400 ms are used on all paths. Figure 6.14 illustrates the mobile node's available credit over the course of two such experiments. The top chart in the figure shows an experiment with Credit-Based Authorization running in Inbound mode, and Outbound mode was used in the experiment shown in the bottom chart. Both curves decrease when a handover takes place at the times marked by a vertical line, and as expected, the declines here are more pronounced than those in reactive mobility management. Relatively speaking, the declines in Outbound mode are more substantial than the ones in Inbound mode given a lower amount of credit available,

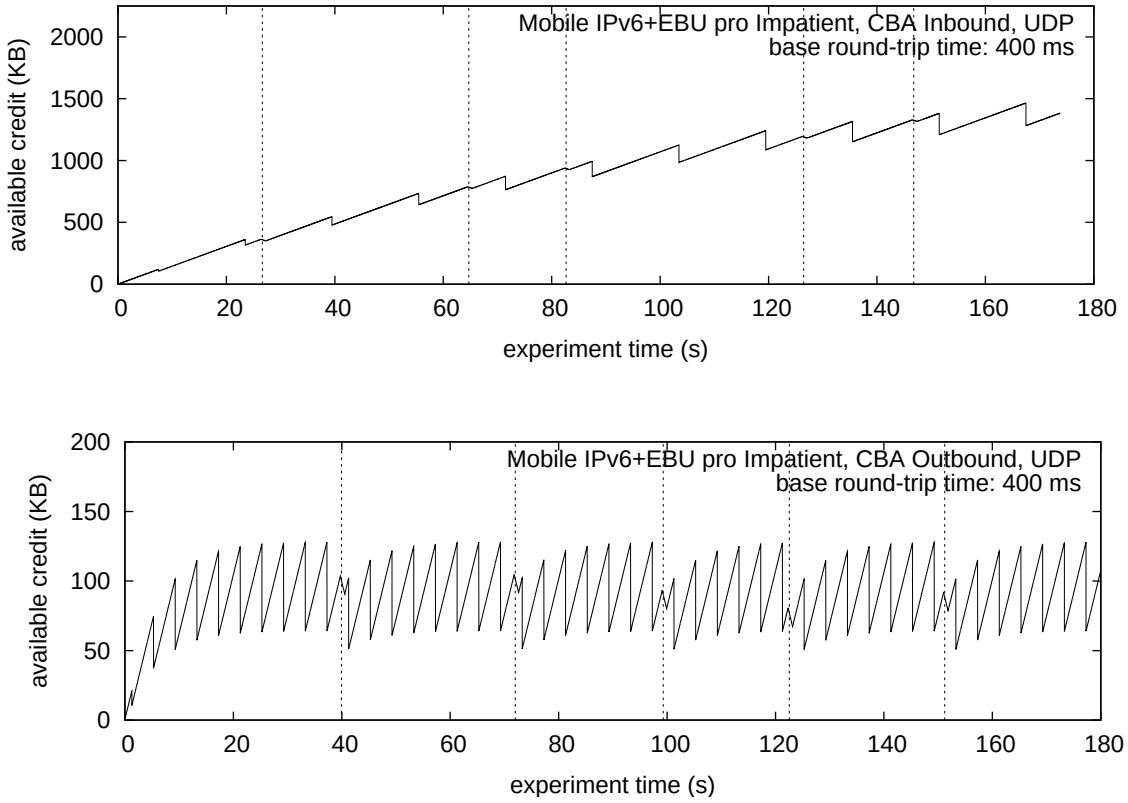



Figure 6.14: Credit availability in CBA Inbound mode for proactive mobility management

which in turn is due to the more rigid credit aging in Outbound mode. But neither in Inbound mode nor in Outbound mode does the handover-related credit consumption jeopardize the availability of credit during handover.

It should be emphasized that the amount of credit consumed during a handover is solely a function of the time that passes between the correspondent node's reception of the proactive early Binding Update message, sent by the mobile node from the old point of attachment, and the delivery of the mobile node's standard Binding Update message which completes reachability verification after the handover. On the other hand, differences between the latencies of proactive home and correspondent registrations do *not* further influence the amount of credit consumed. Credit consumption is therefore orthogonal to packet loss, which does increase with differences in home and correspondent registration latencies, but which may actually be zero in the presence of long, yet equal registration latencies.

Figure 6.15 corroborates the foregoing observation with charts of the mobile node's available credit in asymmetric network topologies when Credit-Based Authorization is operated in Outbound mode. The round-trip time on paths incident to the home agent is 25 ms in the top chart and 400 ms in the bottom chart, while the round-trip time between the mobile node and the correspondent node is 400 ms in the top chart and 25 ms in the bottom chart. The mobile node uses Home Priority mode, so the initiation time of the link layer handover is never synchronized with the redirection of the correspondent node's payload packets from the old to the new

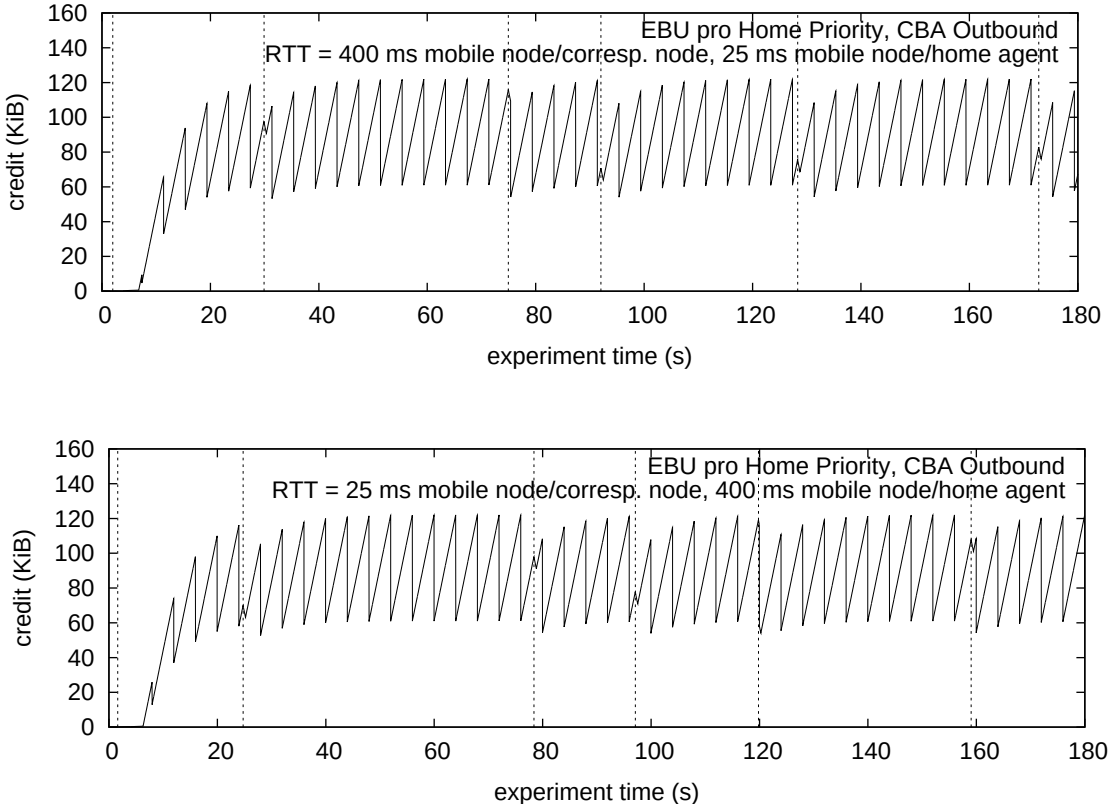



Figure 6.15: Credit availability in CBA Outbound mode for proactive mobility management

care-of address. While this causes increased packet loss compared to a symmetric network topology where all paths have a round-trip time of 400 ms, it actually reduces credit consumption: In the top chart, the mobile node initiates the link layer handover already when it receives the home agent's Binding Acknowledgment message, although it takes significantly longer for the correspondent node's Binding Acknowledgment message to arrive. This shortens the phase during which the new care-of address is in Unverified state, leading to reduced credit consumption during handovers at the cost of packet loss at the old care-of address. The higher home registration latency in the bottom chart causes more credit consumption before the mobile node initiates the link layer handover (in addition to increased packet loss at the new care-of address). But the comparably low round-trip time between the mobile node's new point of attachment and the correspondent node compensates this effect. The consumption of credit is the same when Credit-Based Authorization is operated in Inbound mode. Yet its relative impact is then lower considering the less rigid credit aging and, consequently, higher amount of available credit. A companion diagram for Inbound mode in addition to figure 6.15 is therefore omitted here.

## 6.4 TCP File Transfers

Although real-time applications like Internet telephony are expected to become significantly more prevalent in the short term, the majority of Internet applications is currently still based on TCP. Beyond the classic use of TCP for Web browsing, file

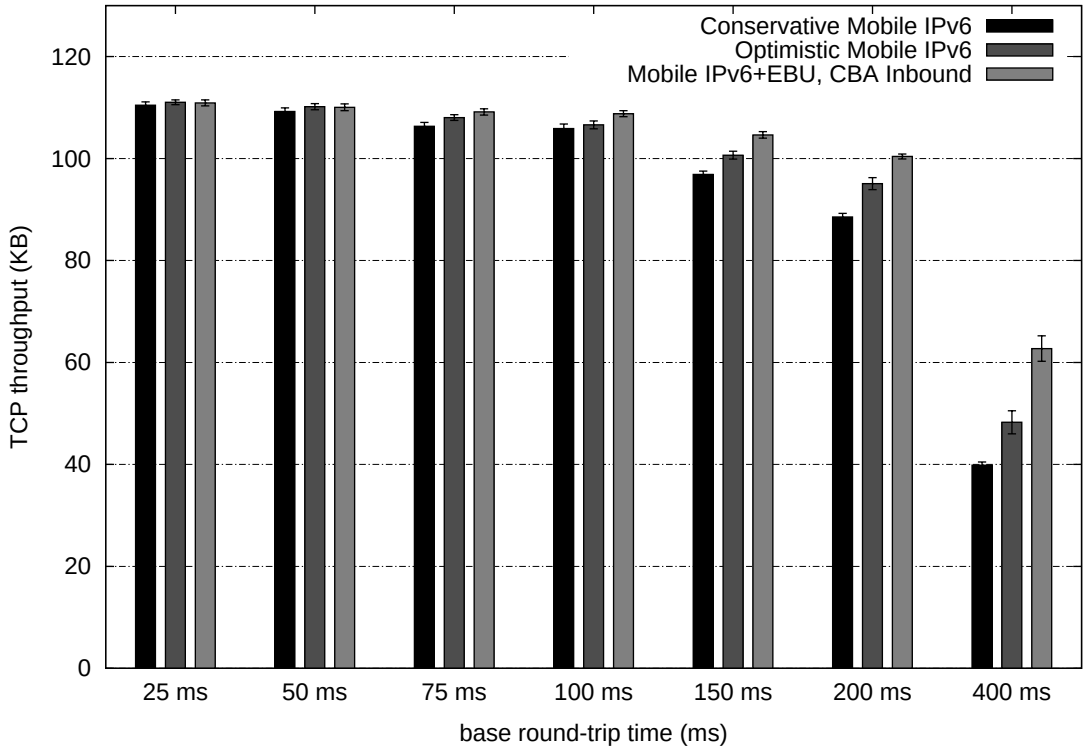



Figure 6.16: Mean payload throughput in TCP SACK file transfers

transfer, and email, the protocol has also been adopted for multi-media applications such as streaming, or even real-time Internet telephony. Modern TCP implementations incorporate a number of sophisticated congestion control and loss recovery mechanisms that are sensitive to IP connectivity changes at either end node. This section evaluates how mobile nodes can benefit from Early Binding Updates and Credit-Based Authorization when downloading a file from a stationary TCP server.

### 6.4.1 Throughput

Mobile users who conduct a file transfer over TCP are mostly interested in the speed of the download. Payload throughput in a mobile scenario is hence an appropriate metric to estimate the benefit of mobility management optimizations. Figure 6.16 compares the performance of TCP SACK over conservative Mobile IPv6, optimistic Mobile IPv6, and Mobile IPv6 with Early Binding Updates regarding the amount of data the Mobile IPv6 variants deliver during a 360-s file transfer that spans 5 handovers. The experiments were repeated for different symmetric network topologies with base round-trip times of between 25 ms and 400 ms on all paths. The figure shows averages and 95% confidence intervals from 20 experiments per Mobile IPv6 variant and network topology.

The measurements evidence a dependency of TCP throughput on the round-trip time across all Mobile IPv6 variants. There are mainly three reasons for this: First, Mobile IPv6 signaling latencies grow with the round-trip time and constitute an increasingly long period during which a file transfer must pause. Second, a longer round-trip time on the transmission path implies a higher bandwidth-delay product. Since TCP attempts to send as much data per round-trip time as fits into the

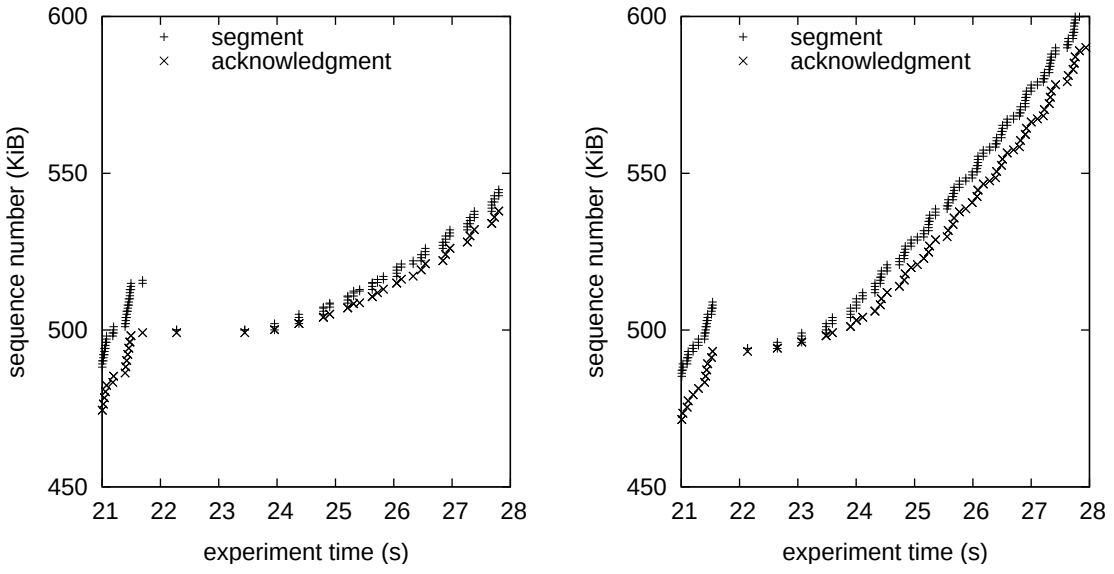
bandwidth-delay product, handover-related packet loss is higher on average when the round-trip time is long, and the time it takes TCP to repair this loss increases as well. Third, the speed at which TCP recovers from packet loss and adapts to the bandwidth available on the new transmission path, both in Slow Start mode and in Fast Recovery mode, is directly proportional to the path's round-trip time. The longer the round-trip time, the more sluggish TCP consequently reacts to a handover.

On the other hand, the measurements also show that the impact of the round-trip time on TCP throughput is lowest for Mobile IPv6 with Early Binding Updates. What is striking is that Early Binding Updates can more than double the throughput compared to conservative Mobile IPv6 when the base round-trip time on the transmission path is 400 ms. The deployment of optimistic Mobile IPv6 in replacement for conservative Mobile IPv6 still leads to a performance gain of more than one third. The benefit of optimistic Mobile IPv6 and Mobile IPv6 with Early Binding Updates decreases notably as round-trip times become smaller. None of the optimizations provides any substantial improvement with round-trip times of 100 ms and below. What makes the optimizations perform so much better over long distances, and what causes the benefit to vanish completely when round-trip times are small? The answers to these questions are found in TCP's loss-recovery mechanisms, as explained next.

### 6.4.2 Retransmission Timeouts

Of the two mechanisms through which TCP recovers from packet loss—that is, Slow Start mode after a retransmission timeout or Fast Recovery mode after a fast retransmit—, it is typically the combination of a retransmission timeout and Slow Start mode which a handover triggers in case mobility is managed reactively. A fast retransmit would require three duplicate acknowledgments to be generated by the mobile node and hence three segments to be delivered to the new care-of address. This seldom happens in reactive mobility management for the following reason: When the mobile node changes IP connectivity, all packets currently in flight to the old care-of address are lost, and an additional one-way worth of packets is lost until the mobile node's Binding Update message is delivered to the correspondent node. The total loss is worth a round-trip time between the correspondent node and the mobile node, and therefore roughly corresponds to the correspondent node's available window. The correspondent node cannot send more segments than permitted by the available window without receiving an acknowledgment. It consequently stalls and runs into a retransmission timeout.

The number of successive retransmission timeouts that the correspondent node goes through during a handover depends on when the correspondent node receives the Binding Update message from the mobile node. If the Binding Update message arrives before the retransmission timer expires for the first time, the correspondent node resends the lost segments the new care-of address. This reestablishes the data transfer. However, if the Binding Update message arrives later than the first retransmission timeout, the correspondent node directs the lost segments to the old care-of address and times out yet again. The timeout period doubles for each successive retransmission, up to an implementation-specific limit which must be 60 s




(a) Two retransmission timeouts (b) Single retransmission timeout

Figure 6.17: Impact of the retransmission timeout count on TCP SACK throughput

at least.[5] Two successive retransmission timeouts are hence three times as long as a single retransmission timeout. What determines the delay that TCP traffic is subjected to during a handover is consequently this overall timeout period at the TCP layer rather than Mobile IPv6 signaling latency alone.

The incidence of two successive retransmission timeouts as a result of a handover has an adverse impact on TCP's adaptivity to the bandwidth available on the new transmission path. Specifically, TCP does not exponentially ramp up its transmission rate in Slow Start mode after the second retransmission timeout as it usually does after a single timeout. Instead, TCP switches to Congestion Avoidance mode within a single round-trip time, accelerating transmission by only one segment per round-trip time. This happens because, after each retransmission timeout, the congestion window is reduced to a single segment size, and the slow-start threshold is set to half of the current amount of outstanding, unacknowledged data. With this configuration, the amount of outstanding data is just a single segment when the retransmission timer expires the second time. The slow-start threshold is then set to its minimum value of two segments and effectively causes the correspondent node to operate in congestion avoidance mode from the very beginning. The typical accelerated growth of the congestion window in Slow Start mode is thus inhibited. Figure 6.17 illustrates this effect with two exemplifying TCP SACK traces from a scenario with 400-ms base round-trip times. The traces show the sequence numbers of transmitted segments and received acknowledgments at the correspondent node when TCP operates over conservative Mobile IPv6 and Mobile IPv6 with Early Binding Updates, respectively. Link layer handover delays on the mobile-node side are zero, and the DNA protocol is used for router discovery and movement detection. The figure clearly shows the more cumbersome throughput acceleration after

[5]The FreeBSD operating system uses a maximum timeout period of 64 s.

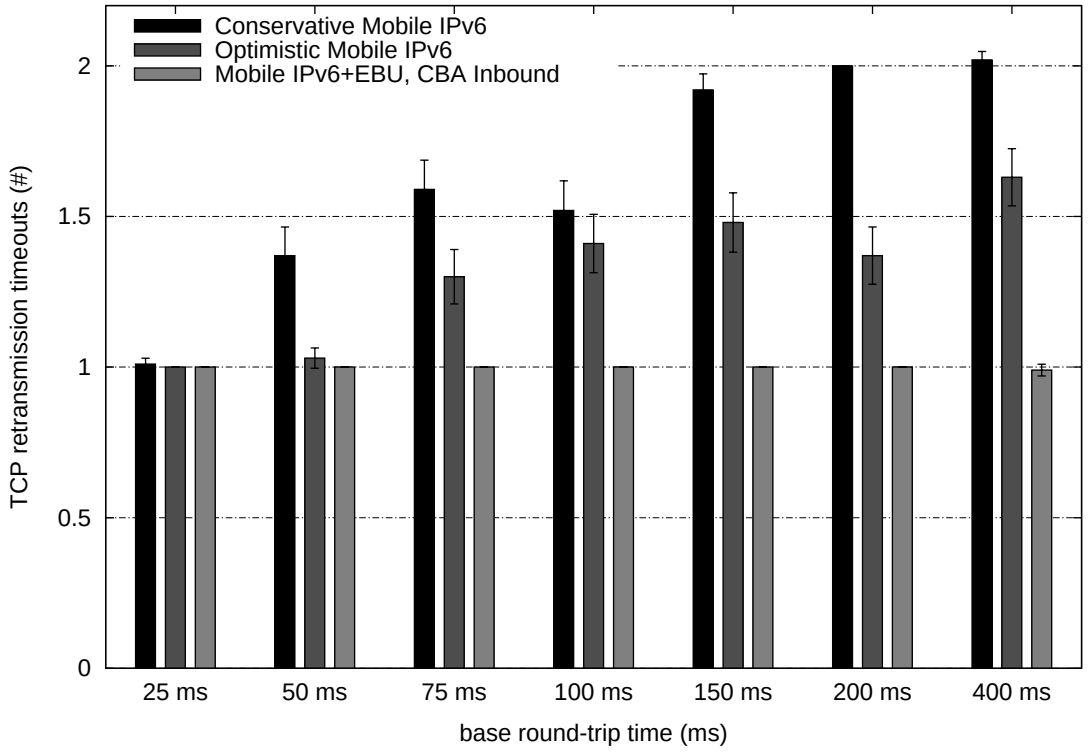
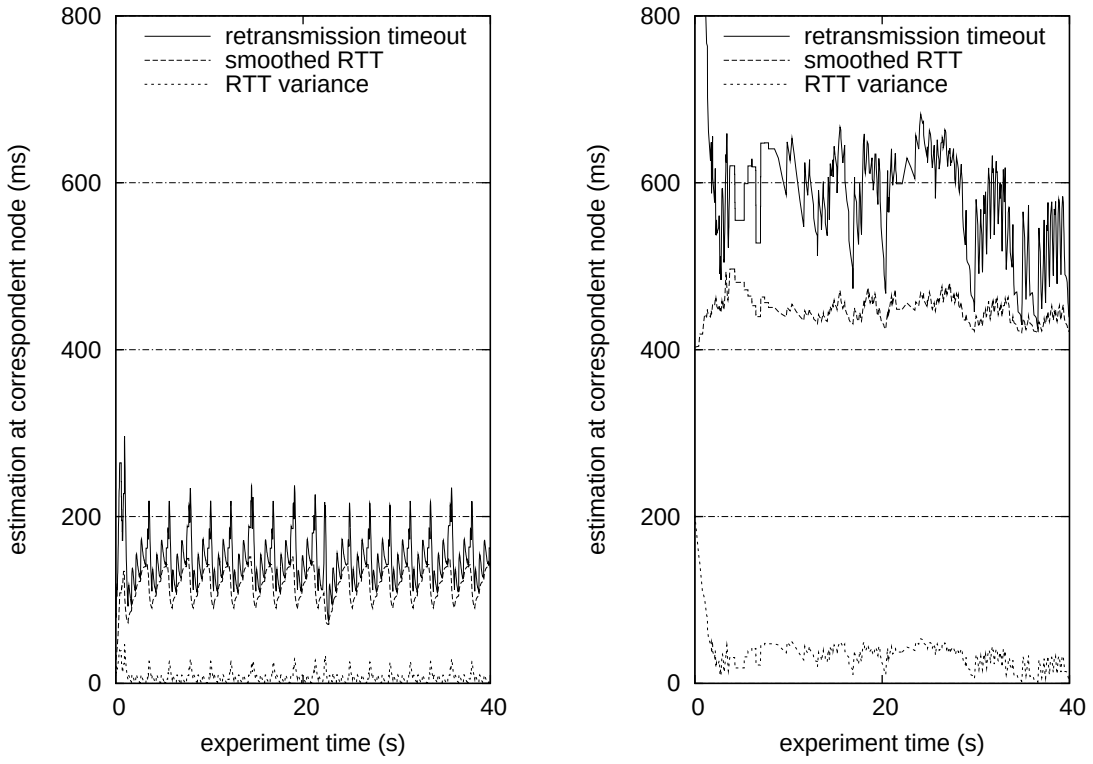



Figure 6.18: Mean retransmission timeout counts during handover in TCP SACK

two successive retransmission timeouts compared to the typical exponential growth after a single retransmission timeout.

These findings highlight the grave impact that repeated retransmission timeouts may have on the performance of TCP, and they demand a closer investigation into the actual number of retransmission timeouts that TCP SACK goes through after a handover in the experiments discussed in section 6.4.1. Figure 6.18 displays the averages and 95% confidence intervals of the retransmission timeout counts in those experiments. Given TCP's dynamic adaptation of the retransmission timeout period to the actual round-trip time on the transmission path, one may expect that the retransmission timeout counts are invariant with respect to the actual round-trip time on the transmission path, and that they only depend on the Mobile IPv6 variant in use beneath TCP. However, the actual measurements indicate that there is a dependency on the round-trip time when TCP operates over conservative or optimistic Mobile IPv6. The number of retransmission timeouts then grows from one to two as the base round-trip time increases from 25 ms to 400 ms. Only Mobile IPv6 with Early Binding Updates keeps the retranmission timeout count stable at one across all topologies.

The reason why the round-trip time may impact the number of retransmission timeouts in the wake of a handover can be found in forward buffering delays of congested access routers. Access routers in the testbed have forward buffering space worth 125 ms of the link bandwidth and may hence delay traffic up to 125 ms in times of congestion. Periodic congestion, in turn, is an immediate consequence of TCP's saw-tooth-like congestion window dimensioning, which steadily probes the network for additional capacity and eventually fills the forward buffer space in a bottleneck




(a) RTT = 25 ms (b) RTT = 400 ms

Figure 6.19: Retransmission timeout estimation at the correspondent node in TCP SACK

router. Forward buffering delays on the old transmission path increase the actual round-trip time on this path to beyond the base round-trip time and thereby enlarge TCP's estimated retransmission timeout period. As the size of a forward buffer is constant, the effect of forward buffering delays is relatively stronger the smaller the base round-trip time is, and it may inflate TCP's round-trip time estimates to several multiples of the base round-trip time. On the other hand, Mobile IPv6 messages are not notably affected by forward buffering delays in reactive mobility management where the messages are exchanged via a new path which TCP did not yet congest. Forward buffering delays hence increase the probability for Mobile IPv6 signaling to complete within one retransmission timeout period, especially when the base round-trip time is small. This leads to the lower retransmission timeout counts for conservative and optimistic Mobile IPv6 in network topologies with small base round-trip times.

The two traces in figure 6.19 illustrate the impact that router buffering delays have on the correspondent node's estimated retransmission timeout period in experiments with TCP SACK and a base round-trip time on the transmission path of 25 ms and 400 ms, respectively. The traces show the smoothed round-trip time estimate, the variation in round-trip time measurements, and the derived retransmission timeout period, but they do not reflect any back-offs in the retransmission timeout. The figure clearly demonstrates that the retransmission timeout period almost always exceeds the base round-trip time by a factor of four or higher when the base round-trip time is small. In contrast, in the scenario with a 400-ms base round-trip time, the

retransmission timeout period is usually below 700 ms. The signaling of conservative Mobile IPv6 is then long enough to always force TCP into a second retransmission timeout. A third retransmission timeout is generally not required given that the second timeout is already twice as long as the first.

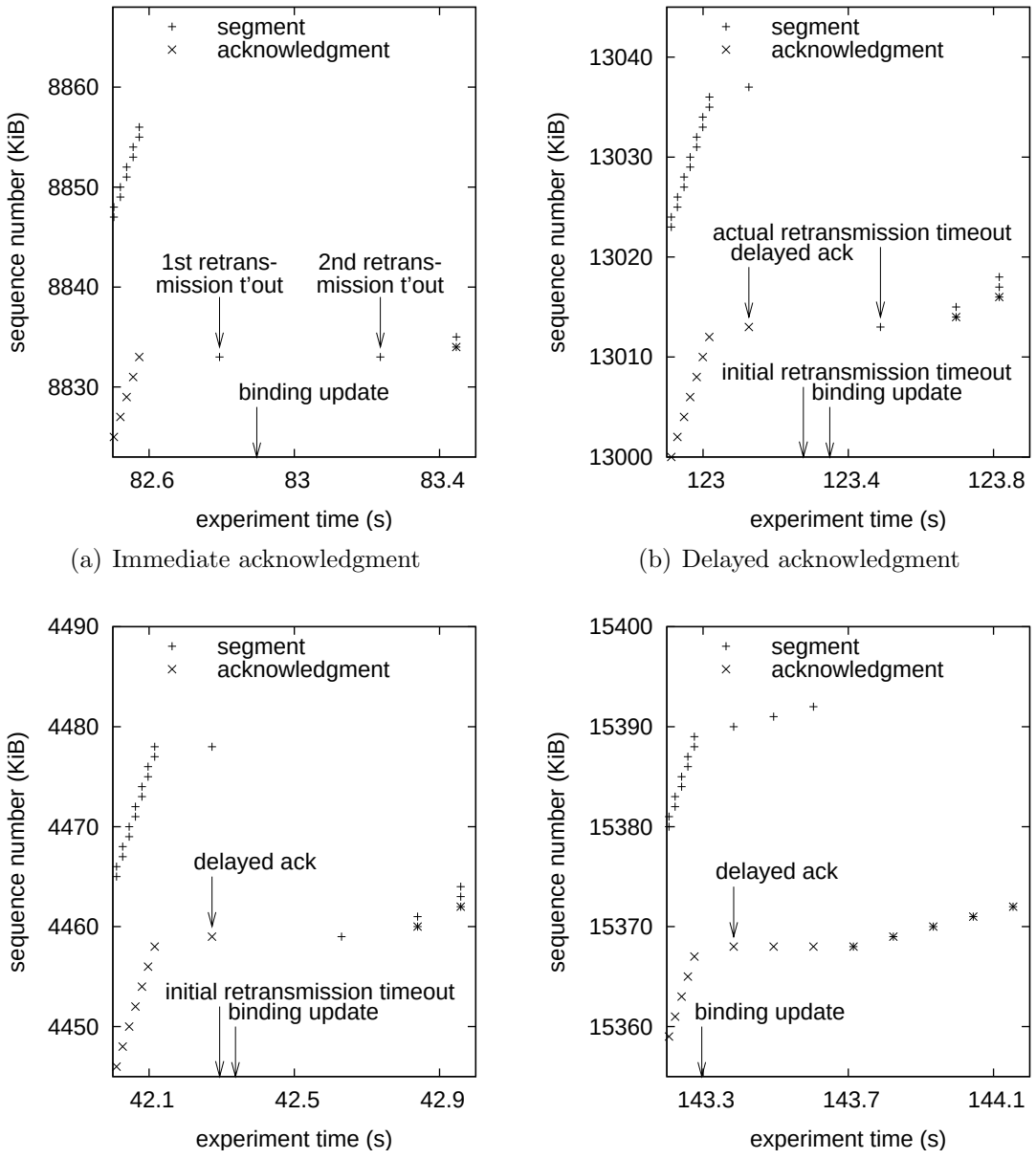
The advantage of Mobile IPv6 with Early Binding Updates compared to conservative and optimistic Mobile IPv6 is that it *always* updates the binding at the correspondent node before the correspondent node's retransmission timer expires for the first time. A second, backed-off retransmission timeout can then be avoided. The expedited binding update hence helps to quickly resume the data transfer and also aids an accelerated adaptation of TCP's transmission rate to the capacity of the new transmission path as the foregoing discussion has shown.

### 6.4.3 Effect of Delayed Acknowledgments

TCP implementations with support for delayed acknowledgments attempt to piggyback acknowledgments for received data onto outgoing segments. A due acknowledgment may for this purpose be delayed until either some maximum delay is reached—which is 100 ms by default in FreeBSD—, or another segment arrives so that two segments can be cumulatively confirmed with a single acknowledgment. The mobile node in the conducted file transfer experiments supports delayed acknowledgments. But since data flows unidirectionally from the correspondent node to the mobile node, the local application on the mobile-node side never delivers any data onto which an acknowledgment could be piggybacked. The mobile node consequently ends up generating an acknowledgment for every other segment that it receives from the correspondent node, or after a delay of 100 ms if no second segment arrives. Chances are in general fifty-fifty that the last segment delivered to the mobile node before a handover triggers a delayed acknowledgment: If the second-to-last segment has already been acknowledged, the mobile node transmits an immediate cumulative acknowledgment once the last segment arrives. Otherwise, the last acknowledgment is delayed.

The effect of a delayed acknowledgment during handover can be momentous: Since the correspondent node resets its retransmission timer whenever it receives an acknowledgment that covers new data, the late arrival of a delayed acknowledgment defers the expiration of the retransmission timer, and it may thus reduce the number of consecutive retransmission timeouts that TCP goes through until communications finally resume via the mobile node's new care-of address. Figure 6.20 visualizes this effect with traces from TCP SACK experiments conducted in a symmetric network topology with base round-trip times of 100 ms. The use of conservative Mobile IPv6 in figure 6.20(a) is responsible for the premature expiration of the retransmission timer and forces the correspondent node into a second retransmission timeout. The retransmission timer is initially scheduled to expire prior to the binding update also in figure 6.20(b), which again shows a trace for conservative Mobile IPv6. But in this case, the late arrival of a delayed acknowledgment causes the correspondent node to reschedule the retransmission timeout to a time after the binding update. The second retransmission timeout is thereby avoided, and communications resume already with the first retransmission timeout.

The exact interval by which the retransmission timeout is postponed on the receipt of a delayed acknowledgment depends on the arrival times of the delayed acknowledgment and the immediately preceding undelayed acknowledgment, as well as on




(a) Immediate acknowledgment

(b) Delayed acknowledgment

(c) Delayed acknowledgment via home agent

(d) Limited Transmit algorithm

Figure 6.20: Impact of delayed acknowledgment on handover performance in TCP SACK

the correspondent node's retransmission timeout intervals at those times. Let the retransmission timeout interval at time $t$ be $rto(t)$. If, according to the illustration in figure 6.21, the correspondent node receives the undelayed acknowledgment at time $T_1$, then the retransmission timer is initially set to $T_1 + rto(T_1)$. This timeout gets amended when the correspondent node receives the delayed acknowledgment, say, at time $T_2$. The retransmission timer is then rescheduled to expire at time $T_2 + rto(T_2)$. Note that the correspondent node's estimated round-trip time may change between $T_1$ and $T_2$, hence $rto(T_2)$ may be different than $rto(T_1)$. The delayed acknowledgment reduces the number of retransmission timeouts from two to one if and only if the binding update completes within the period between time

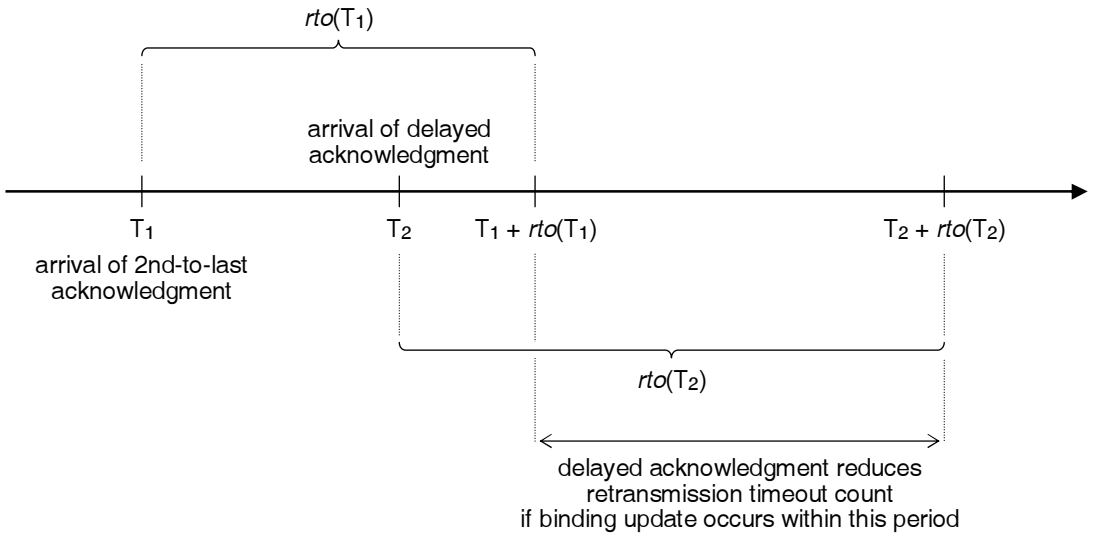



Figure 6.21: Reduced retransmission timeout counts due to delayed acknowledgment

$T_1 + rto(T_1)$ and time $T_2 + rto(T_2)$. Communications always resume with the first retransmission timeout if the binding update completes earlier than this. The delayed acknowledgment then does not change the number of retransmission timeouts that TCP goes through, although it does increase the time until the retransmission timer expires for the first time. Similarly, at least two retransmission timeouts are required irrespective of the delayed acknowledgment if the binding update completes later than time $T_2 + rto(T_2)$. TCP's continuous round-trip time measurements incorporate delayed acknowledgments, so the retransmission timer seldom expires before a delayed acknowledgment arrives. The equation $T_2 < T_1 + rto(T_1)$, which figure 6.21 satisfies, consequently holds in in general.

Delayed acknowledgments that reduce the number of retransmission timeouts from two to one lessen handover delay and provide for a faster throughput adaptation to the available path bandwidth afterwards, as explained in section 6.4.2. The downside of delayed acknowledgments is that they increase the handover delay whenever they fail to change the number of consecutive retransmission timeouts.

The arrival of a delayed acknowledgment at the correspondent node is further deferred when the acknowledgment is routed via the home agent. This may happen with conservative or optimistic Mobile IPv6, where the mobile node cannot update the binding at the correspondent node before the respective home registration is complete. Since the correspondent node would discard route-optimized packets from the mobile node's new care-of address until its binding is up to date, the mobile node may either drop these packets by itself, or temporarily switch to reverse tunneling until the correspondent node has been informed about the new care-of address. Both the conservative and the optimistic Mobile IPv6 implementations that were used in the experiments pursue reverse tunneling while the correspondent node's binding for the mobile node is out of date. The conservative Mobile IPv6 implementation switches to route optimization once the mobile node has received a Binding Acknowledgment message from the correspondent node, whereas the optimistic Mobile IPv6 implementation switches with the transmission of a Binding Update message. A delayed acknowledgment may therefore be routed via the home

agent and thus arrive at the correspondent node with an even larger delay than it would had it been route-optimized. The trace in figure 6.20(c) shows such a situation for TCP SACK over optimistic Mobile IPv6: The mobile node transmits the delayed acknowledgment after it has sent a Binding Update message to the home agent, but before it could send a Binding Update message to the correspondent node. The mobile node therefore directs the delayed acknowledgment via the home agent, increasing the propagation latency of the acknowledgment to about twice as much as it would normally be. In the particular case shown in figure 6.20(c), the correspondent node also updates its round-trip time estimate when it receives the delayed acknowledgment, leading to a higher derived retransmission timeout period. In fact, the retransmission timeout period increases from 179 ms to 357 ms at the time the delayed acknowledgment is processed by the correspondent node. The consequence is that the time between the arrival of the delayed acknowledgment and the actual expiration of the retransmission timer is longer than the initial retransmission timeout period that was scheduled when the last acknowledgment preceding the delayed acknowledgment was received.

Delays can further enable the correspondent node to recover from handover-related packet loss without a retransmission timeout at all in case the correspondent node supports the Limited Transmit algorithm. Since duplicate acknowledgments must be transmitted immediately, delayed acknowledgments by definition cover new data and hence permit the correspondent node to transmit one further segment. This additional segment typically arrives at the mobile node out of order due to previous, handover-related packet loss, and it consequently triggers a duplicate acknowledgment. The Limited Transmit algorithm then enables the correspondent node to send another segment on the receipt of the duplicate acknowledgment. This segment again triggers a duplicate acknowledgment on the mobile-node side, which in turn elicits the transmission of another segment. The correspondent node thus eventually receives three duplicate acknowledgments, performs a fast retransmit, and enters Fast Recovery mode. Provided that it supports TCP NewReno or TCP SACK, the correspondent node can then recover from handover-related packet loss without any retransmission timeouts. The trace in figure 6.20(d) captures this procedure from an experiment with TCP SACK, Mobile IPv6 with Early Binding Updates, and base round-trip times of 100 ms.

The segment that the correspondent node sends after receiving the delayed acknowledgment has no effect if the correspondent node does not support the Limited Transmit algorithm because the single duplicate acknowledgment that the mobile node sends in response to this segment then does not solicit another segment from the correspondent node. Instead, the packet exchange stalls and the correspondent node eventually falls into a retransmission timeout. Moreover, if the correspondent node runs TCP Reno and more than one segment is lost during the handover, the correspondent node eventually falls into a retransmission timeout despite the use of the Limited Transmit algorithm. This is because TCP Reno fails to retransmit more than a single lost segment while in Fast Recovery mode.

The performance of loss recovery in Fast Recovery mode strongly depends on the TCP implementation. TCP NewReno is limited to retransmit only a single lost segment per round-trip time. Loss recovery may then consume considerable time because handover-related packet loss in reactive mobility management usually spans

an entire window worth of data. Since TCP approximates its congestion window to the bandwidth-delay product of the transmission path, the recovery latency is especially long if the round-trip time on the old the transmission path is high. TCP Selective Acknowledgment options communicate the information that the correspondent node needs to recover more efficiently, but the actual recovery performance depends on the correspondent node's TCP SACK implementation. The network stack in FreeBSD retransmits one lost segment per arriving acknowledgment. This allows for expedited loss recovery when multiple segments escape packet loss and generate acknowledgments. But the loss of an entire window of data during handover leaves the correspondent node with either no acknowledgment or just a single delayed acknowledgment. The amount of retransmitted data then does not increase beyond a single segment per round-trip time while the correspondent node is in Fast Recovery mode. The recovery performance of TCP SACK is in this case no better than that of TCP NewReno, as shown in figure 6.20(d) for Mobile IPv6 with Early Binding Updates. The sluggish recovery of TCP NewReno and possibly also TCP SACK is responsible for a significant performance degradation compared to a recovery in Slow Start mode. In fact, the overall performance is higher without any Mobile IPv6 optimizations since TCP then recovers more efficiently in Slow Start mode.

### 6.4.4 Handover Delay

TCP is a reliable transport protocol that guarantees to deliver data in sequence to a receiving application. A mobile node receiving new data after a series of handover-related packet losses may therefore be unable to pass this data on to the application due missing preceding data. This may happen if the correspondent node's available window is large enough so that some of the segments from the window are redirected to the new care-of address. Previous packet loss then renders these segments out-of-order. Data transfer resumes only when the correspondent node sends a segment that arrives *in order* at the mobile node. Such a segment is either triggered by a retransmission timeout at the correspondent node, or it is a fast retransmit after the arrival of three consecutive duplicate acknowledgments.

Changes in IP connectivity on the mobile-node side may consequently disrupt TCP connections due to both packet loss and the subsequent arrival of out-of-order data at the mobile node. Yet the handover phase as defined in section 6.2.2 for the foregoing Internet telephony evaluation covers only packet loss since it is tailored to unreliable UDP. This calls for a refinement of what the handover phase is with respect to a TCP connection: The *TCP handover phase* is defined as the period between the correspondent node's transmission of the segment that immediately follows the last segment successfully received by the mobile node at the old care-of address up to the point at which the correspondent node transmits the first segment that the mobile node again successfully receives at the new care-of address and that causes new data to be delivered to the application. The *TCP handover delay* is accordingly defined as the duration of the TCP handover phase. In the absence of out-of-order segments, and in the special case where no packet loss occurs during a handover at all, these definitions are equivalent to the previous ones. "TCP handover phase" and "TCP handover delay" may be referred to simply as "handover phase" and "handover delay", respectively, provided that the context resolves ambiguity.

It should be noted that, in the rare event that the segment immediately preceding the first handover-related packet loss is dropped by an access router due to a forward

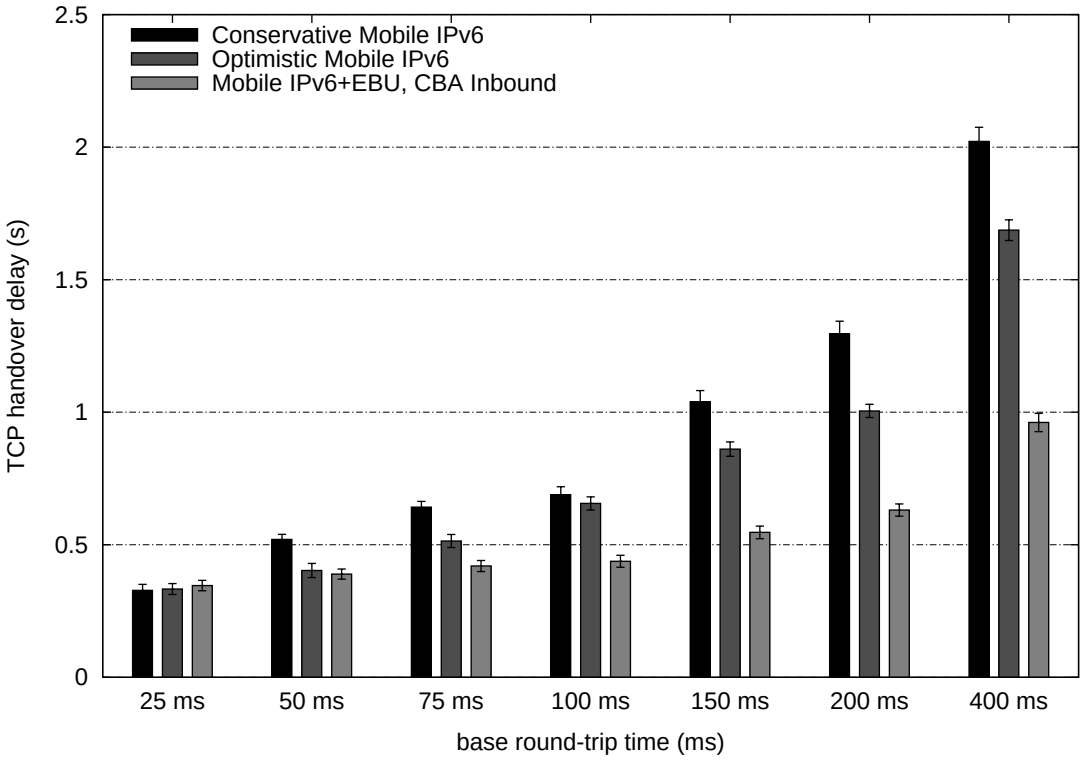



Figure 6.22: Mean handover delays in TCP SACK

buffer overflow, the original transmission and the retransmission of the dropped segment are considered the beginning and the end of the handover phase, respectively. The handover phase then includes handover-*unrelated* packet loss, causing the handover delay that is due to only handover-related packet loss to be overestimated. The error is generally close to the inter-packet arrival time, $I_{TCP}$, for transmitting packets of 1108 B across a testbed path with an available bandwidth of 1024 kbps:

$$I_{TCP} = \frac{1108\,\mathrm{B}}{1024\,\mathrm{kbps}} = \frac{1108\,\mathrm{B}}{128\,000\,\mathrm{Bps}} = 8.7\,\mathrm{ms}$$

However, the probability of a packet to get dropped due to congestion on the transmission path has been found to be between 0.1% and 0.3% in all experiments. The probability that the handover delay is incorrectly determined is hence very low.

Figure 6.22 compares the handover delays of TCP SACK observed in experiments with conservative Mobile IPv6, optimistic Mobile IPv6, and Mobile IPv6 with Early Binding Updates. The conservative mobile node was configured to wait for a Binding Acknowledgment message from the correspondent node before route-optimizing any payload packets from a new care-of address, whereas the optimistic mobile node resumes route optimization right after sending an early Binding Update message to the correspondent node. The experiments were conducted with a symmetric network topology using base round-trip times of between 25 ms and 400 ms. The figure shows averages and 95% confidence intervals from 100 handovers per Mobile IPv6 variant and network topology. The measurements verify an expected dependency of the handover delay on the round-trip times across all three Mobile IPv6 variants,

and this dependency is highest for conservative Mobile IPv6 and lowest for Mobile IPv6 with Early Binding Updates. This is due to the impact the round-trip time and the Mobile IPv6 variant have on the duration and number of retransmission timeouts that TCP goes through on the correspondent node side, as discussed in sections 6.4.2 and 6.4.3. Round-trip times as small as 25 ms enable all three Mobile IPv6 variants to complete a correspondent registration before the retransmission timer expires for the first time. This leads to very similar handover delays amongst the Mobile IPv6 variants. The somewhat higher handover delay of conservative Mobile IPv6 is due to a different routing of delayed acknowledgments: Both optimistic Mobile IPv6 and Mobile IPv6 with Early Binding Updates transmit a Binding Update message to the correspondent node before the 100-ms hold time for delayed acknowledgments elapses at the transport layer. Any delayed acknowledgment is therefore route-optimized. In contrast, a conservative mobile node always sends a delayed acknowledgment via its home agent, because it does not perform route optimization before it receives a Binding Acknowledgment message from the correspondent node, and this happens only shortly after the delayed acknowledgment has been dispatched. The deferring impact on the correspondent node's retransmission timeout, which a delayed acknowledgment that goes through the home agent has, is stronger than that of a route-optimized delayed acknowledgment. This leads to the minor disadvantage of conservative Mobile IPv6 relative to the other two Mobile IPv6 variants when round-trip times are low.

With base round-trip times of 50 ms and beyond, optimistic Mobile IPv6 can no longer complete the return-routability procedure within the 100-ms hold time for delayed acknowledgments, so it then reverse-tunnels any delayed acknowledgments as well. Mobile IPv6 with Early Binding Updates route-optimizes delayed acknowledgments irrespective of the base round-trip time since it does not need to wait for the return routability procedure.

The performance benefit of optimistic Mobile IPv6 and Mobile IPv6 with Early Binding Updates increases as round-trip times grow to 50 ms and beyond. The higher handover delay of conservative Mobile IPv6 is then due to a higher average number of consecutive retransmission timeouts, as visualized in figure 6.18. In these scenarios, optimistic Mobile IPv6 can no longer complete the return-routability procedure and send a Binding Update message to the correspondent node within the 100-ms hold time for delayed acknowledgments, so it reverse-tunnels any delayed acknowledgment just like conservative Mobile IPv6. On the other hand, Mobile IPv6 with Early Binding Updates route-optimizes delayed acknowledgments irrespective of the base round-trip time since it does not need to wait for the return routability procedure. Route-optimized delayed acknowledgments are the primary reason for the better performance of Mobile IPv6 with Early Binding Updates compared to optimistic Mobile IPv6 in the experiments with base round-trip times of 50 ms or 75 ms. They continue to be a contributing factor also in the scenarios with higher base round-trip times, but the primary reason for the better performance of Mobile IPv6 with Early Binding Updates is then a lower average number of retransmission timeouts, as shown in figure 6.18.

### 6.4.5 Packet Loss

Subsequent to the handover phase, TCP sets about retransmitting the segments that were lost during the handover. Handover-related packet loss in reactive, end-to-end

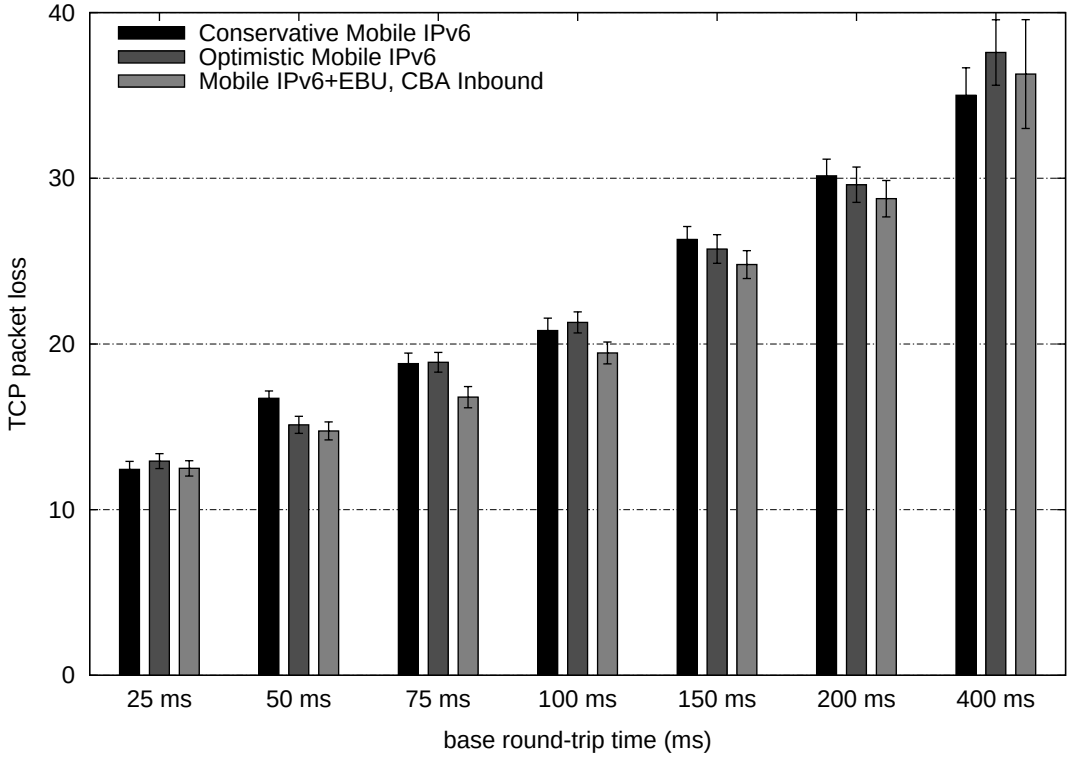



Figure 6.23: Mean packet loss in TCP SACK file transfers

mobility management is relatively stable across different Mobile IPv6 variants. The latency of a binding update is always long enough to cause the minimum loss of a full window worth of data, and any additional packet loss is solely due to vain retransmissions, which the correspondent node may pursue prior to receiving the binding update. Figure 6.23 shows the average number of handover-related packet losses along with the respective 95% confidence intervals in TCP SACK experiments with conservative Mobile IPv6, optimistic Mobile IPv6, and Mobile IPv6 with Early Binding Updates. The experiments were conducted in symmetric network topologies with base round-trip times of between 25 ms and 400 ms. The values in the figure are based on 100 handovers per Mobile IPv6 variant and topology.

The relationship between the number of segments lost during a handover and the correspondent node's available window right before the handover gets corroborated in figure 6.24. This shows averages and 95% confidence intervals of the correspondent node's congestion window size in terms of the maximum TCP segment size (MSS) at the time the correspondent node sends the last segment to the mobile node's old care-of address, hence right before packet loss is detected. The congestion window usually defines the correspondent node's available window in these experiments because it is usually smaller than the send and receive windows. As expected, the measured congestion window sizes are roughly consistent with the mean packet loss counts across all topologies, indicating that about one window worth of data is lost during handover.

The correspondent node's congestion window differs from the available window only in the rare event of handover-unrelated packet loss shortly before a handover. Operation in Fast Recovery mode requires the correspondent node to buffer two windows

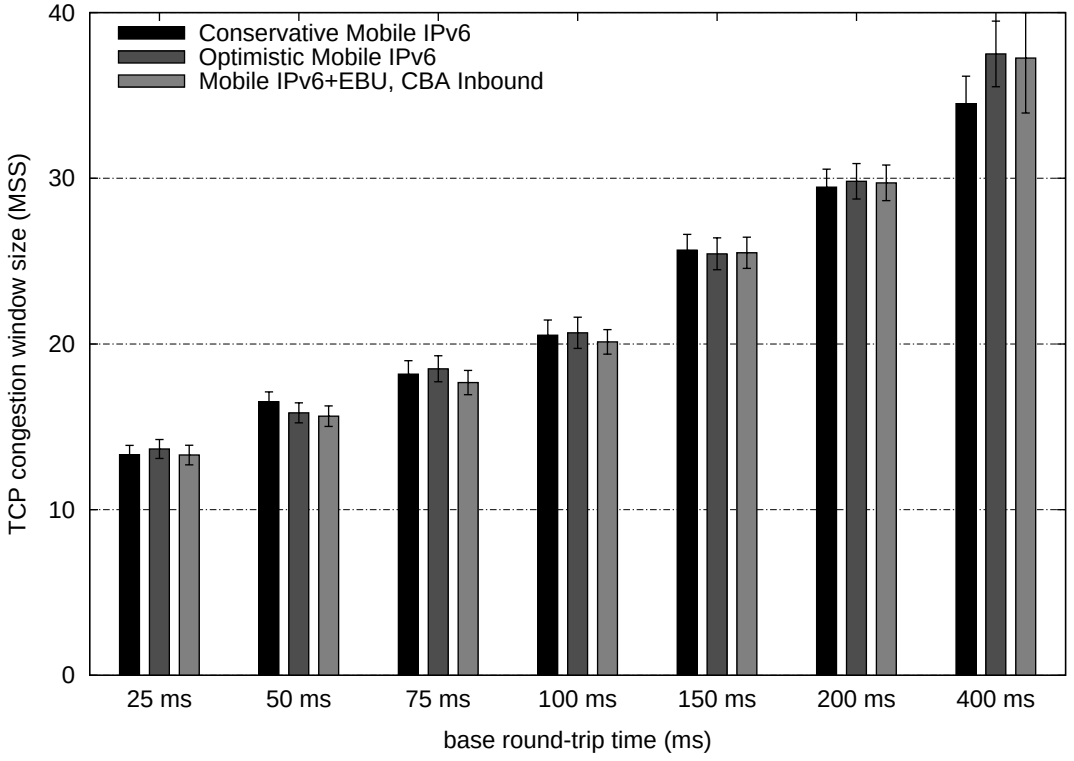



Figure 6.24: Mean TCP congestion window size right before handover

worth of data while the loss is being retransmitted and acknowledged, loading the send buffer to a level which may narrow the available window to less than the congestion window. At the same time, the mobile node must hold one window worth of data until it receives the retransmitted segment that fills the gap in the receive buffer. Depending on the size of the receive buffer, the mobile node may hence limit the correspondent node's available window through a small advertised window. The congestion window, in turn, may inflate to a very large size in Fast Recovery mode, and it may easily grow beyond the available window. However, as explained in section 6.1.7, the probability for handover-unrelated packet loss is small, and so is the likelihood for the available window to be less than the congestion window at the time the handover phase begins.

The are multiple reasons for the conceivable small differences between the congestion window size and the actual packet loss that can be found by comparing figures 6.23 and 6.24. First of all, the congestion window grows in steps smaller than one maximum segment size when the correspondent node is in Congestion Avoidance mode. The congestion window therefore typically exceeds the amount of outstanding data by some fraction of the maximum segment size. Packet loss, however, primarily depends on the amount of outstanding data and therefore differs from the congestion window size. Second, when Mobile IPv6 with Early Binding Updates is used beneath TCP, the earlier binding update at the correspondent node causes the correspondent node to redirect one, sometimes two segments to the new care-of address after the bulk of the window has been sent to the old care-of address. These segments escape packet loss and therefore lead to a smaller packet loss than the available window. This happens mostly due to a delayed acknowledgment, which causes the correspon-

dent node to transmit the last segment after the deferred delivery of the delayed acknowledgment, and hence with an increased probability to the mobile node's new care-of address. Delayed acknowledgments may have the same effect with optimistic Mobile IPv6, provided the involved round-trip times are sufficiently small so that the binding update at the correspondent node happens before the delayed acknowledgment arrives. Such occurrences are limited to smaller round-trip times, however, since the constant deferment of the delayed acknowledgment is higher relative to the required Mobile IPv6 signaling latency then. Packet loss beyond one window worth of data may be due to the occurrence of more than one retransmission timeout during handover. Third, on each premature timeout, TCP retransmits one segment to the old care-of address in vain. The additional packet loss adds to the window worth of data that was already lost previously.

The increased confidence intervals for the experiments with a base round-trip time of 400 ms are due to larger deviations in the correspondent node's congestion window across different experiments. In normal operation, the congestion window grows beyond the bandwidth-delay product of the transmission path until the forward buffer in a bottleneck router overflows, and it is then cut in half. This saw tooth is larger when the base round-trip time, hence the bandwidth-delay product is high. A handover may happen independently of the current congestion window size, so the longer the base round-trip time is the stronger gets the deviation in the congestion window size at the time of a handover.

### 6.4.6 Credit Availability

The amount of credit that a mobile node has available at any given point in time is less apparent for TCP-based file transfers than it is for UDP-based Internet telephony applications due to TCP's congestion avoidance mechanisms. The saw-tooth behavior in throughput that goes along with TCP's additive-increase, multiple-decrease congestion window sizing policy may lead to a low credit acquisition rate prior to handover and, consequently, to limited credit during the handover. Handover-unrelated packet loss shortly before a handover may further contribute to reduced credit acquisition due to the contraction of the congestion window that follows it. Moreover, the unidirectionality of the file transfers from the correspondent node to the mobile node implies that the mobile node sends at an average rate that is much lower than that of the correspondent node. The time it takes to consume a certain amount of credit during handover is therefore less than the time it takes to acquire the same amount of credit.

Figure 6.25 shows the amount of credit the mobile node accumulates over time in experiments where Credit-Based Authorization is operated in Inbound mode. The experiments in both charts where conducted in a symmetric network topology, whereby the base round-trip time was set to 25 ms in the top chart and to 400 ms in the bottom chart. Credit is aged by 1/8 every 16 s according to the default aging parameters defined in section 4.2.4. Both charts evidence that, despite the periodic aging, the credit continually grows until it reaches a point at which the new credit earned within a period of one crediting interval equals the amount of credit subtracted due to aging. The upper credit limit is thereby determined by the mobile node's sending rate. While this is constant for Internet telephony, the throughput of TCP-based file transfers shrinks with higher round-trip times as shown in section

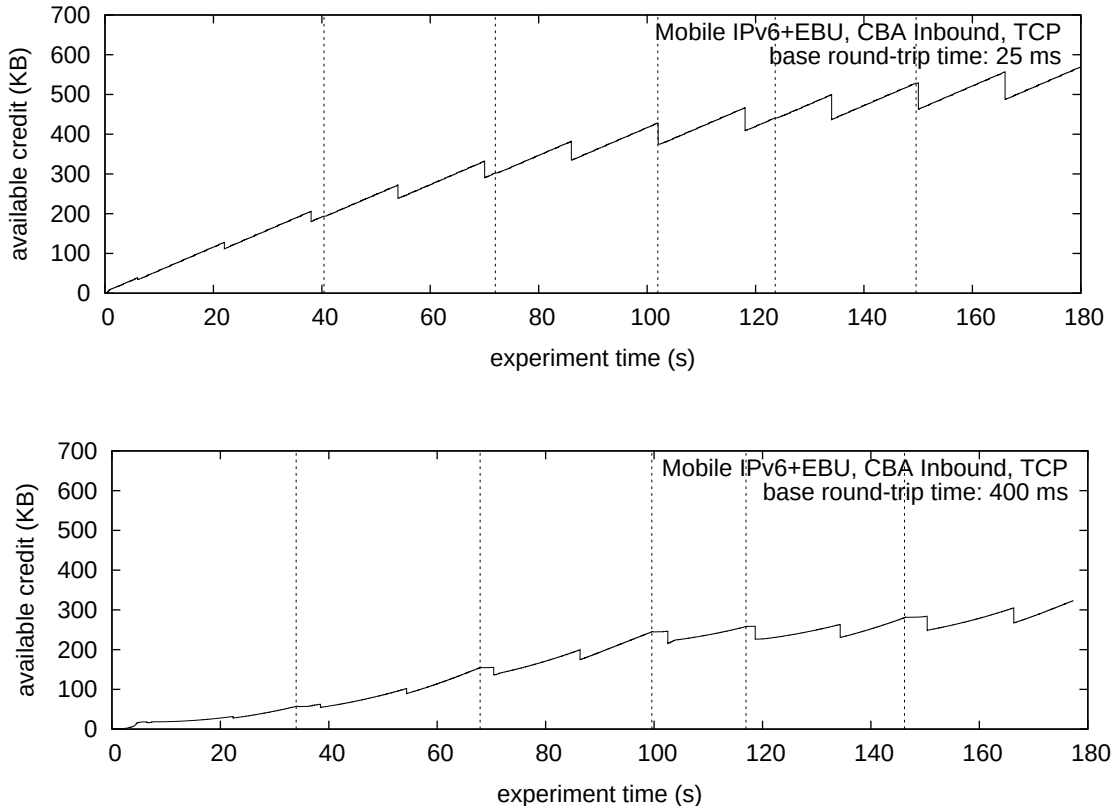



Figure 6.25: Credit availability in CBA Inbound mode

6.4.1. This leads to the less aggressive credit acquisition in the bottom chart of figure 6.25 compared to top chart. The lower amount of credit available matches a lower demand for credit in the same scenarios.

The vertical lines in the charts mark the times at which the mobile node pursues a handover. Handover-related credit consumption is in general low for TCP connections if mobility is management in reactive manner. It is typically equivalent to the single segment that the correspondent node retransmits upon the expiration of its retransmission timer, that is, 1108 B. One round-trip time passes until this segment is acknowledged and the correspondent node sends another segment. Reachability verification is then already complete and the new care-of address has been moved to Verified state. This procedure alone is independent of the round-trip time on the transmission path. On the other hand, figure 6.25 indicates that the reduction of the available credit at around the time of a handover is still more substantial in the experiment with a base round-trip time of 400 ms than it is in the experiment with a base round-trip time of only 25 ms. The reason here is that the longer the base round-trip time is, the longer it takes TCP to ramp its throughput up to the bandwidth available on the new transmission path. This holds for both Slow Start mode and Congestion Avoidance mode, since in both cases the throughput increment per round-trip time does not depend on the round-trip time itself. It hence takes much longer to fill the transmission path's available bandwidth-delay product when the round-trip time is long. Since the mobile node's acquisition of new credit is proportional to throughput at any given point in time, longer round-trip times inevitably

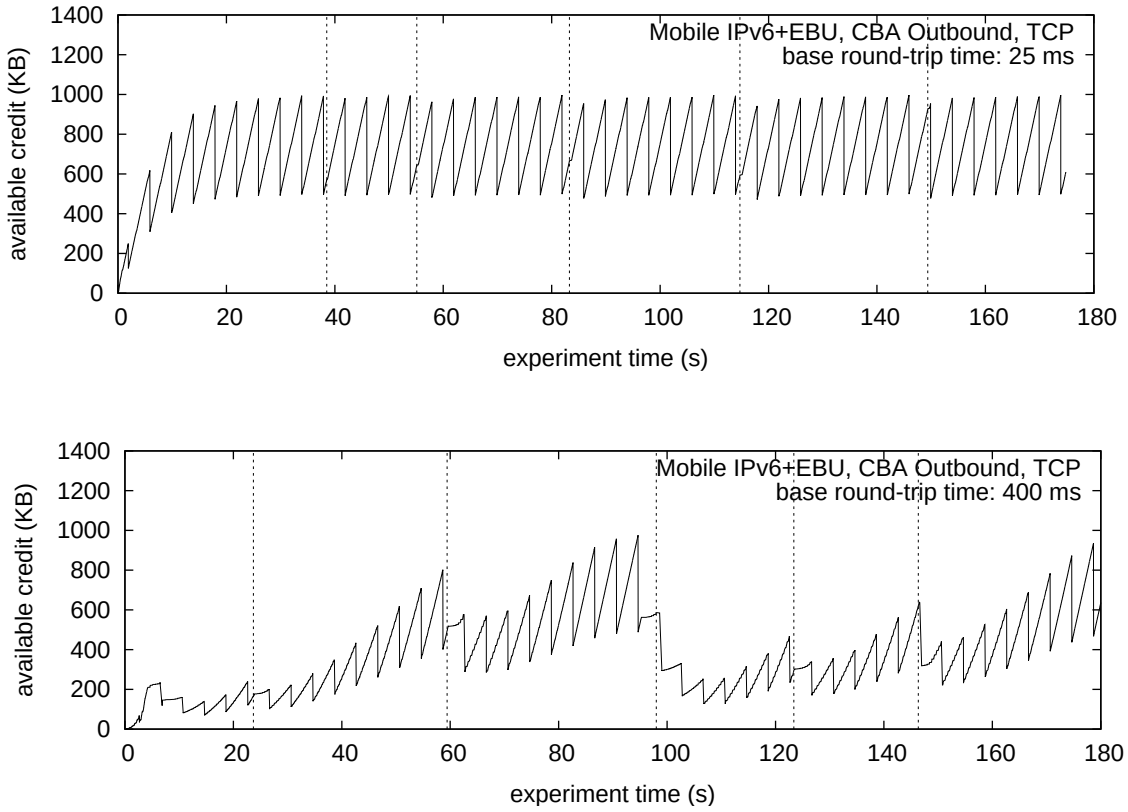



Figure 6.26: Credit availability in CBA Outbound mode

lead to lower credit acquisition after a handover. The effect of credit aging may then annihilate any newly earned credit, or it may even lead to a temporary decrease in the available credit as is visible in the bottom chart of figure 6.25.

Delayed acknowledgments may further contribute to credit consumption during a handover when the round-trip time between the mobile node and the correspondent node is 100 ms or higher. A delayed acknowledgment then reaches the correspondent node before reachability verification for the new care-of address completes—and also before the retransmission timer expires—so that the solicited new segment consumes credit. On the other hand, if the base round-trip time between the mobile node and the correspondent node is less than 100 ms, the delayed acknowledgment arrives at the correspondent node at a time reachability detection has already completed and the new care-of address has been moved to Verified state. Very occasionally, a delayed acknowledgment might cause the correspondent node to transmit two new segments. This happens when the congestion window size, which the correspondent node successively increases for each received acknowledgment, reaches the point at which the correspondent node can increase the amount of in-flight data by one segment.

The more rigid credit aging in Credit-Based Authorization Outbound mode leads to stronger fluctuations in credit availability, although the fundamental characteristics remain the same as those in Inbound mode. The two charts in figure 6.26 show the progression of the mobile node's available credit in experiments that differ from those discussed before only in the Credit-Based Authorization mode. As expected,

the credit now grows much faster since it is accumulated based on the correspondent node's transmission rate rather than on the mobile node's. The stronger credit aging compensates this effect, however, since it leads to sharper periodic reductions. Since the aging functions of Inbound mode and Outbound mode are optimized for low processing overhead rather than for enforcing comparable credit availability in both modes, the amount of credit that actually ends up being available on a durable basis is not exactly the same in Outbound mode as it is in Inbound mode.

The bottom chart in figure 6.26 also illustrates the impact that handover-unrelated retransmission timeouts may have on the available credit. A retransmission timeout occasionally occurs even in the absence of a handover on the mobile-node side when packets are lost on a path with a bandwidth-delay product that is close to the mobile node's receive buffer capacity. The mobile node may in such a situation be forced to drop data that was previously received out of order and selectively acknowledged. A bug in the TCP implementation of FreeBSD in this case causes the mobile node to continue selectively acknowledging the dropped data. The correspondent node hence does not retransmit the dropped data until after a retransmission timeout.

As in the context of Internet telephony, the use of spot checks in conjunction with the Outbound mode of Credit-Based Authorization does not lead to a shortage of credit during handover. Again, the only major difference which spot checks make pertains to the loss of one round-trip time worth of segments that the correspondent node sends to the mobile node's old care-of address at a time the mobile node has already left the old point of attachment. This loss can be detected by the correspondent node only through the use of spot checks. For a TCP connection, an upper bound for this amount of lost data is the 64 KB of send buffer space on the correspondent node side because the correspondent node can never send more than this without getting an acknowledgment. The foregoing discussion has shown that the amount of credit available during a handover is high enough to spare another 64 KB, so the use of spot checks does not become a performance-limiting factor. A separate discussion of Outbound mode with spot checks is hence omitted at this place.

## 6.5 TCP File Transfers with Proactive Mobility Management

The classic approach of handling end-to-end mobility in reactive manner has been found to force TCP into at least one retransmission timeout due to the loss of an entire window of data. Mobility optimizations can reduce the number of consecutive retransmission timeouts from two to one, but a single timeout always remains. On the other hand, proactive mobility management facilitates post-handover TCP recovery in Fast Recovery mode, provided that the network topology is adequate and handover-related latencies except those of Mobile IPv6 itself are sufficiently small. The long delays that come along with one, possibly two consecutive retransmission timeout can then be avoided. The following analysis shows in which scenarios retransmission timeouts can be avoided, and how this impacts TCP's responsiveness to packet loss and loss recovery performance .

### 6.5.1 Packet Loss

The theoretic elaboration in section 5.3 shows that proactive mobility management based on Mobile IPv6 with Early Binding Updates can eliminate packet loss when the

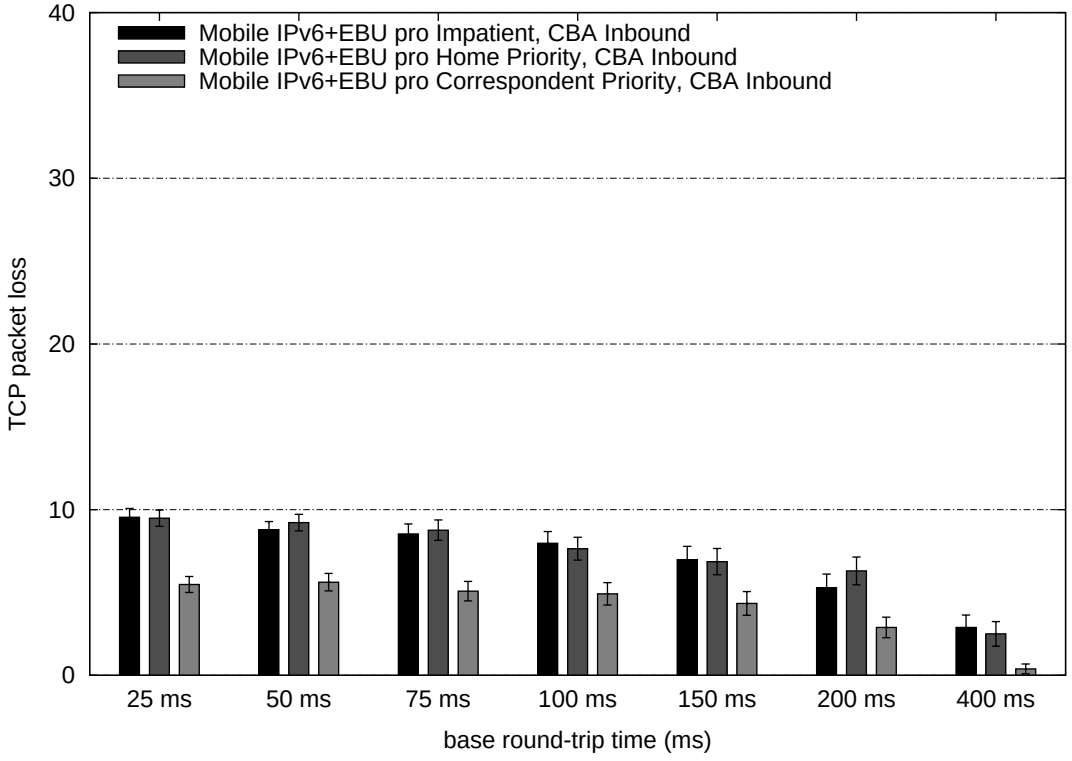



Figure 6.27: Mean packet loss in TCP SACK

actual propagation latency on the path from the correspondent node to the mobile node's old care-of address equals the actual propagation latency on the path from the correspondent node to the mobile node's new care-of address. (The assumption here is that additional handover delays at the link layer and elsewhere at the IP layer are negligible.) In reality however, propagation latencies on two paths seldom match exactly. Propagation latencies are variable even on a specific path since they depend to a significant degree on forward buffering delays in routers.

The first set of experiments demonstrate how forward buffering delays in routers determine the amount of handover-related packet loss for proactive mobility management with Impatient mode, Home Priority mode, and Correspondent Priority mode. The network topology examined are symmetric, so differences in the actual round-trip times of any two paths are solely due to higher forward buffering delays in congested routers. The link layer handover delays are zero, and the use of the DNA protocol and Optimistic Duplicate Address Detection eliminates the delays of router discovery, movement detection, and IP address auto-configuration. Figure 6.27 displays the mean number of packet losses measured in these experiments. The results are fundamentally different from those gained for reactive mobility management (see section 6.4.5): While packet losses increase with the round-trip time in reactive mobility management, losses actually shrink as the round-trip time grows when proactive mobility management is used in the observed topology. This circumstance is more closely illuminated in the following.

TCP's continuous probing for additional available bandwidth fills the forward buffer in the bottleneck router on the transmission path. In the testbed, this bottleneck router is the mobile node's serving access router. Figure 6.28 illustrates the origin

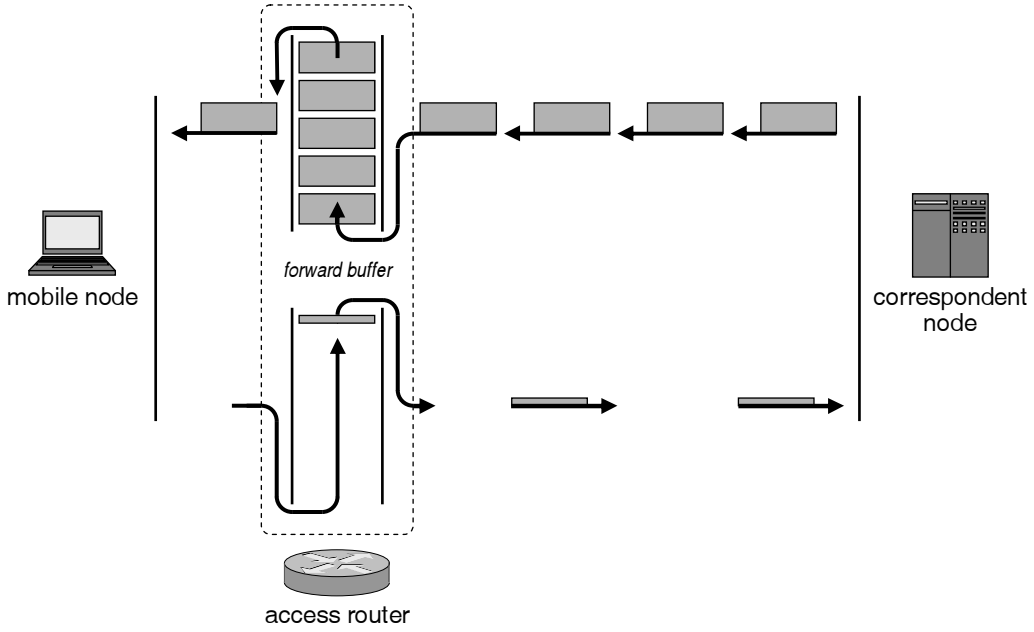



Figure 6.28: Forward buffering delays in opposite traffic directions

and impact of forward buffering. The dark-shaded rectangles symbolize the full-sized TCP segments that the correspondent node sends. These fill the forward buffer in the access router, increasing the actual propagation latency on the path from the correspondent node to the mobile node to beyond the base propagation latency. In contrast, the mobile node sends only one small acknowledgment per each two received segments, denoted in the figure by the thinner rectangles. The access router can easily forward these packets as they arrive, so forward buffering delays on the reverse path are negligible and the actual propagation latency on this path more or less equals the base propagation latency.

The low forward buffering delays on the path from the mobile node to the correspondent node are similar to those on the path from the mobile node to the home agent because the workload on both paths is essentially equally low. This affords simultaneous delivery of the mobile node's Binding Update messages to the home agent and to the correspondent node. Yet, forward buffering on the path from the correspondent node to the mobile node delays the correspondent node's Binding Acknowledgment message vis-à-vis the home agent's Binding Acknowledgment message. Figure 6.29 illustrates the situation in the testbed at the time the Binding Acknowledgment messages from the home agent and the correspondent node—symbolized by the thin, light-shaded rectangles—arrive at the mobile node's serving access router. The two Binding Acknowledgment messages are enqueued in separate forward buffers as they come via different paths. The buffer for the path from the home agent is empty because no packets were received via this path recently. But the Binding Acknowledgment message from the correspondent node follows half a window worth of TCP segments, so it gets delayed by forward buffering until these segments have been forwarded. The time lag between the arrival of the home agent's and the correspondent node's Binding Acknowledgment messages discriminates the modes of proactive mobility management. In Home Priority mode, the mobile node initiates the link layer handover already when it receives the Binding Acknowledg-

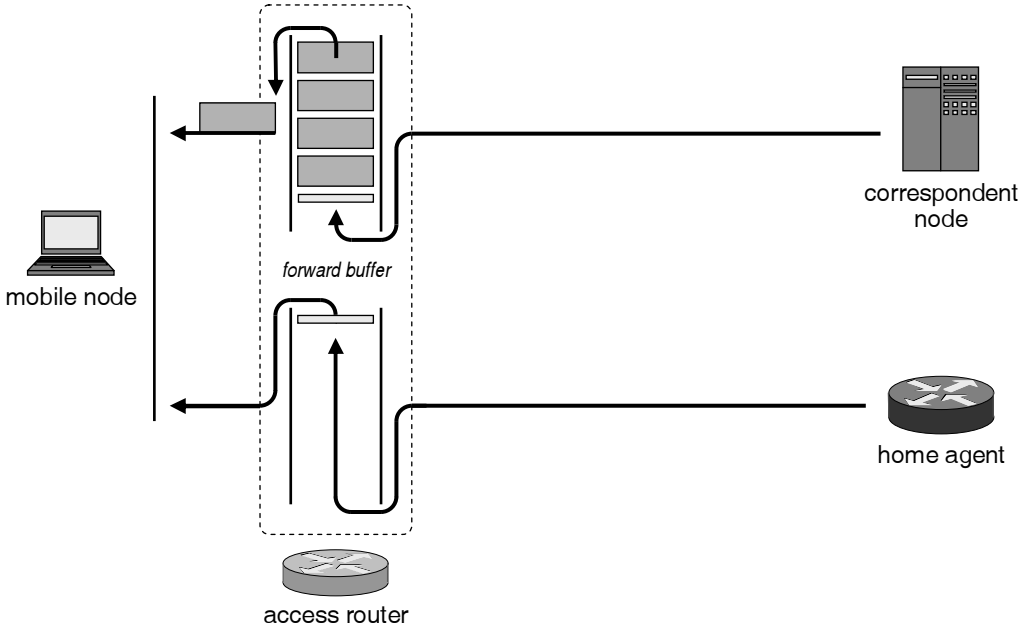



Figure 6.29: Forward buffering delays on different signaling paths

ment message from the home agent, so it never receives the correspondent node's Binding Acknowledgment message at the old point of attachment. The same happens with Impatient mode, where the mobile node initiates the link layer handover on the first received Binding Acknowledgment message, independent of whether the acknowledgment pertains to the home registration or to the correspondent registration. The mobile node waits for the correspondent node's Binding Acknowledgment message only if it runs in Correspondent Priority mode.

Forward buffering is also responsible for an overlap in the delivery of TCP segments at the mobile node's old and new point of attachment. Proactive mobility management enables the correspondent node to send without interruption while redirecting the mobile node's packets from the old to the new care-of address. The new transmission path is much less loaded than the old one, so the first packets sent to the new care-of address do not experience the forward buffering delays that packets sent to the old care-of address may be subject to.[6] Packets buffered on the old transmission path at the time of the handover therefore arrive at the mobile node's old access router while packets already appear at the new access router. This inevitably leads to packet loss. In fact, as the base round-trip times on the old and the new transmission path are identical in the observed experiments, any packet losses are solely due to forward buffering in an access router.

It depends on the mode of proactive mobility management whether the mobile node moves to the target access point early and consequently misses part of the segments in flight on the old transmission path, or if it collects these segments at the cost of

[6]It should be acknowledged that the properties of real Internet paths are in most cases less predictable than the properties of paths in the testbed due to the coexistence of other packet flows. However, the testbed conditions clearly demonstrate how forward buffering delays can increase the actual propagation latency for packets on congested paths and how this interferes with proactive mobility management. The findings are thus important in understanding the behavior and performance that proactive mobility management would exhibit in the real Internet.

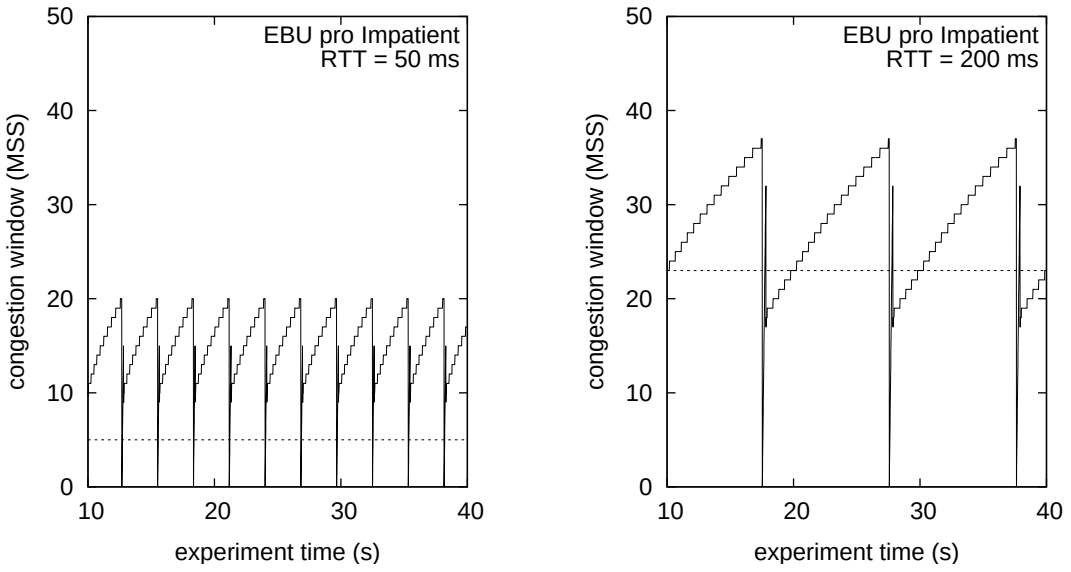



Figure 6.30: Evolution of TCP's congestion window with proactive mobility management in Impatient mode and round-trip times of 50 ms as well as 200 ms

losing packets that meanwhile arrive at the target access router. In the experiments with Home Priority mode or Impatient mode, the mobile node misses any packets that routers on the old transmission path may have queued at the time the home agent's Binding Acknowledgment message is delivered. Segments in flight on the old transmission path are collected by the mobile node only in Correspondent Priority mode. On the other hand, a mobile node operating in Home Priority or Impatient mode typically arrives in time at the target access point to receive all segments transmitted to the new care-of address, whereas the mobile node typically misses segments at the new transmission path when it uses Correspondent Priority mode.

The number of segments buffered and therefore lost during a handover depends on the size of the correspondent node's TCP congestion window, at the time the correspondent node receives the Binding Update message, relative to the transmission path's bandwidth-delay product: An access router's forward buffer is empty until the congestion window reaches the bandwidth-delay product of the transmission path. So in a symmetric network topology, packet loss is usually zero if the handover occurs before this point. However, if the handover happens at a time the congestion window exceeds the transmission path's bandwidth-delay product, forward buffer space will be occupied, causing simultaneous packet delivery and packet loss as explained before. The load of the forward buffer is then given by the congestion window size minus the bandwidth-delay product. Forward buffers in testbed routers can hold a maximum of 14 packets, and packet loss cannot be more than this when mobility is handled in proactive manner in a symmetric network topology.

What causes the packet loss rates shown in figure 6.27 to shrink with increasing base round-trip times is that the probability for the congestion window to be large enough to load a forward buffer at the time of the handover decreases as the round-trip time grows. The number of packets that fit into the bandwidth-delay product of the transmission path is proportional to the path's base round-trip time. It is

hence relatively higher compared to the fixed capacity of the forward buffer when the base round-trip time is large. Therefore, the longer the base round-trip time, the higher the number of round-trips it takes for the congestion window to saturate the bandwidth-delay product, whereas the number of additional round-trips required to fill a forward buffer is constant. A lower average forward buffer load thus leads to fewer handover-related packet losses on average. The left and right charts of figure 6.30 show the evolution of the congestion window in terms of the number of maximum TCP segment sizes for transmission paths with 50 ms and 200 ms, respectively. The horizontal lines in the traces demarcate the number of packets that fit into the bandwidth-delay product and hence can be sent without forward buffering. The forward buffer becomes loaded as the congestion window size passes this threshold. Clearly, the integral of the part of the congestion window curve above the horizontal line is smaller for the long transmission path than it is for the short transmission path.

### 6.5.2 Packet Loss in an Asymmetric Topology

While the experiments discussed in the previous section were obtained in scenarios where the symmetric network topology accommodates an efficient proactive mobility management, the experiments observed in the following illuminate the performance of proactive mobility management when the network topology is asymmetric. The experiments hence reproduce situations in which the mobile node's link layer handover cannot be optimally synchronized with the redirection of segments between two care-of addresses. Experiments were conducted for scenarios where the base round-trip time between the mobile node and the home agent is fixed at either 25 ms or 400 ms, while the round-trip time between the mobile node and the correspondent node varies from 25 ms to 400 ms 6.5.1. All experiment parameters except the base round-trip time are the same as in the experiments from section 6.5.1.

Figure 6.31 shows the average packet losses obtained from the experiments with a fixed base round-trip time of 25 ms on all paths incident to the home agent. The x-axis in the figure is labeled by the common base round-trip time between the mobile node and the correspondent node. As one would expect, packet losses grow with the difference between the two base round-trip times if the mobile node operates in Home Priority mode because the initiation time for the link layer handover is then increasingly premature relative to the redirection of TCP segments from the old to the new care-of address. The early link layer handover leads to loss of segments in flight on the old transmission path, including those kept in the old access router's forward buffer. Where the round-trip time between the mobile node and the correspondent node is more than 50 ms, the correspondent node does not receive the mobile node's Binding Update message until the home agent's Binding Acknowledgment message has already been delivered to the mobile node. The correspondent node consequently continues sending segments to the old care-of address even after the mobile node has left the old point of attachment, increasing packet loss further. Barring some experimentation variations, Impatient mode performs identical to Home Priority mode in these experiments because the home registration latency is always less than or equal to the correspondent registration latency.

The asymmetric network topology is of no impact if the mobile node operates in Correspondent Priority mode because the initiation time of the link layer handover

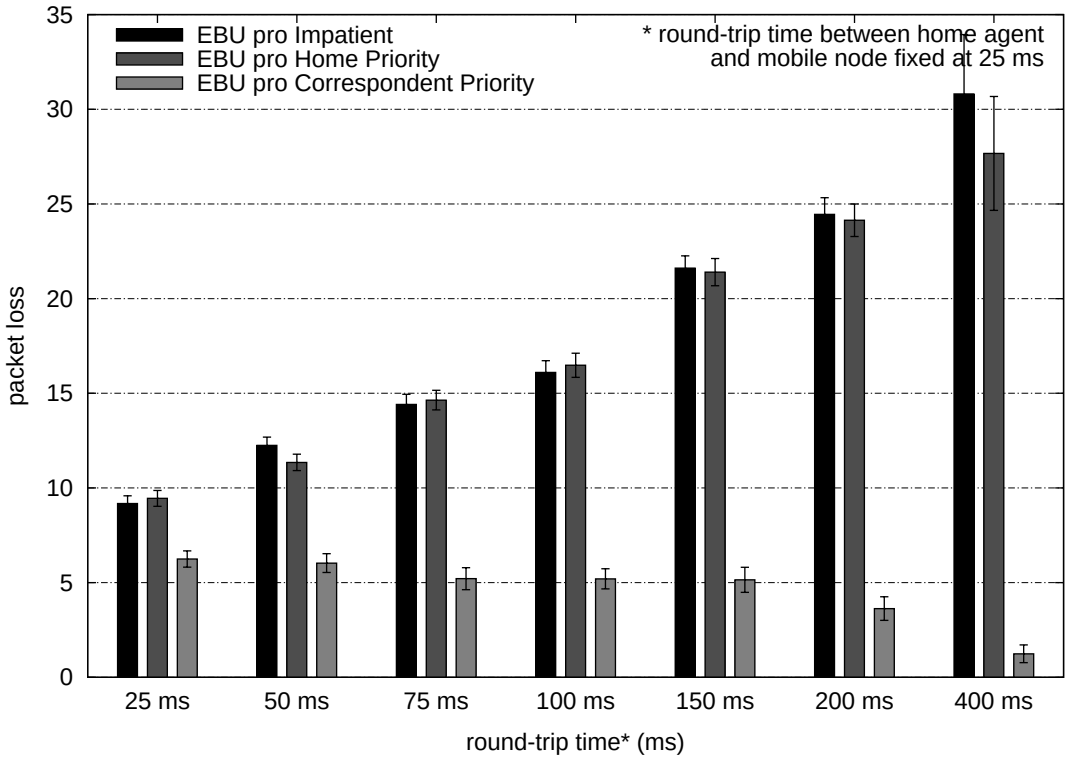



Figure 6.31: Mean packet loss in TCP SACK

is then optimized for the correspondent registration. The packet loss measurements for Correspondent Priority mode in these experiments are therefore comparable to those obtained in section 6.5.1.

Figure 6.32 shows the packet loss measurements for experiments where the base round-trip time on all paths incident to the home agent is fixed at 400 ms. Again, Correspondent Priority mode performs similar in these experiments as it does in the experiments from section 6.5.1 because the longer home registration latency then does not influence the initiation time for the link layer handover. The behavior of Correspondent Priority mode and Impatient mode is now akin because a correspondent registration never completes later than the home registration. The following discussion therefore focuses on Home Priority mode. With this mode, for a base round-trip time of up to 150 ms, packet loss increases with the base round-trip time between the mobile node and the correspondent node as it does when the base round-trip time between the home agent and any other node in the testbed is limited to 25 ms. This is because the difference between the round-trip times on the paths between the mobile node on one side and the home agent or correspondent node on the other side is large enough to keep the mobile node at the old point of attachment until a full window of data has been delivered to the new point of attachment. These segments are generally lost except the last three, which the target access router keeps in its IP address resolution buffer until the mobile node arrives at the new attachment point and fetches them. The correspondent node's available window is statistically larger with a higher bandwidth-delay product on the transmission path, which explains why the number of packet losses grows with the base round-trip time.

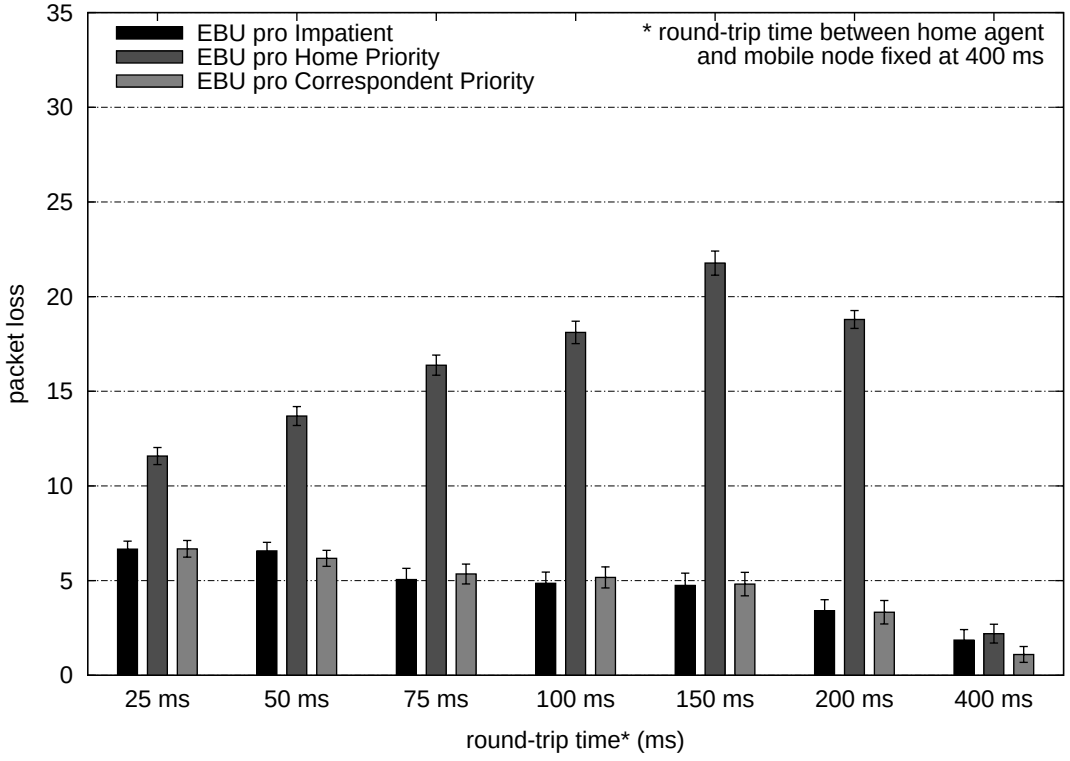



Figure 6.32: Mean packet loss in TCP SACK

Packet loss for Home Priority mode is lower when the base round-trip time between the mobile node and the correspondent node reaches 200 ms because some of the packets that the correspondent node sends to the new care-of address prior to loss detection are then received by the mobile node without the help of the target access router's IP address resolution buffer, increasing the number of received packets to more than three. The improved packet delivery can be derived mathematically as follows: Let $x$ be the base round-trip time between the mobile node and the correspondent node. In the absence of forward buffering delays on the path from the mobile node to the correspondent node, the correspondent node then receives the mobile node's Binding Update message $x/2$ time after the message was sent. TCP segments that the correspondent node sends before this time are destined to the mobile node's old care-of address, whereas segments sent subsequently are directed to the new care-of address. The stream of segments to the new care-of address is initially driven by acknowledgments that the correspondent node still receives from the old care-of address. The correspondent node dispatches new segments whenever it receives a non-duplicate acknowledgment from the mobile node, and the mobile node continues sending such acknowledgments from the old point of attachment as long as segments are delivered to the old care-of address. The stream of segments to the old care-of address typically terminates with the correspondent node's Binding Acknowledgment message, which the mobile node receives $x + n\,I_{TCP}$ time after it has sent the Binding Update message, assuming that $n$ out of the 14 slots in the old access s router's forward buffer hold full-sized TCP segments at the time the Binding Acknowledgment message is enqueued. (Recall from section 6.4.4 that $I_{TCP}$is the inter-arrival time in a continuous stream of full-sized TCP segments.) The last acknowledgment that the mobile node sends from the old point of attachment

hence arrives at the correspondent node $1.5\,x + n\,I_{TCP}$ time after the Binding Update message was sent. This triggers the last segment that the correspondent node sends to the new care-of address prior to loss detection, which, in turn, arrives at the mobile node's new point of attachment a maximum of $2\,x + n\,I_{TCP}$ time after the mobile node has sent the Binding Update message.[7] Up to three of the segments that arrive at the mobile node's new point of attachment before the mobile node connects can be stored in the target access router's IP address resolution buffer until the mobile node fetches them. These are always delivered to the mobile node eventually. Additional segments that arrive in the absence of the mobile node are lost. Packet delivery can hence be increased only if the mobile node arrives at the new point of attachment early enough to catch some of the initial segments which the correspondent node sends to the new care-of address prior to loss detection, beyond the three segments in the access router's IP address resolution buffer. Specifically, the mobile node receives these segments for a period of

$$\max\left(0,\, 2\,x + n\,I_{TCP} - 400\,\mathrm{ms}\right)$$

because it initiates the link layer handover only upon the receipt of the home agent's Binding Update message when operating in Home Priority mode, that is, 400 ms after it has sent the Binding Update messages. The formula shows that, if $x$ is 150 ms or less, the mobile node does not receive any of the aforementioned segments except the ones that the target access router's IP address resolution buffer would salvage anyway. On the other hand, if $x$ is 200 ms or higher, the mobile node typically receives at least some of these initial segments as they arrive. This explains why packet losses are lower with a round-trip time of 200 ms between the mobile node and the correspondent node than they are if the round-trip time is 150 ms.

One property of the discussed experiments is that the mobile node communicates with a single correspondent node only. Any conflicts in scheduling the link layer handover hence always arise because the link layer handover time that is optimal for bidirectionally tunneled communication sessions does not match the link layer handover time that is optimal for route-optimized communication sessions. A similar situation occurs when the mobile node communicates with multiple correspondent nodes, and there is a conflict between the optimal link layer handover times for communication sessions with different correspondent nodes. Synchronizing the link layer handover with the redirection of one route-optimized session then implies decreased performance of route-optimized sessions that the mobile node pursues with other correspondent nodes. All of these situations have in common that the time of the link layer handover is sub-optimal for a particular communication session. The experiments discussed in this section reflect this property, even though they only incorporate a single correspondent node, so the obtained measurements can be transferred also to situations where the mobile node communicates with more than one correspondent node.

[7] This formula is based on the observation that forward buffering delays on the path from the correspondent node to the mobile node's new point of attachment are still negligible when the last segment sent prior to loss detection meets the target access router.

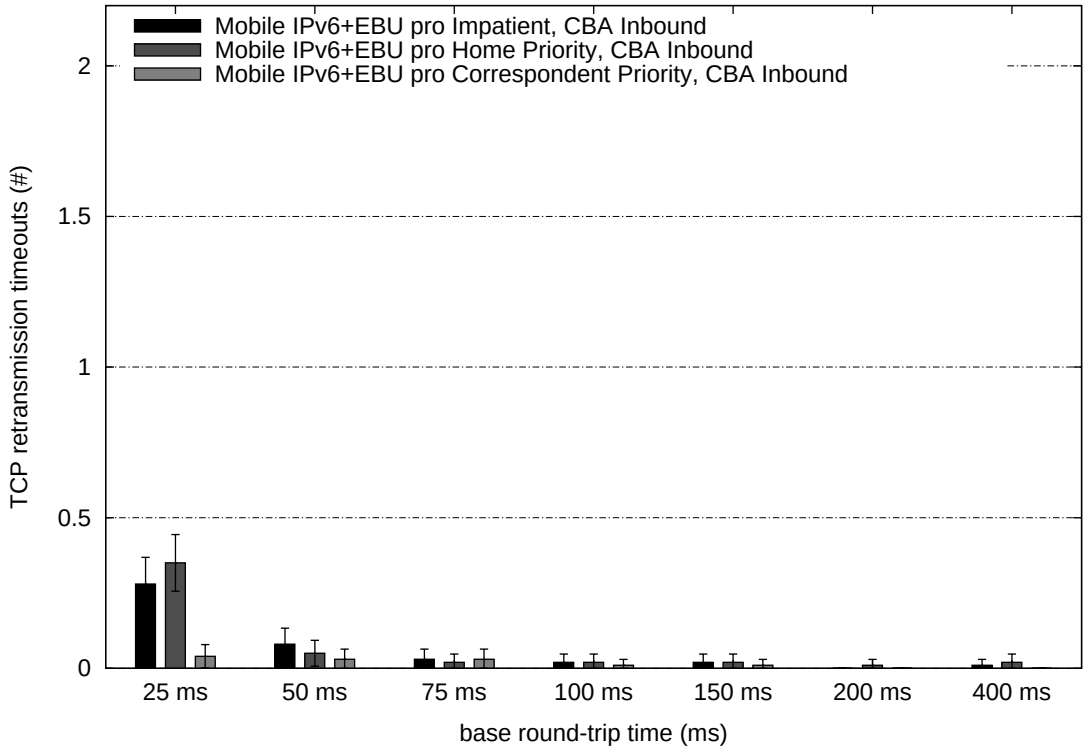



Figure 6.33: Mean retransmission timeout count in TCP SACK

### 6.5.3 Retransmission Timeouts

The benefits of proactive mobility management in the absence of link layer handover delays and other handover-related delays outside Mobile IPv6 are strongest in a symmetric network topology. Figure 6.33 shows how this influences the average number of retransmission timeouts that the correspondent node goes through during a handover on the mobile-node side. It juxtaposes averages and 95% confidence intervals for Home Priority mode, Correspondent Priority mode, and Impatient mode measured in the same experiments that are used in section 6.5.1. The measurements indicate that loss recovery usually proceeds without a retransmission timeout unless the base round-trip time is only 25 ms and the mobile node runs in either Home Priority mode or Impatient mode. The low probability for a retransmission timeout is because the number of out-of-order segments, which the mobile node typically receives at the new point of attachment after a handover involving packet loss, is sufficient to prompt three or more duplicate acknowledgments. The duplicate acknowledgments, in turn, trigger a fast retransmit at the correspondent node prior to the expiration of the retransmission timer, and the included Selective Acknowledgment options permit the correspondent node to quickly retransmit the lost segments. The delivery of at least three of the early segments that the correspondent node sends to the new care-of address is thereby aided by the target access router's IP address resolution buffer. This temporarily holds the last three segments that the access router has received before the mobile node arrives at the new point of attachment.

Loss recovery through a retransmission timeout is in most cases due to the mobile node receiving less than three out-of-order segments after arriving at the new point of attachment. Since the correspondent node does not support the Limited Transmit

algorithm in the observed experiments, the mobile node is then unable to generate the threshold of three duplicate acknowledgments required to trigger a fast retransmit on the correspondent node side. This primarily happens in the experiments with Home Priority mode or Impatient mode when the base round-trip time is only 25 ms, as explained next.

Segments that the correspondent node sends to a new care-of address are clocked by the reception of non-duplicate acknowledgments after the mobile node's Binding Update message has been processed. How many of these segments the correspondent node sends before it detects handover-related packet loss thus depends on the amount of data that is covered by the acknowledgments it receives up to that point. These acknowledgments comprise the ones in flight at the time the Binding Update message arrives at the correspondent node as well as the additional acknowledgments that the mobile node generates before it initiates the link layer handover to the new attachment point. Consequently, if the mobile node operates in Home Priority mode or—equivalent in these experiments—in Impatient mode, the relevant non-duplicate acknowledgments are those which the mobile node generates between the transmission of the Binding Update messages and the reception of the home agent's Binding Acknowledgment message. This period is close to the base round-trip time between the mobile node and the home agent due to the lack of any substantial forward buffering delays. it is slightly longer given some processing delays in both end nodes. So with a base round-trip time of only 25 ms and an inter-arrival time for full-sized segments of $I_{TCP} = 8.7\,\text{ms}$, the number of segments received between the transmission of the Binding Update messages and the reception of the home agent's Binding Acknowledgment message is usually three or four. It may be less if, for instance, recent packet loss has led to a reduction in the correspondent node's congestion window, causing less segments to be in flight simultaneously.

The segments call for the transmission of up to three cumulative acknowledgments. Each of these acknowledgments except the last one covers exactly two segments, which in the case of the first acknowledgment may include a segment received before the Binding Update messages were sent. The last acknowledgment may cover either one or two segments, and it is delayed if it covers one. Together, the acknowledgments hence confirm the receipt of up to five segments. A retransmission timeout occurs if the acknowledgments cover only three segments and the last acknowledgment is delayed. The last acknowledgment is in this case held until the mobile node receives the first out-of-order segment on the new link, at which point it is sent in replacement for the first duplicate acknowledgment. The number of duplicate acknowledgments that the correspondent node receives is then less than three as well, foiling a fast retransmit. The measurements for a base round-trip time of 25 ms in figure 6.33 indicate that chances are slightly lower for a retransmission timeout to occur than they are for a fast retransmit if the mobile node operates in Home Priority mode or Impatient mode.

A handover-related retransmission timeout may occasionally also occur—independently of any round-trip time or the mode of proactive mobility management—when the correspondent node's congestion window is small at the time of the handover. This leads to a reduced amount of in-flight data and causes the mobile node to receive less than three segments between the transmission of the Binding Update messages and the initiation of the link layer handover. The

amount of data for which the correspondent node receives an acknowledgment after processing the mobile node's Binding Update message may then be less than three segments, meaning that the number of segments the correspondent node sends to the new care-of address prior to loss detection is less than three, too. The mobile node thus cannot generate the three duplicate acknowledgments necessary to trigger a fast retransmit and to preempt the expiration of the correspondent node's retransmission timer.

Use of the Limited Transmit algorithm on the correspondent node side can facilitate loss recovery in Fast Recovery mode also in cases where lack of three duplicate acknowledgments would otherwise cause the expiration of the correspondent node's retransmission timer. The new segments which the correspondent node then sends upon the receipt of the first and second duplicate acknowledgment eventually permit the mobile node to generate a third duplicate acknowledgment. This triggers a fast retransmit on the correspondent node side, affording loss recovery without a retransmission timeout.

The probability for a retransmission timeout to occur is low despite a base round-trip time of only 25 ms and irrespective of lack of support for the Limited Transmit algorithm on the correspondent node side if the mobile node operates in Correspondent Priority mode. This mode requires the mobile node to rest at the old attachment point until it receives the correspondent node's Binding Acknowledgment message, so any segments delayed in the old access router's forward buffer are successfully delivered to the mobile node during this time. The additional non-duplicate acknowledgments that the mobile node thereby generates cause the correspondent node to send a higher number of segments to the new care-of address. These segments typically fill the three slots that the IP address resolution buffer in the target access router provides. The buffered segments are eventually delivered to the mobile node when the mobile node arrives on the new link, facilitating the transmission of at least three duplicate acknowledgments and, consequently, the circumvention of a retransmission timeout.

Neither the Limited Transmit algorithm on the correspondent node side nor the use of Correspondent Priority mode on the mobile-node side helps to avoid a retransmission timeout if the correspondent node performs a fast retransmit shortly before the mobile node pursues a handover and the retransmitted segment gets lost during the handover. Most TCP SACK implementations are unable to detect the loss of retransmitted segments, including the implementation that was used in the testbed. This also hold also true for the TCP SACK algorithm proposed in [20].

### 6.5.4 Retransmission Timeouts in an Asymmetric Topology

The number of handover-related retransmission timeouts that a correspondent node goes through in an asymmetric network topology was studied based on the same experiments that were used in section 6.5.2. Figure 6.34 shows the averages and 95% confidence intervals measured in a network topology where the base round-trip time on any path incident to the home agent is fixed at 25 ms, and the base round-trip time between the mobile node and the correspondent node varies between 25 ms and 400 ms. In line with the packet loss measurements for the same experiments, the performance of Correspondent Priority mode is again not affected by the relatively short latency of the home registration because the mobile node here optimizes the

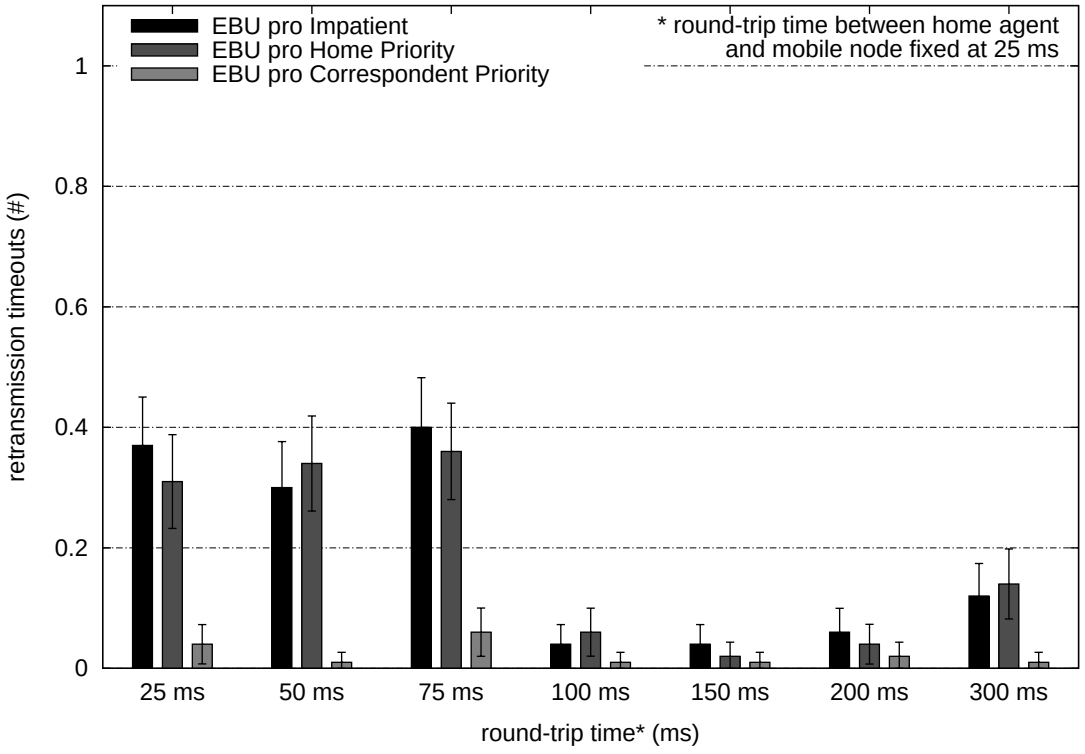



Figure 6.34: Mean retransmission timeout count in TCP SACK

link layer handover initiation time for the communication session with the correspondent node. The situation is different if the mobile node runs in Home Priority mode and hence already initiates the link layer handover when it receives the home agent's Binding Acknowledgment message after a round-trip time of 25 ms. This may lead to a retransmission timeout in the scenarios where the base round-trip time between the mobile node and the correspondent node is 75 ms or less, or where it is 400 ms, as explained below. Impatient mode is not considered separately here given that it is equivalent to Home Priority mode in these experiments due the home agent's Binding Acknowledgment message always arriving before the correspondent node's.

The reason for the increased number of handover-related retransmission timeouts with Home Priority mode or Impatient mode, when the base round-trip time between the mobile node and the correspondent node is 75 ms or less, turns out to be the same as the one observed in section 6.5.3 for the symmetric network topology in which all paths have a base round-trip time of 25 ms. In both cases, the early link layer handover effected by the small base round-trip time between the mobile node and the home agent limits to usually three or four the number of TCP segments that the mobile node receives at the old point of attachment after it has sent the Binding Update messages. When there are only three segments and the last acknowledgment generated for these segments is delayed to a time after the mobile node receives the first out-of-order segment on the new link, the delayed acknowledgment replaces the first duplicate acknowledgment. The number of duplicate acknowledgments that the correspondent node then receives is less than three, foiling a fast retransmit and hence leading to a retransmission timeout. This situation may arise only if the base

round-trip time between the mobile node and the correspondent node is below the maximum hold time of 100 ms for delayed acknowledgments, yielding the increased retransmission timeout probabilities for base round-trip times between the mobile node and the correspondent node of 75 ms or less.

Handover-related retransmission timeouts are an exception when the base round-trip time between the mobile node and the correspondent node takes a value of between 100 ms and 200 ms. In these cases, out-of-order segments do not arrive at the new link until after the mobile node has dispatched any delayed acknowledgment, so a delayed acknowledgment is never merged with a subsequent duplicate acknowledgment. This yields an additional acknowledgment, which in turn helps to reach the threshold of three duplicate acknowledgments and, thus, to avert the expiration of the correspondent node's retransmission timer.

The minimum base round-trip time between the mobile node and the correspondent node as of which the described effect of delayed acknowledgments becomes negligible can be derived more specifically as follows: The correspondent node sends the first segment to the mobile node's new care-of address when it receives the first acknowledgment following the Binding Update message. The latency between the transmission of this acknowledgment and the delivery of the first segment to the new care-of address equals the sum of the actual one-way time from the mobile node's old care-of address to the correspondent node and the actual one-way time from the correspondent node to the mobile node's new care-of address. Since this does not include any forward buffering delays except for the segment itself, the latency amounts to $x + I_{TCP} = x + 8.7\,\text{ms}$, where $x$ is the base round-trip time between the mobile node and the correspondent node. On the other hand, the time gap between the transmissions of the second-to-last and the last acknowledgments following the Binding Update message from the old link is 108.7 ms, which in this case comprises the inter-arrival time for full-sized TCP segments, $I_{TCP}$, plus the 100 ms of hold time for delayed acknowledgments. If $x$ takes a value of 100 ms and more, TCP has already generated the delayed acknowledgment when the first out-of-order segment arrives at the mobile node.

The probability for a handover-related retransmission timeout is again higher with a 400-ms base round-trip time between the mobile node and the correspondent node, as shown in figure 6.34. Similar to the scenarios with short round-trip times, a timeout happens when the mobile node receives less than three out-of-order segments after a handover involving packet loss, and consequently fails to generate the three duplicate acknowledgments required to trigger a fast retransmit on the correspondent node side. However, the lack of segments arriving at the new point of attachment is in this case abetted by a lower average TCP throughput, which in turn comes as a result of the TCP send and receive buffers being relatively small compared to the bandwidth-delay product of the transmission path. Specifically, the buffers become a throughput-limiting factor when the correspondent node, after detecting packet loss which may or may not be handover-related, has one window worth of outstanding data. The small buffers then force the correspondent node to stop sending new data shortly after having retransmitted the lost data. Transmission of new data can continue only when the receipt of a non-duplicate acknowledgment, typically covering the retransmitted data plus the one window worth of new data received prior to the delivery of the retransmission, advances the left edge of the

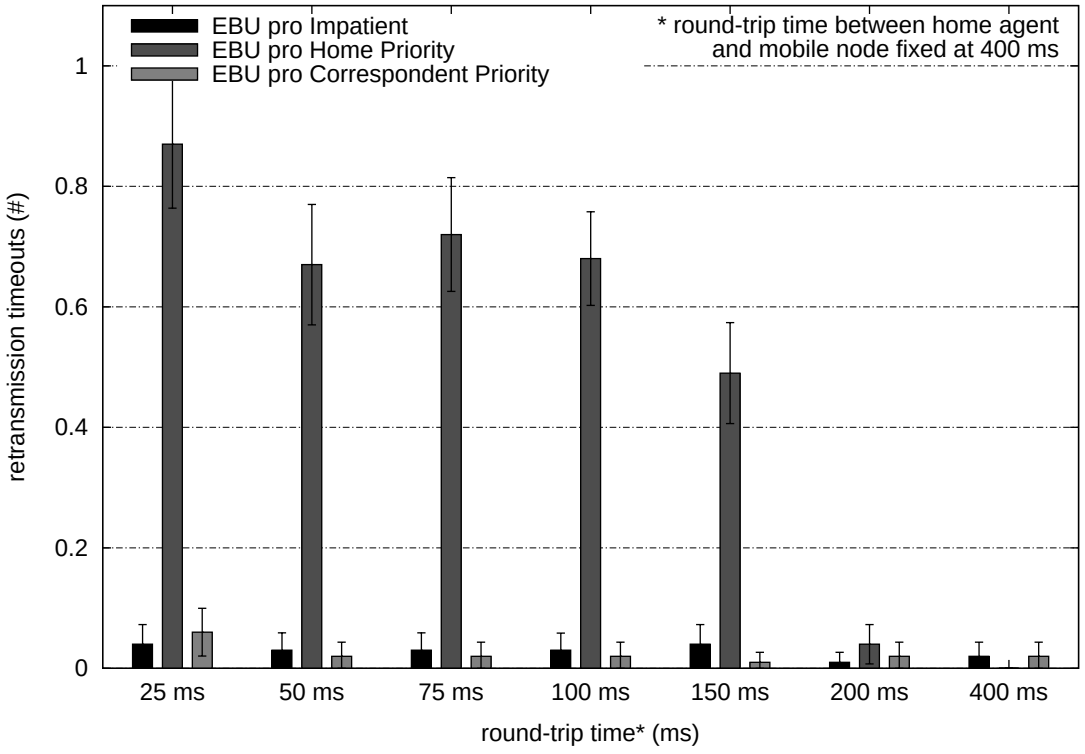



Figure 6.35: Mean retransmission timeout count in TCP SACK

send window. The result of the transmission pause, which happens whenever the correspondent node enters Fast Recovery mode after detecting packet loss, is a drain in the amount of outstanding data down to a few segments only. Compared to the normal TCP throughput behavior on paths with a lower bandwidth-delay product, where the amount of outstanding data is reduced to half in the event of packet loss, this leads to a very low utilization of the capacity available on the transmission path and hence to an increased chance that the mobile node receives less than three out-of-order segments subsequent to a handover.

Figure 6.35 shows the mean number of retransmission timeouts on the correspondent node side when the base round-trip times on the paths incident to the home agent are fixed at 400 ms. The figure demonstrates a high probability for the correspondent node's retransmission timer to expire when the mobile node operates in Home Priority mode and the base round-trip time between the mobile node and the correspondent node is 150 ms or less. This is because the time gap between the arrival of the last non-duplicate acknowledgment from the old point of attachment at the correspondent node and the arrival of the third duplicate acknowledgment from the new point of attachment is in these cases larger than the correspondent node's retransmission timeout period. The correspondent node thus falls into a retransmission timeout, even though it eventually receives three duplicate acknowledgments from the mobile node.

The probability for the three acknowledgments to reach the correspondent node prior to the expiration of the retransmission timer is slightly higher in the scenario where the base round-trip time between the mobile node and the correspondent node is 150 ms than it is for shorter round-trip times, as indicated in figure 6.35.

Whether or not a retransmission timeout happens depends on the time the mobile node generates the last non-duplicate acknowledgment on the old link relative to the transmission of the Binding Update messages, as well as on the correspondent node's estimated retransmission timeout period at the time the correspondent node receives this last acknowledgment and resets the retransmission timer for the last time prior to loss detection.

Handover-related retransmission timeouts become very seldom in Home Priority mode if the actual round-trip times between the mobile node on one side and the home agent or correspondent node on the other side differ by less than the correspondent node's estimated retransmission timeout period. This is the case when the base round-trip time between the mobile node and the correspondent node is 200 ms or more. The phase during which the mobile node does not send any acknowledgment while waiting for the home agent's Binding Acknowledgment message on the old link is then shorter than the correspondent node's retransmission timeout period, which in turn is always longer than or equal to the actual round-trip time between the mobile node and the correspondent node.

As with the experiments discussed in section 6.5.3, a different cause for the correspondent node to fall into a retransmission timeout is that the correspondent node is in Fast Recovery mode at the time the mobile node pursues the handover, and a retransmitted segment falls victim to packet loss. This may happen independently of the round-trip times involved or the mode of proactive mobility management, and it explains why a retransmission timeout may occasionally occur across all scenarios shown in figures 6.34 and 6.35.

### 6.5.5 Handover Delay

In section 6.5.1, the amount of handover-related packet loss caused by proactive mobility management in a symmetric network topology has been found to depend primarily on forward buffering delays on a transmission path, while the actual round-trip times in the topology turn out to be of minor impact only. In contrast, the correspondent node's detection of this loss is based on the delivery of three duplicate acknowledgments or the expiration of the retransmission timer, which both is a function of the actual round-trip time between the correspondent node and the mobile node. Since the end of the handover phase coincides with packet loss detection at the correspondent node, the handover delay, too, is round-trip-time-dependent.

Figure 6.36 shows the average handover delays and 95% confidence intervals for proactive mobility managmenet in a symmetric network topology. The measurements were obtained from the experiments used in sections 6.5.1 and 6.5.3. The influence that the round-trip time has on the handover delay more than compensates the trend towards lower packet losses on paths with high round-trip times, which was found in section 6.5.1. This is why handover delays increase slightly as the propagation latency gets higher.

The measurements for Impatient mode and Home Priority mode at a base round-trip time of 25 ms fall slightly out of line since they actually exceed the corresponding measurements at a base round-trip time of 50 ms. The reason is a higher probability for a handover-related retransmission timeout in the experiments with a base round-trip time of 25 ms, as explained in section 6.5.3. The retransmission timeout

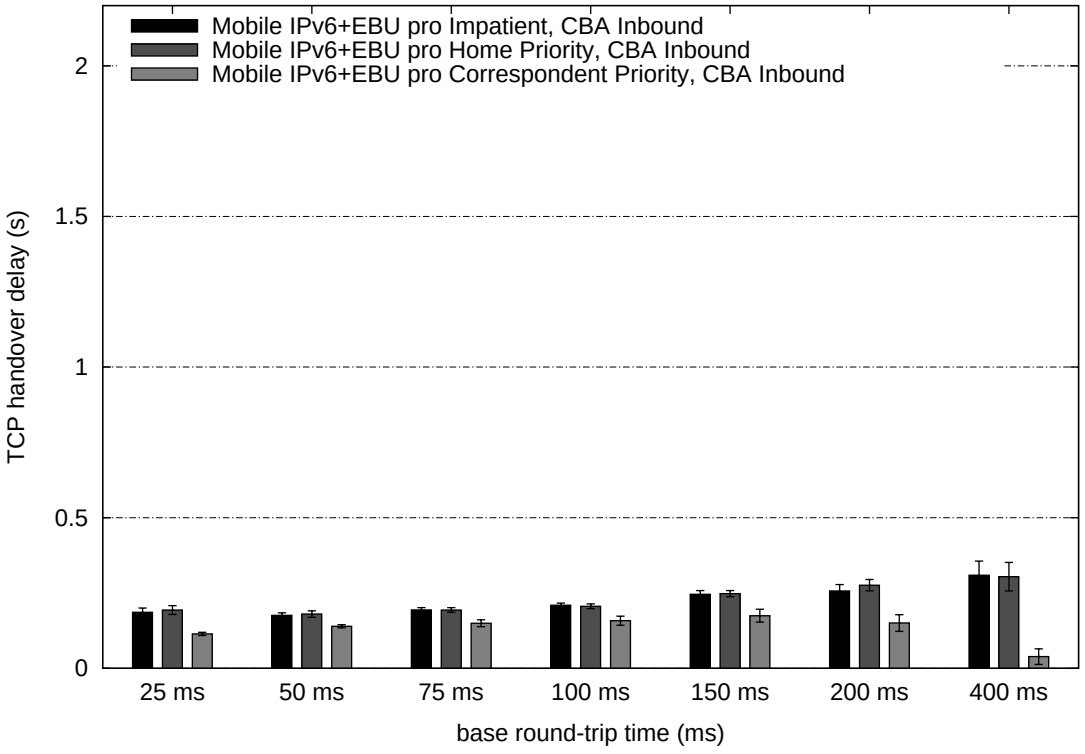
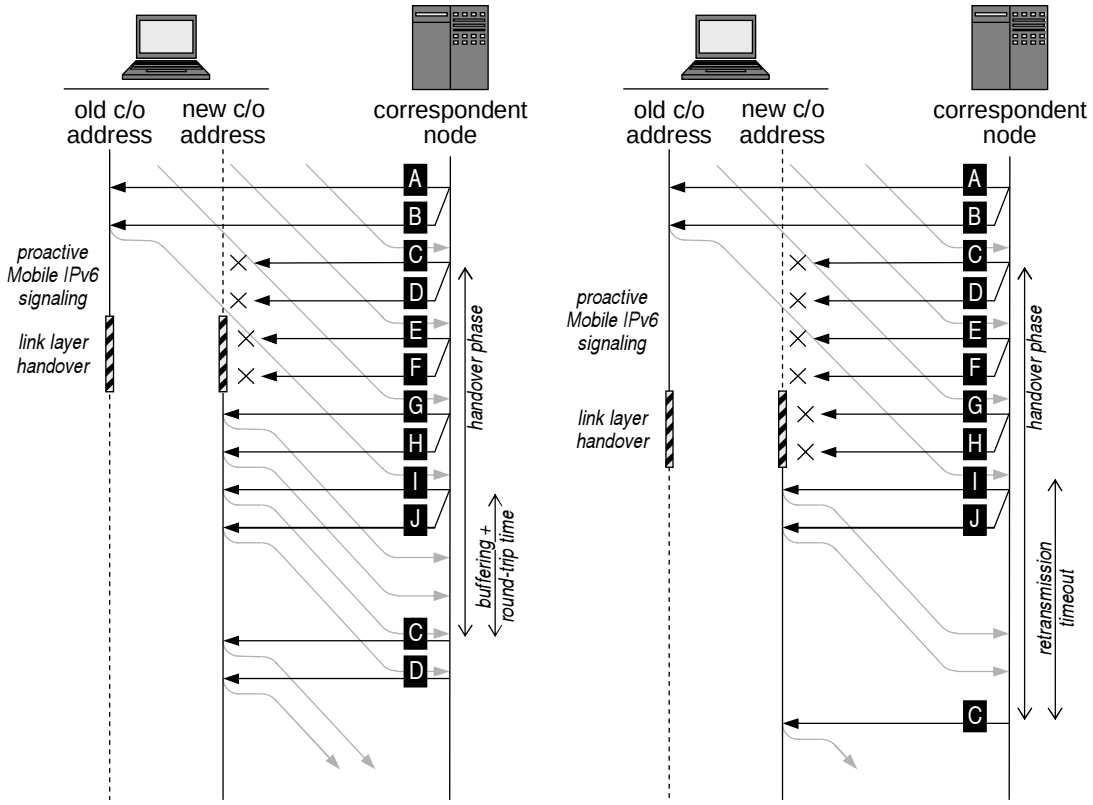



Figure 6.36: Mean handover delays in TCP SACK

period incorporates forward buffering delays on the old transmission path as well as a considerable cushion for round-trip time variations and so leads to higher expected handover delays. Handover delays tend to be lower when the base round-trip time between the mobile node and the correspondent node is 50 ms or higher because loss recovery is then typically triggered by the delivery of three duplicate acknowledgments to the correspondent node. Another point that requires explanation is that the depicted handover delay averages are smaller than the base round-trip time when the latter is 400 ms, or when it is 200 ms and the mobile node runs in Correspondent Priority mode. This is due to an increased probability of handovers without any packet losses in these scenarios, and hence with zero handover delay.

Figure 6.37(a) depicts the composition of the handover phase when the correspondent node detects packet loss through the receipt of three duplicate acknowledgments from the mobile node. The handover delay here consists of two parts: First, the time it takes the correspondent node to transmit the lost segments and three more segments that are received by the mobile node and prompt duplicate acknowledgments. These are segments $C$ through $I$ in the figure. Second, an actual round-trip time between the correspondent node and the mobile node's new point of attachment during which the last of the aforementioned segments, $I$, gets delivered to the mobile node and the third duplicate acknowledgment, the one for segment $I$, reaches the correspondent node. This round-trip time might include some delay in the new access router's IP address resolution buffer, which segments $G$, $H$, and $I$ may experience.[8] It does not include any substantial forward buffering delays, however,

[8]If the IP address resolution buffering delay exceeds the inter-arrival time, $I_{TCP}$, for full-sized TCP segments, then segment $G$ would be replaced by segment $J$ in the buffer.




(a) in Fast Recovery mode (b) with retransmission timeout

Figure 6.37: Composition of handover phase in TCP SACK

given that traffic on the new transmission path is initially low in the experiments. Once the correspondent node has received the three duplicate acknowledgments, it performs a fast retransmit of the first lost segment, $C$, and enters Fast Recovery mode. This marks the end of the handover phase. Packet loss occurs exclusively at the new care-of address in figure 6.37(a), but in general, packets may also get lost at the old care-of address.

On the other hand, if the correspondent node fails to receive three duplicate acknowledgments from the mobile node prior to the expiration of the retransmission timer, the handover delay consists of the following two parts, as illustrated in figure 6.37(b): First, the period between the transmission of the first segment, $C$, that falls victim to handover-related packet loss and the reception of the last non-duplicate acknowledgment from the old care-of address, which is the acknowledgment for segment $B$ in the figure. Second, the length of the estimated retransmission timeout period at the time the correspondent node receives this last acknowledgment and resets the retransmission timer for the last time prior to the expiration of the timer. Forward buffering delays on the old transmission path as well as a considerable cushion for round-trip time variations are incorporated into the correspondent node's estimated retransmission timeout period, so loss recovery through the retransmission timer takes usually longer than loss recovery through three duplicate acknowledgments. Note that the mobile node acknowledges the retransmitted segment $C$ immediately, whereas it normally delays an acknowledgment for a the single segment. This is because the retransmission fills a gap in the mobile node's receive buffer space.

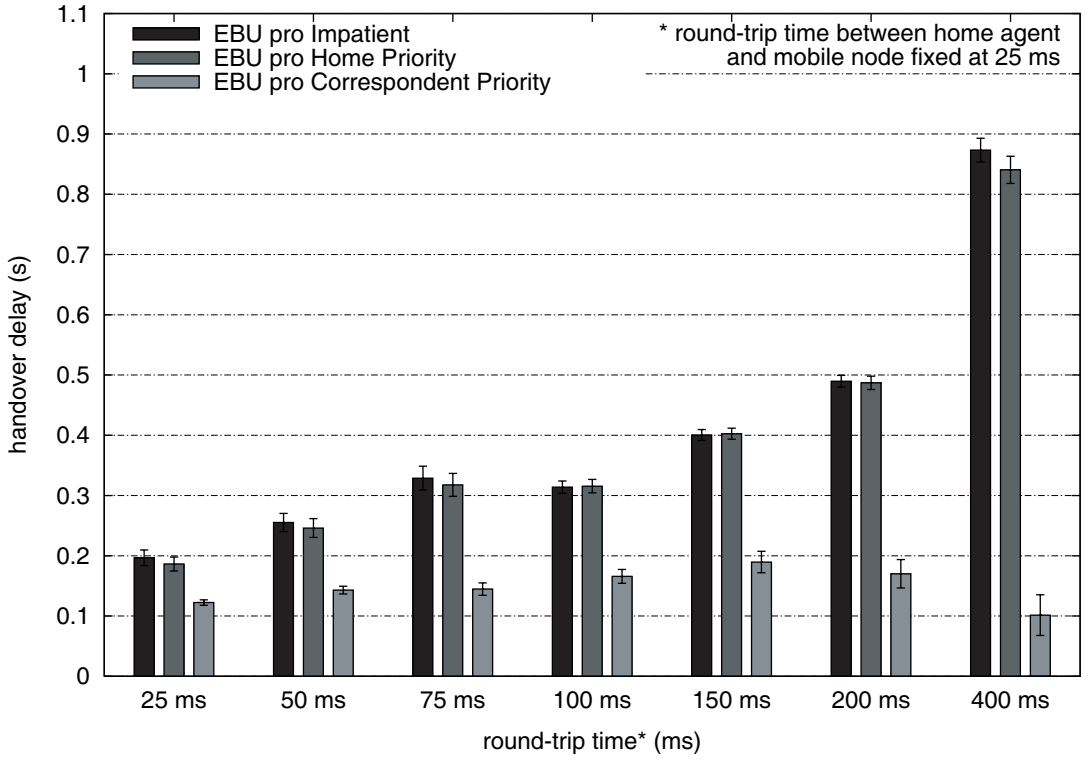



Figure 6.38: Mean handover delays in TCP SACK

### 6.5.6 Handover Delay in an Asymmetric Topology

The composition of the handover delay as described in section 6.5.5 for symmetric topologies still holds for asymmetric network topologies. Yet higher packet loss or an increased probability for handover-related retransmission timeouts may then lead to longer handover delays. Figure 6.38 summarizes the mean handover delays and 95% confidence intervals measured in an asymmetric network topology with base a round-trip time of 25 ms on all paths incident to the home agent, and a base round-trip time between the mobile node and the correspondent node varying from 25 ms to 400 ms. These measurements were obtained from the same experiments as those used in sections 6.5.2 and 6.5.4.

The asymmetry in the network topology leads to higher handover delays compared to a symmetric network topology when the mobile node runs Home Priority mode or—equivalent in this case—Impatient mode. In the experiments with base round-trip times of 75 ms or less between the mobile node and the correspondent node, this is primarily due to an increased probability for handover-related retransmission timeouts (compare figures 6.33 and 6.34). Retransmission timeouts typically go hand in hand with a delayed acknowledgment in these scenarios, as explained in section 6.5.4. A delayed acknowledgment, in turn, causes the correspondent node to reset its retransmission timer and possibly also to update its estimated retransmission timeout period to a higher value. Both increases the handover delay. Packet loss, too, is higher in asymmetric network topologies when the base round-trip time on the transmission path is 75 ms or less (see figures 6.27 and 6.31). But the impact of packet loss on the handover delay is then small compared to the impact of the higher retransmission timeout probability.

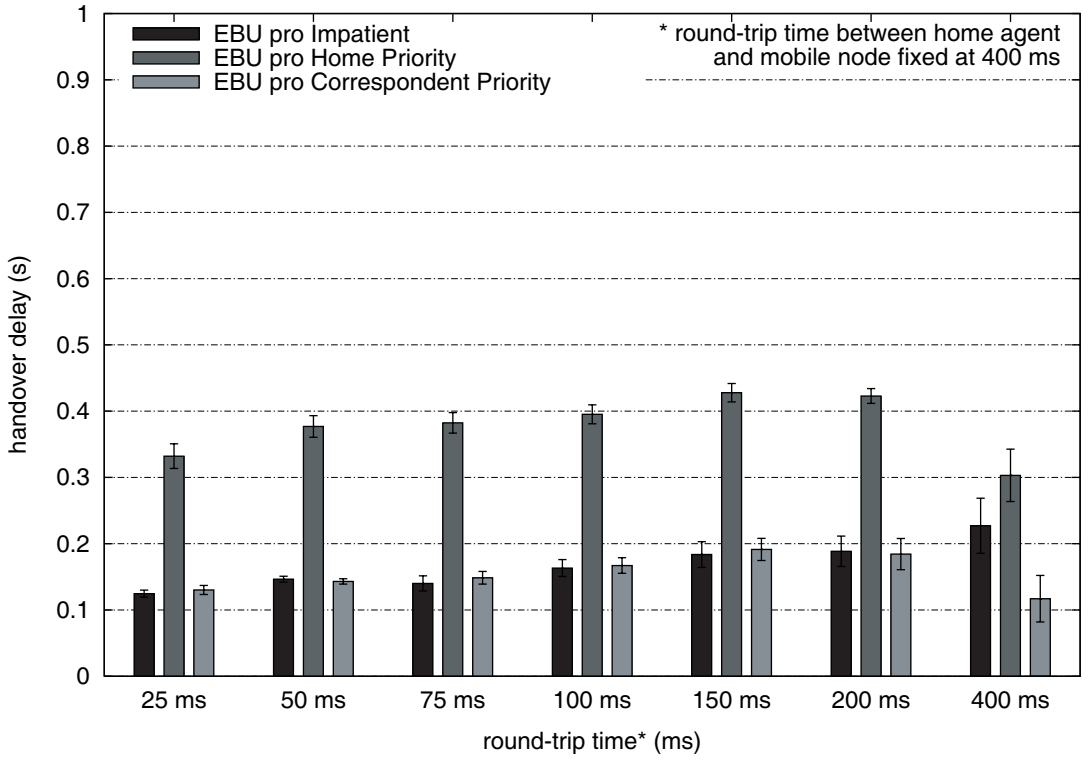



Figure 6.39: Mean handover delays in TCP SACK

Higher packet loss becomes the dominating reason for the observably longer handover delays in asymmetric network topologies compared to symmetric topologies as the base round-trip time between the mobile node and the correspondent node reaches 100 ms and beyond. The influence of packet loss thereby increases with the base round-trip time due to a growing discrepancy in packet loss in symmetric versus asymmetric network topologies. Handover-related retransmission timeouts have no effect on handover delays when the base round-trip time between the mobile node and the correspondent node is between 100 ms and 200 ms because the probability for them to occur is then negligible in both symmetric and asymmetric network topologies. The probability for a retransmission timeout is slightly higher in an asymmetric scenario where the base round-trip time is 400 ms. A timeout is here not necessarily accompanied by a delayed acknowledgments, but if it is, the delayed acknowledgment further contributes to the higher mean handover delay.

The relatively short home registration latency compared to the round-trip times between the mobile node and the correspondent node has no influence on the handover delay when the mobile node operates in Correspondent Priority mode since the initiation of the link layer handover is then always optimum for the redirection of segments from the old care-of address to the new one.

Figure 6.39 summarizes the average handover delays and 95% confidence intervals in experiments with asymmetric network topologies where the base round-trip time on the paths incident to the home agent is fixed at 400 ms, and the base round-trip time between the mobile node and the correspondent node ranges from 25 ms to 400 ms. The measurements indicate hat, with the exception of the scenarios with a base round-trip time of 400 ms on all paths, handover delays are very stable in face

of different base round-trip times between the mobile node and the correspondent node. This may surprise for the scenarios with Home Priority mode, because packet loss and retransmission timeout counts in the same experiments have been found to depend strongly on the round-trip time between the mobile node and the correspondent node (compare figures 6.32 and 6.35). The reason for the homogeneity across the measurements is that the handover delay is in Home Priority mode always given by the 400-ms latency of the home registration, as explained in the following.

Let $x$ be the base round-trip time between the mobile node and the correspondent node and $t_0$ the time at which the mobile node dispatches the Binding Update messages for the home agent and the correspondent node from the old link. The correspondent node then receives the Binding Update message at time $t_0 + x/2$ since no notable forward buffering delays exist on the transmission path in the direction towards the correspondent node. After the Binding Update message has been processed, the first TCP acknowledgment that the correspondent node receives triggers the transmission of the first segment to the mobile node's new care-of address. TCP acknowledgments for a steady stream of segments arrive at the correspondent node roughly every $2\,I_{TCP}$, so the correspondent node normally sends the first segment to the new care-of address at a time between $t_0 + x/2$ and $t_0 + x/2 + 2\,I_{TCP}$. This first segment gets lost since Home Priority mode forces the mobile node to stay on the old link until the home agent's Binding Acknowledgment message arrives. The transmission of the first segment therefore defines the beginning of the handover phase.

In the absence of forward buffering delays on the path between the mobile node and the home agent, the mobile node receives the home agent's Binding Acknowledgment message at time $t_0 + 400\ ms$, at which point it initiates the link layer handover towards the new link. After attaching to the new link, the mobile node receives the three segments cached in the new access router's IP address resolution buffer. These segments arrive out of order and hence solicit one duplicate acknowledgment each. The three duplicate acknowledgments, in turn, trigger a fast retransmit at the correspondent node half a round-trip time later, at time $t_0 + x/2 + 400\,\text{ms}$. This marks the end of the handover phase, leading to a rather constant handover delay of between 400 ms and $400\,\text{ms} - 2\,I_{TCP}$ for Home Priority mode.

Retransmission timeouts enable the correspondent node to retransmit lost packets more efficiently in these scenarios. This may happen when the round-trip time between the mobile node and the correspondent node—and with it the correspondent node's estimated retransmission timeout period—is small enough to cause a retransmission timeout before the mobile node arrives at the new link. The correspondent node then already retransmits the first loss before it receives the mobile node's three duplicate acknowledgments, which in most cases happens at a time the mobile node is still on the old link. Since the correspondent node has reduced its congestion window down to a single segment after the retransmission timeout, the retransmitted segment cannot be dropped from the new access router's IP address resolution buffer due to segments that succeed it. The segment is hence delivered once the mobile node arrives on the new link. Compared to fast-retransmit-based loss recovery, this spares the one round-trip time of handover delay during which the mobile node's three duplicate acknowledgments would otherwise propagate to the correspondent node and the retransmitted segment would be delivered to the mobile node. Loss

recovery after the handover generally proceeds efficiently as well because forward buffering delays on the old transmission path increase the correspondent node's estimated retransmission timeout period to a value long enough to prevent the correspondent node from falling into two consecutive retransmission timeouts after a handover. (Recall from section 6.4.2 that two consecutive retransmission timeouts would reduce TCP's slow-start threshold to the minimum of 2 full-sized segments and thereby cause the congestion window to open only very slowly in Congestion Avoidance mode.)

The smaller handover delays for Home Priority mode and Correspondent Priority mode in the experiments with a round-trip time of 400 ms are due to some handovers without any packet losses and, hence, zero handover delay.

Impatient mode is equivalent to Correspondent Priority mode in all experiments where the base round-trip time between the mobile node and the correspondent node is 200 ms or less. The latency of a correspondent registration is then always smaller than the latency of a home registration. On the other hand, forward buffering delays on the path from the correspondent node to the mobile node lead to a later completion of the correspondent registration when the base round-trip time on all paths is 400 ms. Impatient mode then becomes equivalent to Home Priority mode.

### 6.5.7 Credit Availability

A TCP session's demand for credit during a handover may be much higher in proactive mobility management than it is in reactive mobility management due to a higher number of segments that the correspondent node sends to a new care-of address in Unverified state. A proactive early Binding Update message enables the correspondent node to direct segments to the new care-of address at a time these segments are still clocked by acknowledgments received from the old care-of address. The correspondent node thus initially continues to send at the same rate as it did prior to processing the early Binding Update message. In contrast, with reactive mobility management, the amount of credit consumed during a handover is typically equivalent to the single segment which the correspondent node sends when entering Slow Start mode after a retransmission timeout.

The amount of credit that a mobile node has available over time is studied below for a symmetric network topology. The study thereby focuses on the use of proactive mobility management in Correspondent Priority mode. This mode provides a reasonable basis for evaluating whether Credit-Based Authorization can ensure sufficient credit availability also for proactive mobility management because handover-related credit consumption in a symmetric network topology is typically highest in this mode. Correspondent Priority mode ensures that all segments in flight towards an old care-of address are received by the mobile node prior to the link layer handover, so the acknowledgments that the mobile node thereby sends from the old point of attachment clock the segments that the correspondent node sends to the new care-of address while this is in Unverified state. Since the mobile node receives all segments at the old care-of address, a maximum number of segments is sent to the new care-of address before it moves to Verified state, and a maximum amount of credit is consumed. A separate evaluation of Home Priority mode is left to section 6.5.8, where the effect of different home and correspondent registration latencies are emphasized by different asymmetric network topologies.

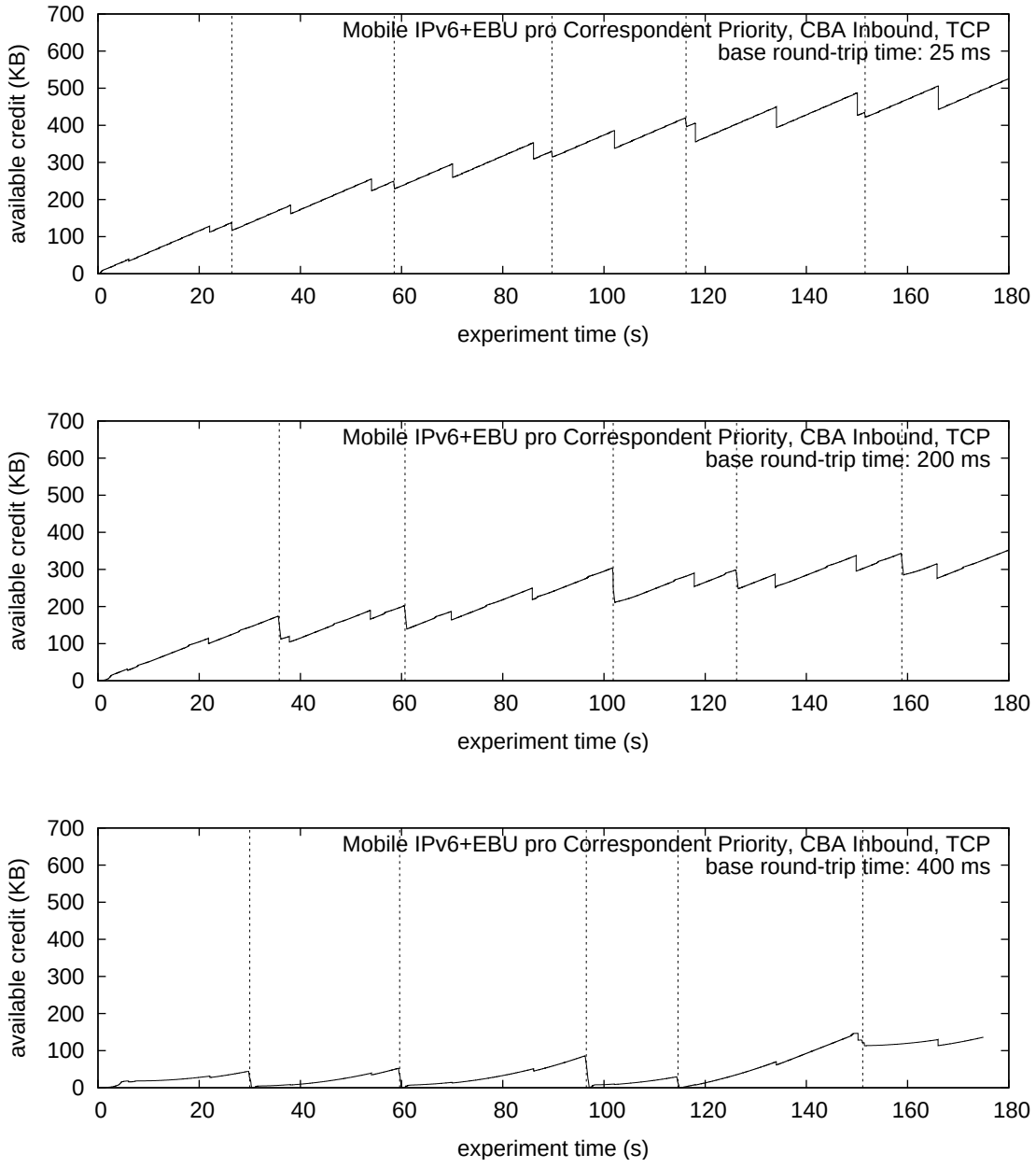



Figure 6.40: Credit availability in Inbound mode for TCP SACK

Figure 6.40 shows the mobile node's available credit as a function of time from three experiments with Credit-Based Authorization in Inbound mode. The experiments were conducted in symmetric network topologies with base round-trip times of 25 ms in the top chart, 200 ms in the middle chart, and 400 ms in the bottom chart. The measurements clearly indicate that the higher the base round-trip time, the lower the amount of credit available to the mobile node over time. When the base round-trip time is as high as 400 ms, lack of credit may even cause the correspondent node to drop some payload packets which it could otherwise send to the mobile node's unverified care-of address.

There are multiple reasons for the low amount of available credit in the experiments with long base round-trip times. For one, the period between the time the cor-

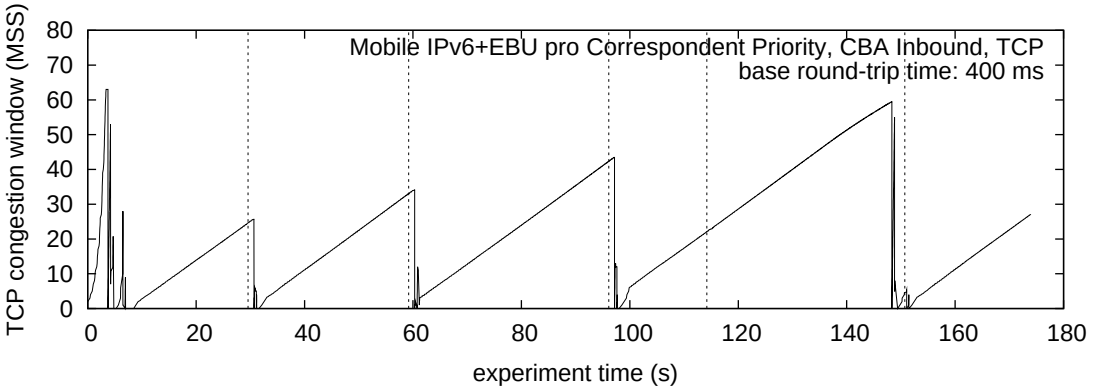



Figure 6.41: Progression of the TCP congestion window size over time in the same experiment as shown in the bottom chart of figure 6.40

respondent node registers the mobile node's new care-of address and the time the correspondent node receives the mobile node's standard Binding Update message, which confirms the mobile node's reachability at the new care-of address, grows with the base round-trip time. Each of the segments that the correspondent node sends throughout this time takes up some of the credit available, so the amount of credit consumed per handover grows with the base round-trip time. Second, the bug in FreeBSD's implementation of TCP selective acknowledgments that was mentioned in section 6.4.6 leads to sporadic handover-unrelated retransmission timeouts when the bandwidth-delay product on the transmission path is close to the mobile node's receive buffer capacity. Each retransmission timeout causes a pause in TCP throughput and, consequently, a temporary stalling in credit acquisition. This leads to the more sluggish credit acquisition in the scenario with 400-ms base round-trip times.

A third reason for the observed shortage in credit when the base round-trip time on the transmission path is long are the mobile node's relatively frequent handovers in the experiments under consideration. The time between successive handovers is randomly distributed between 15 s and 45 s, so there are only 30 s on average during which the mobile node can replenish its credit in preparation for the next handover. Moreover, the long base round-trip time increases the time TCP needs to fill the bandwidth available on the new transmission path to beyond the pause time between successive handovers. TCP therefore never reaches the throughput that would theoretically be possible on the transmission path, reducing the rate at which credit can be acquired.[9] Figure 6.41 illustrates this observation with a trace of TCP's congestion window size that was taken from the same experiment shown in the bottom chart of figure 6.40. The vertical lines across the trace indicate the times at which the mobile node changes IP connectivity. Obviously, the increase of the congestion window spans the entire time between two handovers. Packet loss then terminates any further growth so that throughput never reaches the maximum possible. Additional experiments where the mean pause time between successive

[9]Recall that the mobile node sends a cumulative acknowledgment for every second segment, so the correspondent node's congestion window grows only by one segment size per twice the actual round-trip time. The actual round-trip time, in turn, may include up to $14\,I_{TCP} = 121.2\,\text{ms}$ of forward buffering delay in addition to the base round-trip time.

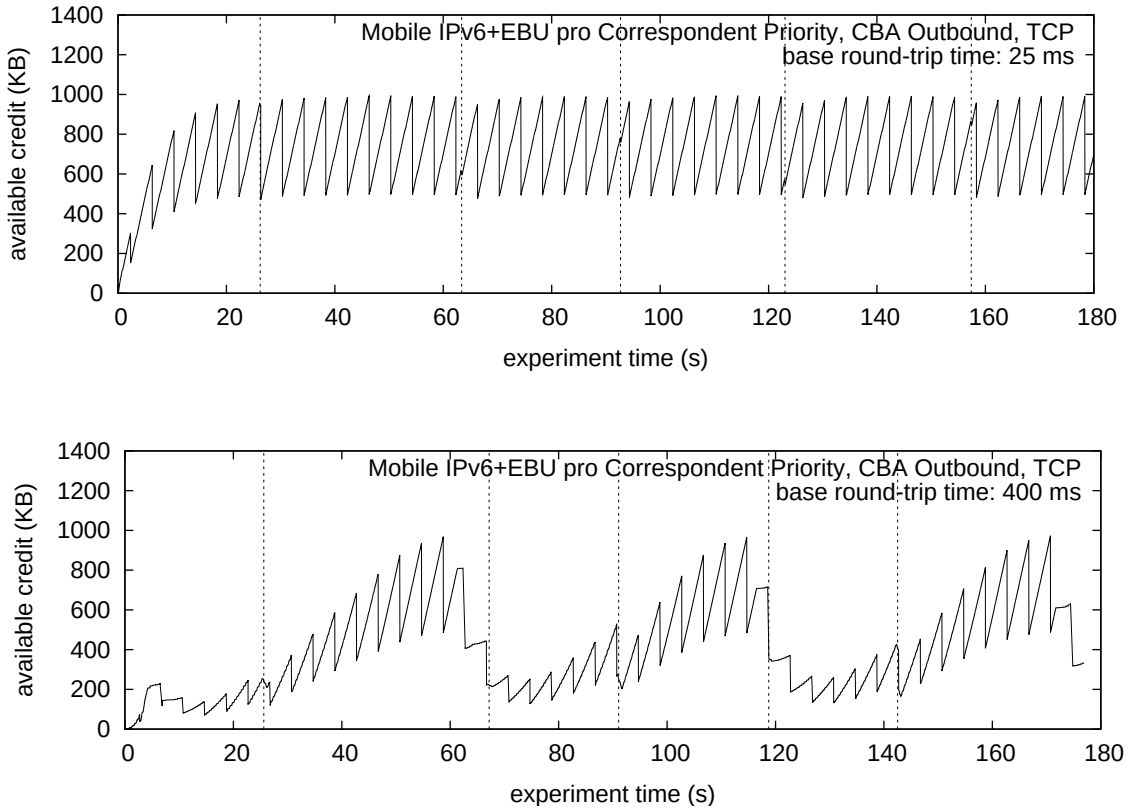



Figure 6.42: Credit availability in Outbound mode for TCP SACK

handovers was increased from 30 s to 60 s have shown that the mobile node may then obtain sufficient credit even when the base round-trip time is 400 ms.

Figure 6.42 shows the credit availability in the same symmetric network topologies as above when Credit-Based Authorization is operated in Outbound mode. The mobile node again uses proactive mobility management in Correspondent Priority mode. Credit is now acquired more quickly than it is with Inbound mode. The more rigid aging compensates this effect partly, but overall, the mobile node's available credit now proves sufficient even for a base round-trip time of 400 ms on the transmission path. This demonstrates the advantage of Outbound mode over Inbound mode in scenarios that impede.

### 6.5.8 Credit Availability in an Asymmetric Topology

The amount of credit a mobile node consumes during a handover is partly determined by the time the mobile node remains on the old link while segments from the correspondent node already arrive at the new, unverified care-of address. Internet telephony applications continue to send throughout this entire period, so the amount of credit meanwhile consumed is proportional to the length of the period. On the other hand, for TCP connections, the amount of credit consumed is bound by the correspondent node's available window at the time of the handover since this limits the amount of data that the correspondent node can send without receiving an acknowledgment. The available window, in turn, depends on the bandwidth-delay product and forward buffering capacity of the old transmission path. Credit

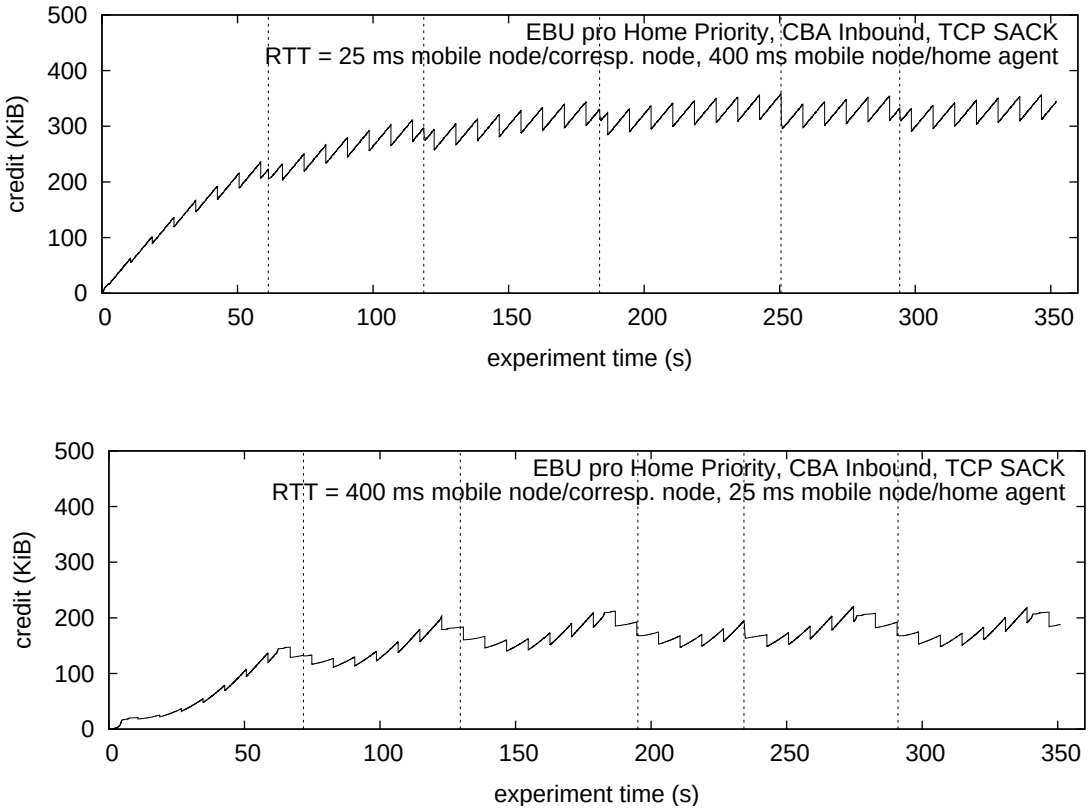



Figure 6.43: Credit availability in Inbound mode for TCP SACK

consumption due to network asymmetry in TCP connections hence cannot grow to the same extent as it can for UDP-based Internet telephony sessions, where no transport-layer limitation exists on the amount of data the correspondent node sends to a care-of address in Unverified state. The following evaluation studies the impact that differences between the home and correspondent registration latencies have on the mobile node's available credit during a TCP-based file transfer from the correspondent node to the mobile node. The evaluation focuses on Home Priority mode. The initiation time of the mobile node's link layer handover is hence always determined by the home registration latency, which leads to an increasingly adverse impact on the TCP connection between the mobile node and the correspondent node as the asymmetry in the network topology grows.

Figure 6.43 shows a mobile node's available credit as a function of time from two experiments with Credit-Based Authorization in Inbound mode. The chart at the top was obtained from an asymmetric network topology where the base round-trip time between the mobile node and the correspondent node is 25 ms, and the base round-trip time on all paths incident to the home agent is 400 ms. Although handover delays are in this example increased by the mobile node's long stay on the old link relative to the redirection of segments from the old care-of address to the new one, handover-related credit consumption is actually low for two reasons. First, the small bandwidth-delay product on the transmission path limits the correspondent node's congestion window and thus the amount of data the correspondent node can send to the new care-of address without getting an acknowledgment. Second, the long time

that the mobile node stays on the old link after it has received the last segment at the old care-of address causes the correspondent node to fall into a retransmission timeout. Data transfer subsequently resumes with only one segment per round-trip time, so the amount of data that the correspondent node can send to the new care-of address shortly after the retransmission timeout is low as well. The downside of the low throughput in terms of credit availability is that new credit can only be acquired at a reduced rate after the new care-of address has moved to Verified state. In fact, the acquisition of new credit initially stalls after the handover due to the retransmission timeout. Credit may then actually reduce slightly at the end of an aging interval. This is not a problem on the longer term, however. The exponential congestion window growth in Slow Start mode facilitates an expedited throughput resumption, especially with the short round-trip time between the mobile node and the correspondent node. This allows the mobile node to eventually renew the credit consumed previously.

The credit consumed during a handover is also low in the reverse network topology, where the base round-trip time between the mobile node and the correspondent node is 400 ms, and the base round-trip time on all paths incident to the home agent is 25 ms. Since the mobile node pursues the link layer handover already when it receives the home agent's Binding Acknowledgment message, it here loses a significant part of the segments which the correspondent node sends to the old care-of address until it receives the mobile node's Binding Update message. This reduces the number of non-duplicate acknowledgments that the mobile node sends from the old link after having dispatched the Binding Update messages, and hence the number of new segments that the correspondent node can proactively direct to the new care-of address. The consequence is only moderate credit consumption during the handover, as can be seen in the bottom chart of figure 6.43. The amount of data sent to the new care-of address while in Unverified state may further reduce when the correspondent node contracts its congestion window upon receiving the three duplicate acknowledgments that the mobile node typically transmits from the new point of attachment for received out-of-order segments. On the longer term, TCP throughput in this case recovers more slowly from the handover-related reduction than it does in the scenario shown in the top chart of the figure due to a longer round-trip time on the transmission path. This and the lower average TCP throughput on paths with long round-trip times (see section 6.4.1) lead to a lower amount of credit available.

Figure 6.44 shows the evolution of the mobile node's available credit as a function of time when Credit-Based Authorization is operated in Outbound mode. The network topologies in which the two charts were generated are the same as above, and the mobile node again runs proactive mobility management in Home Priority mode. Credit lost during handover is replenished faster with Outbound mode than it is with Inbound mode due to the higher transmission rate of the correspondent node compared to the mobile node's. On the other hand, credit aging in Outbound mode is more rigid as well, leading to a notable credit reduction when the acquisition of new credit pauses while the mobile node's care-of address is in Unverified state. This does not have a serious impact when the round-trip time on the transmission path is small, as shown in the top chart of figure 6.44. Credit acquisition then usually picks up quickly after the handover due to a rapid opening of the correspondent node's congestion window. However, the longer the round-trip time on the transmission

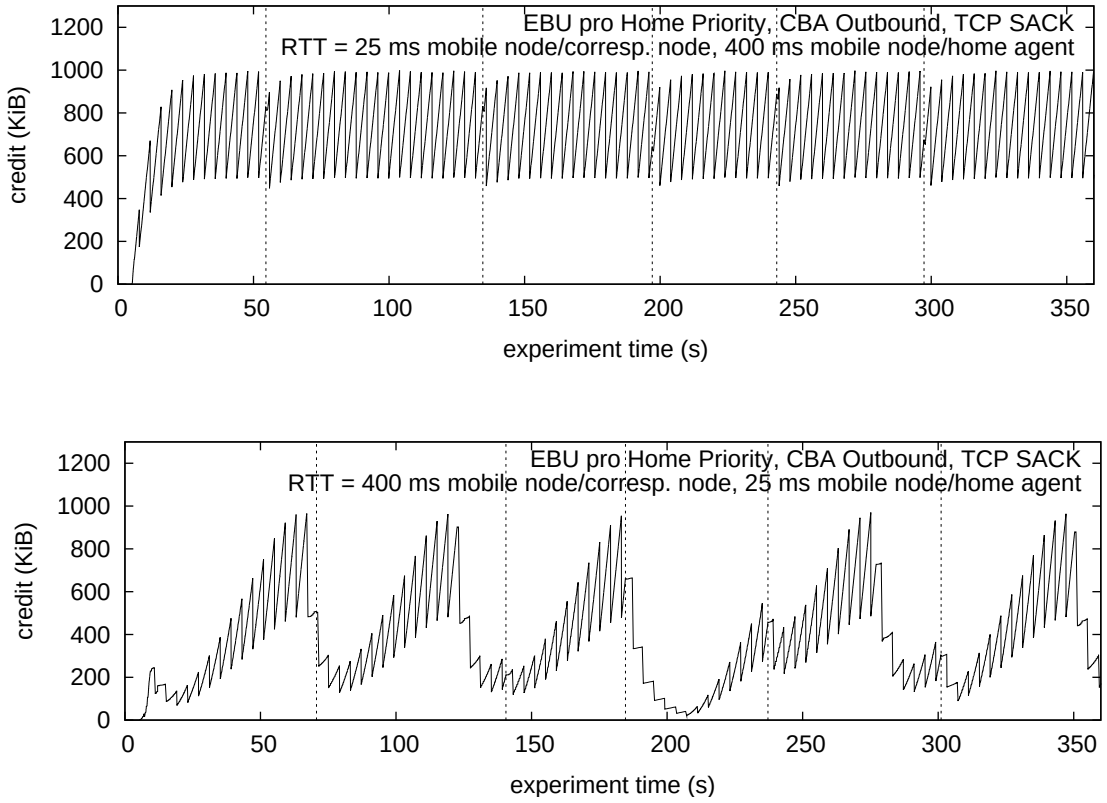



Figure 6.44: Credit availability in Outbound mode for TCP SACK

path gets, the longer it takes for TCP to ramp up its throughput again after a handover. The acquisition of new credit then accelerates more slowly as well so that the impact of credit aging is stronger. The deeper cuts in the available credit in the bottom chart of figure 6.44 evidence this behavior.

Asymmetry in the network topology has no effect if the mobile node operates in Correspondent Priority mode as the round-trip time between the mobile node and the correspondent node then does not affect the timing of the mobile node's link layer handover. The performance of Impatient mode is equivalent to that of either Home Priority mode or Correspondent Priority mode, depending on whether the home registration latency is shorter than the correspondent registration latency, or vice versa, respectively. Correspondent Priority mode and Impatient mode are hence not discussed separately in this section.

## 6.6 Summary

The Internet telephony applications used in the conducted experiments generate payload packets in equidistant intervals. Measurements for reactive mobility management with conservative Mobile IPv6, optimistic Mobile IPv6, and Mobile IPv6 with Early Binding Updates accordingly demonstrate a direct impact that Mobile IPv6 optimizations at the IP layer have on application layer handover delays and handover-related packet loss. Since the traffic volume sent in either direction is the same, a mobile node acquires credit at the same rate as it consumes credit during

a handover. Available credit was therefore found to be sufficient in both Inbound mode and Outbound mode during all measured handovers, although the amount of credit actually available at any given time was usually lower in Outbound mode than in Inbound mode due to more rigid credit aging.

Proactive mobility management can eliminate handover delays and handover-related packet loss entirely for Internet telephony if the actual round-trip times are the same on the old and the new transmission path, and the time for the link layer handover is chosen well. Packet delivery via the new care-of address then begins just when it ceases via the old care-of address. Mobile IPv6 Correspondent Priority mode enables the mobile node to pursue the link layer handover at this optimum time. Handover delays and packet loss increase when the link layer handover occurs sooner or later than the optimum time, as was demonstrated by the measurements with Home Priority mode.

TCP-based file transfers always suffer the loss of about one window worth of data during handover, and hence experience the expiration of the retransmission timer. It is the number of consecutive retransmission timeouts that determines the handover delay and, more importantly, the pace at which TCP repairs this loss and adapts to the bandwidth available on the new transmission path afterwards. If transmission resumes after a single retransmission timeout, then throughput begins to increase exponentially after the handover. This permits quick utilization of available bandwidth. However, TCP throughput increases only linearly after two consecutive retransmission timeouts, which leads to sluggish loss repair and long underutilization of the available bandwidth.

Mobile IPv6 optimizations may fail to improve the handover performance of TCP in scenarios where the actual round-trip time on the old transmission path prior to handover is substantially longer than the actual round-trip time on the new transmission path. Since TCP estimates the retransmission timeout period based on the actual round-trip time of the old transmission path, but reactive Mobile IPv6 signaling latencies depend on the actual round-trip time of the new transmission path, all Mobile IPv6 variants may then complete the binding update before the retransmission timer expires for the first time. This effect was observed in experiments where forward buffering delays extended the actual round-trip time on the old transmission path to beyond the actual round-trip time on the yet-uncongested new transmission path.

Sufficient credit has been found to be available during handover in all measured scenarios. Credit acquisition is slower in the scenarios with long round-trip times, however, because the speed at which TCP adapts to the bandwidth available on a new transmission path—both in Slow Start mode and Congestion Avoidance mode—decreases as the round-trip time grows.

Compared to reactive mobility management, proactive mobility management usually averts the loss of a full window of data. This enables TCP to recover from packet loss in Fast Recovery mode, hence without waiting for the expiration of a retransmission timer. Zero packet loss was hardly ever observed, however, due to forward buffering on the old transmission path causing an overlap between the delivery of payload packets on the old and new transmission paths. A full window worth of data is lost only when the mobile node leaves the old access link very early after sending

an early Binding Update message. Lack of acknowledgments from the mobile node then causes the correspondent node to stall data transmission and eventually run into a retransmission timeout.

Credit consumption is higher for proactive mobility management than it is for reactive mobility management because the mobile node initiates the binding update for a new, unverified care-of address now already while the mobile node is still on the old link.

# 7. Conclusions

New applications and changing needs of Internet users call for an efficient and secure mobility management in an increasingly heterogeneous networking environment, and a lot of effort has recently gone into suitable solutions. One of the challenges encountered in this context is a potential for unprecedented redirection-based flooding attacks, for which the ability to redirect packets from one IP address to another could be misused if appropriate protection is not in place. An unprotected mobility protocol would enable an attacker to trick any correspondent node supporting the protocol into participating in a redirection-based flooding attack against an arbitrary victim. The attacker could make a correspondent node send a much larger amount of data to the victim than the attacker sends itself. Among the flooding attack types that exist today, only distributed denial-of-service attacks can compare to the level of amplification that redirection-based flooding attacks could provide, and even those bring the extra cost for the attacker to develop and distribute viral software. The threat of redirection-based flooding pertains to any type of mobility protocol at any protocol layer, and more generally, it also affects multi-homing protocols.

A widely accepted technique that modern mobility protocols use to prevent redirection-based flooding attacks is to verify a mobile node's reachability at a new IP address before permitting payload packets to be sent to that IP address. This approach, standard reachability verification, provides the desired level of security, yet it introduces new handover delays and thus has an adverse impact on application quality. Besides, the need to be reachable at a new IP address before packets can be sent to that IP address prevents more efficient proactive mobility management.

The goal of this thesis was to find an alternative to standard reachability verification that is as secure, but without extra handover delays. The alternative was furthermore required to be inexpensive to deploy, universally applicable without special infrastructural requirements, generic enough to be easily integrable into any mobility protocol, and suitable for proactive mobility management. An analysis has shown that concurrent reachability verification is most suitable according to these objectives, provided that it be supplemented by a reliable mechanism to secure the period during which packets are sent to a yet-to-be-verified IP address. This thesis has developed Credit-Based Authorization for this purpose, a mechanism that ac-

companies concurrent reachability verification without compromising it in terms of the aforementioned objectives.

Complementary to the development of Credit-Based Authorization are the enhancements of Mobile IPv6 route optimization, which this thesis has contributed. The analysis of Mobile IPv6 with respect to tuning opportunities within the boundaries of the protocol specification is of immediate practical value for implementors, helping them produce more efficient, but still standard-compliant software. This work originated with the observation that popular Mobile IPv6 implementations unnecessarily increase handover delays through all-too-conservative behavior. Since standard Mobile IPv6 route optimization as such is incompatible with concurrent reachability verification, Early Binding Updates have been designed as a set of modifications to the standard protocol signaling that facilitate the integration of concurrent reachability verification and Credit-Based Authorization. All optimizations together reduce the IP layer handover delay of Mobile IPv6 route optimization to the minimum possible for reactive, end-to-end mobility management. Extensions of Early Binding Updates for proactive mobility management help reduce handover delays even further.

An effort that went hand in hand with the work on Early Binding Updates and Credit-Based Authorization was the continued participation in related IETF working groups with an objective to help eliminate handover-related re-configuration latencies in the IPv6 protocol suite. This effort was driven by the objective for a more efficient reactive mobility management and to also afford proactive mobility management [139, 138, 141]. On the basis of completed and ongoing activities within the IETF, this thesis has shown how the proposed Mobile IPv6 optimizations integrate with optimizations for Neighbor Discovery and Stateless Address Autoconfiguration. The thesis has further proposed a way of using IEEE 802.21 Media Independent Handover Services to synchronize handover-related activities at a mobile node's link and IP layers, a prerequisite for efficient mobility management. The result is a comprehensive protocol set-up which, driven by events at the link layer, enables IP layer handovers with minimum delays.

The thesis has further demonstrated how Mobile IPv6 with Early Binding Updates can be used to support proactive mobility management. The proposed proactive mobility management procedure builds on Media Independent Handover Services for cross-layer synchronization, the anticipation of changes in link layer attachment, and the proactive auto-configuration of an IP address for use on a potential target access link. Since Media Independent Handover Services require network support to facilitate proactive IP address auto-configuration, an additional, autonomous fallback mechanism was devised for mobile nodes to apply in scenarios where such network support is unavailable. The thesis also shows that the handover performance of proactive mobility management depends substantially on the link layer handover initiation time chosen by the mobile node. Selection of the right time can notably improve the performance relative to reactive mobility management.

All proposed Mobile IPv6 enhancements were implemented for the FreeBSD operating system and thoroughly evaluated in an experimental testbed. The results have evidenced a strong advantage of concurrent reachability verification over standard reachability verification, both for delay- and loss-sensitive UDP-based Internet telephony applications, as well as for classic TCP-based file transfers. Additional

experiments have demonstrated the benefits of proactive over reactive mobility management in a variety of network topologies.

This thesis concludes with some recommendations for further research in the area of efficient and secure mobility support. One interesting topic would be to follow up on the work presented herein by evaluating the security and the performance of alternative credit assignment, consumption, or aging algorithms. Are there secure algorithms that offer better credit availability to mobile nodes that behave legitimately, preferably for a large set of application layer traffic patterns? Could a correspondent node use statistics on incoming and outgoing data to tune its credit aging function dynamically? Could a variation of Inbound mode for credit assignment, combined with a novel credit aging algorithm, replace Outbound mode as a secure algorithm for credit assignment and interoperate well with various kinds of application layer traffic patterns? And finally, what maximum handover rate do these credit assignment, consumption, and aging functions support without causing lack of credit during a handover?

Given that a typical network protocol stack may include multiple mobility or multi-homing protocols at different layers, it would be interesting to prototype, and evaluate the benefits of, an operating system service that centrally provides the functionality of Credit-Based Authorization for all of these protocols. This could reduce implementation complexity as well as signaling overhead when multiple protocols use spot checks. The Credit-Based Authorization service would require an interface for mobility and multi-homing protocols to associate a remote node's on-link IP address with a binding identifier, to update a binding identifier to a different on-link IP address, and to toggle an on-link IP address between Unverified and Verified states. The service would independently maintain a credit account for each binding identifier, change the credit as packets are sent to or received from the remote node to which this binding identifier belongs, or drop packets that cannot be sent to an unverified on-link IP address due to lack of credit.

Beyond these opportunities for continued work on Credit-Based Authorization itself, further research is necessary in other parts of the network stack, so as to realize smooth handover operations from the link layer all the way up to the application. This foremost includes reductions in the link layer handover delay. It also encompasses handover indications to upper-layer protocols and the appropriate actions in response to them [43]. For proactive mobility management, more work is necessary on handover prediction so that proactive signaling can be initiated in time. Yet another challenge is the optimization and, probably more importantly, harmonization of authentication and authorization mechanisms for network access, which are of fundamental importance in many real-life deployments. Any efficient and secure mobility solution is of limited use if mobile nodes are required to go through a time-consuming reauthentication process whenever they move between administratively discontiguous access networks—especially if this process involves human interaction. Automated, fast, and widely deployed mechanisms are needed. Certainly, the path towards truly efficient mobility management continues to bear challenging opportunities for further research.

# A. Impact of Link Layer Handover Delays

An important assumption underlying the evaluation in chapter 6 is that the link layer can switch between different points of attachment instantaneously. No extra handover delay was therefore added at the link layer beyond the latency of filter reconfigurations (see section 6.1.2). The effective link layer handover delays were consequently negligible. The motivation for this was that sufficiently small link layer handover delays are prerequisite for true support of interactive real-time applications in mobile environments, and research efforts are under way to make them a reality, for example [118, 28]. On the other hand, link layer handover delays today are still considerable for many access technologies [80, 19, 28], limiting instantaneous access point switches to dual-interfaced mobile nodes [109, 22]. The purpose of this appendix is to shed light on the impact of link layer handover delays on the application performance, helping to assess the benefit of Mobile IPv6 optimizations in the presence of non-zero link layer handover delays.

For the purpose of brevity, this appendix only considers reactive mobility management. The amount of credit available during a handover is likewise not considered because the evaluation in chapter 6 has shown that the mobile node in general has a sufficient amount of credit available during handover anyway.

## A.1 Internet Telephony

Figure A.1 juxtaposes UDP handover delay averages and 95% confidence intervals of an Internet telephony application that were measured in experiments with conservative Mobile IPv6, optimistic Mobile IPv6, and Mobile IPv6 with Early Binding Updates, where the link layer handover delay was varied from 0 ms to 400 ms. The experiments were conducted in a symmetric network topology with a fixed base round-trip time of 100 ms. The measurements demonstrate that, for the constant-bit-rate Internet telephony traffic, an additional link layer handover delay directly translates into an increment in UDP handover delays of the same size. As the round-trip time remains constant in the figure, signaling latencies for each Mobile

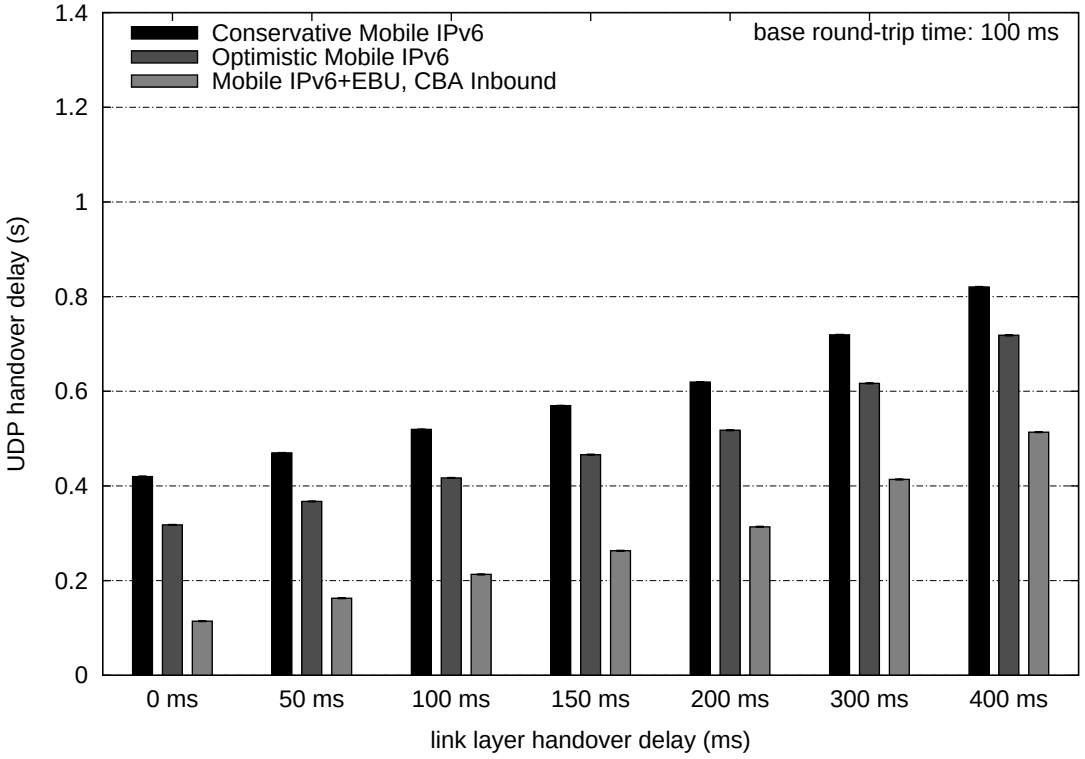



Figure A.1: Mean UDP handover delay as a function of link layer handover delay in a symmetric network topology with 100-ms base round-trip times

IPv6 variant do not change as well. The difference in UDP handover delays between any two Mobile IPv6 variants is hence the same for all link layer handover delays.

Figure A.2 compares the above UDP handover delay measurements with results from experiments where the base round-trip time was increased to 200 ms. Mobile IPv6 signaling latencies are now longer, leading to a greater benefit of Mobile IPv6 optimizations. The impact of link layer handover delays on the overall UDP handover delay is independent of the round-trip time, however.

## A.2 TCP File Transfers

Figure A.3 illustrates the composition of the TCP handover delay in the presence of link layer handover delays. The mobile node in part (a) of the figure runs conservative Mobile IPv6, so it takes the link layer handover delay plus a few round-trips of Mobile IPv6 signaling until the mobile node can inform the correspondent node about its new care-of address. The TCP handover delay is then likely to span more than one retransmission timeout period. The mobile and correspondent nodes in part (b) of the figure support Mobile IPv6 with Early Binding Updates so that the mobile node can send an early Binding Update message immediately after it has attached to the new access link. The TCP handover delay thus depends on the link layer handover delay only. And provided that the link layer handover delay does not exceed the correspondent node's estimated retransmission timeout period, the binding update completes before the retransmission timer expires for the first time.

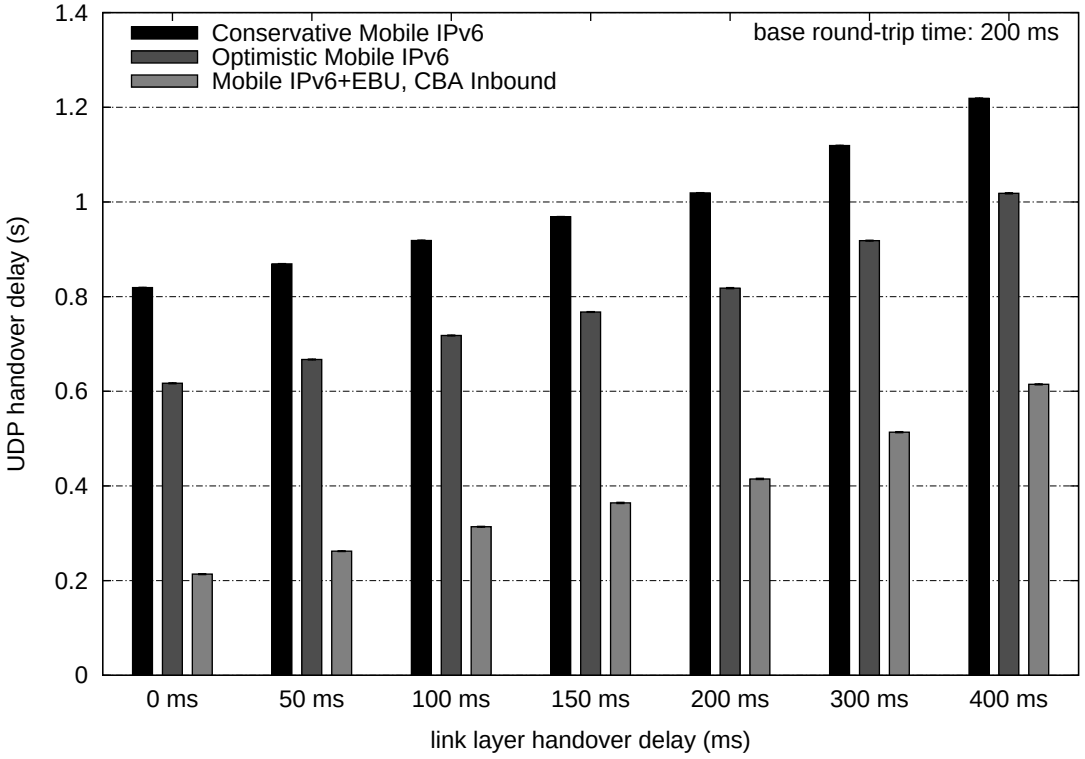
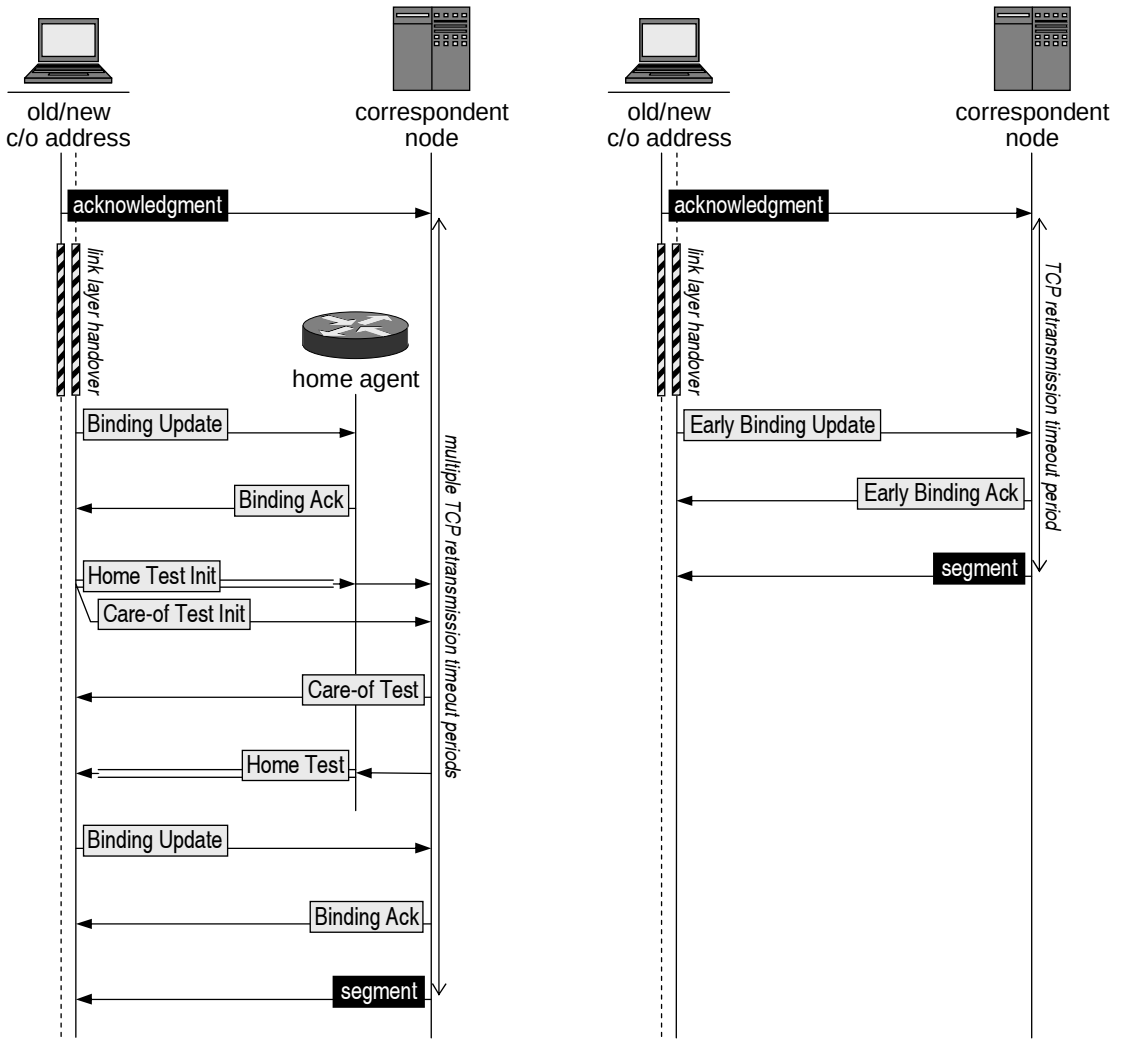



Figure A.2: Mean UDP handover delay as a function of link layer handover delay in a symmetric network topology with 200-ms base round-trip times

Figure A.4 summarizes handover delay averages and 95% confidence intervals measured in TCP SACK file transfer experiments per Mobile IPv6 variant per link layer handover delay. The experiments were conducted in a symmetric network topology with a fixed base round-trip time of 100 ms. For conservative and optimistic Mobile IPv6, the TCP handover delays increase steadily with the handover delay at the link layer, with temporarily stable values for optimistic Mobile IPv6 when the link layer handover delay is between 100 ms and 200 ms. Mobile IPv6 with Early Binding Updates makes the TCP handover delays more robust to increasing link layer handover delays. TCP handover delays then vary relatively little as the link layer handover delay grows to 150 ms, and they make a leap towards another relatively stable level as the link layer handover delays reaches 300 ms and beyond.

The measured TCP handover delays go hand in hand with the number of retransmission timeouts that TCP goes through during a handover in the same experiments. The averages and 95% confidence intervals of these are shown in figure A.5. Accordingly, the steady growth in TCP handover delays for conservative and optimistic Mobile IPv6 is due to an increasing probability that TCP undergoes two or even three consecutive retransmission timeouts. It may surprise, though, that the multiple round-trips of Mobile IPv6 signaling may still complete within a single retransmission timeout period if link layer handover delays are 50 ms or less. The reason for this is the combination of two things: One is the prolonging effect of forward buffering delays on the retransmission timeout period. Recall from section 6.4.2 that TCP's estimation of the retransmission timeout period incorporates forward buffering on the old transmission path and hence usually exceeds the configured base round-trip




(a) Conservative Mobile IPv6 (b) Mobile IPv6 with Early Binding Updates

Figure A.3: Impact of link layer handover delay on TCP handover delay

time. The retransmission timeout period may thus actually reach a multiple of the base round-trip time if the latter is small. In contrast, Mobile IPv6 signaling takes an uncongested new transmission path, so it experiences none or only minor forwarding delays. This effect reduces the number of consecutive retransmission timeouts per handover. If, in addition, the mobile node delays its acknowledgment for the last segment received before the handover, then the binding update may actually complete within one retransmission timeout despite the link layer handover delay and the several round-trips of Mobile IPv6 signaling.

Since the maximum delay for acknowledgments was set to 100 ms in these experiments, delayed acknowledgments end up being dropped at the mobile node's link layer when the link layer handover delay is 100 ms or longer. This is why a single retransmission timeout is then insufficient for both conservative and optimistic Mobile IPv6. It should be noted that delayed acknowledgments are dropped even when the maximum acknowledgment delay and the link layer handover delay are both 100 ms long because the acknowledgment delay begins already when the mobile node re-

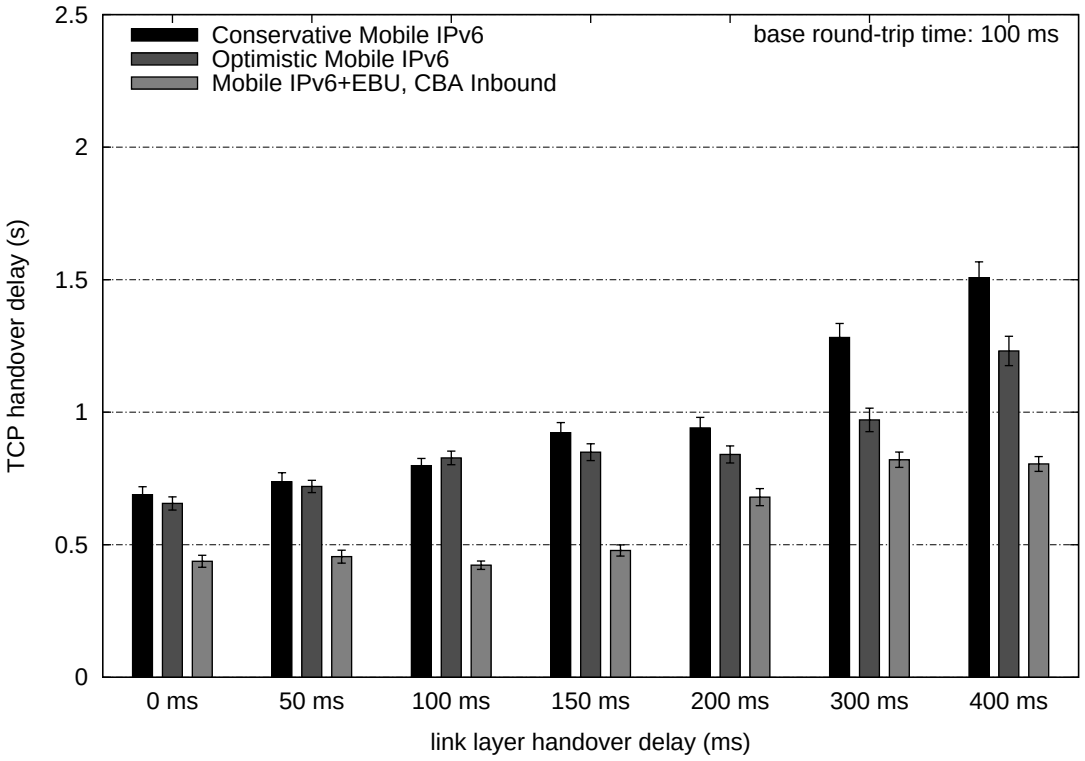



Figure A.4: Mean TCP handover delay as a function of link layer handover delay in a symmetric network topology with 100-ms base round-trip times

ceives the last segment prior to handover and therefore usually a bit earlier than the link layer handover.

Conservative Mobile IPv6 is sporadically responsible for three consecutive retransmission timeouts per handover when the link layer handover delay is between 150 ms and 200 ms. This was found to happen exactly in those cases where packet loss occurs a few round-trip times before the handover. TCP contracts its congestion window when it detects the packet loss, reducing the amount of outstanding data and, consequently, forward buffering delays on the transmission path. The correspondent node's round-trip time measurements then shrink as well, and so does the estimated retransmission timeout period. The retransmission timer hence expires earlier during handover, and more consecutive retransmission timeouts set in before the mobile node has updated the binding at the correspondent node. The same occasionally happens for optimistic Mobile IPv6 in the experiments with link layer handover delays of 300 ms. Three consecutive retransmission timeouts become increasingly likely for conservative and optimistic Mobile IPv6 as the link layer handover delay reaches 300 ms and beyond. The total TCP handover delay is then seven times as long as a single retransmission timeout period due to the exponential back-off.

For Mobile IPv6 with support for Early Binding Updates, a single retransmission timeout is always sufficient if the link layer handover delay is 100 ms or less, and it is still sufficient in most cases if the link layer handover delay is 150 ms. This is again because forward buffering delays on the old transmission path increase TCP's estimated retransmission timeout period to beyond the link layer handover delay. The mobile node's early Binding Update message then reaches the correspondent node

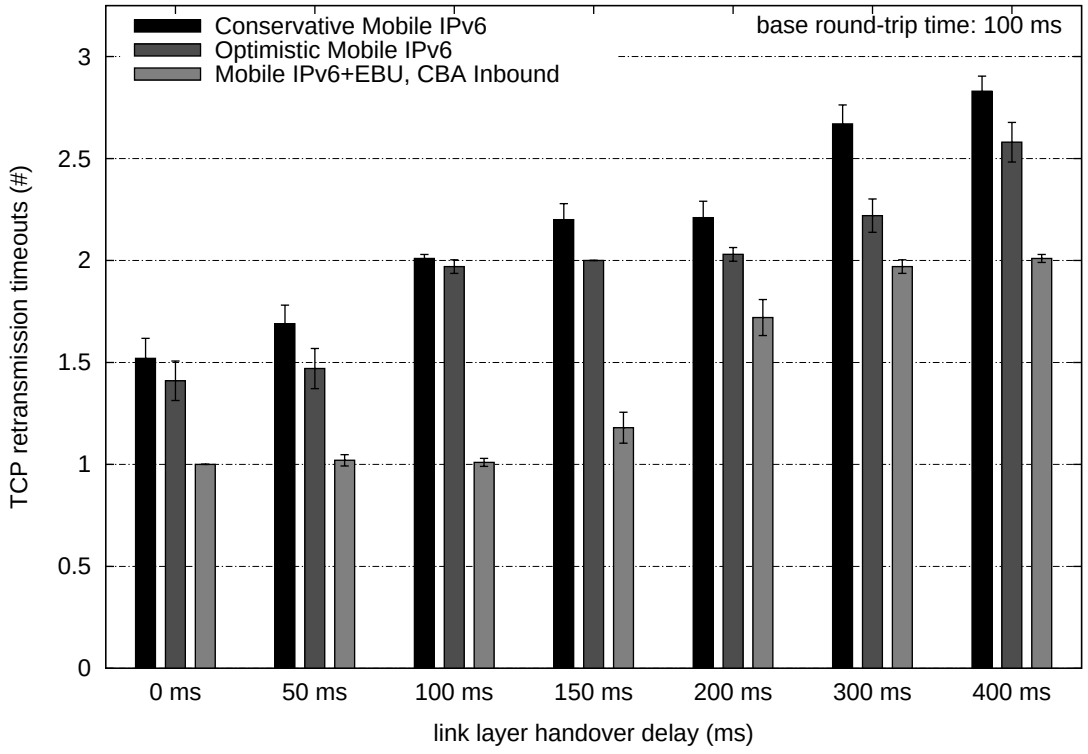



Figure A.5: Mean TCP retransmission timeout count by link layer handover delay in a symmetric network topology with 100-ms base round-trip times

before the retransmission timer expires for the first time. However, link layer handover delays of more than 150 ms generally cause the retransmission timer to expire before the early Binding Update reaches the correspondent node. The retransmission timer then expires two times consecutively in these cases. Two retransmission timeouts are always sufficient for Mobile IPv6 with Early Binding Updates even if the link layer handover delay is 400 ms. This is a considerable advantage given that the total TCP handover delay is then only three times of what a single retransmission timeout period would be, compared to seven times in the case of conservative or optimistic Mobile IPv6.

Delayed acknowledgments are responsible for the small decrease in the TCP handover delay which figure A.4 shows for Mobile IPv6 with Early Binding Updates at a link layer handover delay of 100 ms. The number of retransmission timeouts for Mobile IPv6 with Early Binding Updates is never higher than one if the link layer handover delay is 100 ms or lower, and delayed acknowledgments cannot reduce the number of retransmission timeouts. The belated arrival of delayed acknowledgments at the correspondent node then only *increases* the TCP handover delay, as explained in section 6.4.3. This effect discontinues once the link layer handover delay reaches 100 ms because delayed acknowledgments are then dropped, producing the small decrease in TCP handover delays shown in the figure.

### Increased Round-Trip Times

In an effort to determine the impact that the round-trip time has on TCP handover performance in the presence of link layer handover delays, the experiments discussed

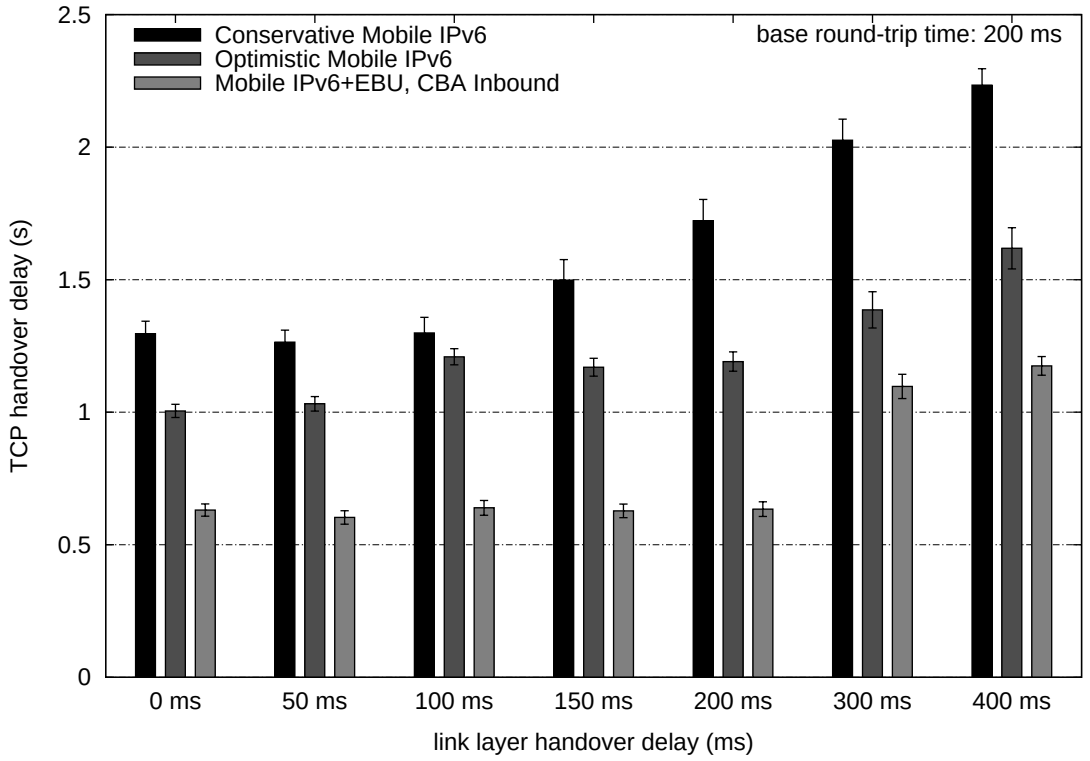



Figure A.6: Mean TCP handover delay as a function of link layer handover delay in a symmetric network topology with 200-ms base round-trip times

above were repeated with a higher base round-trip time of 200 ms. The TCP handover delays and retransmission timeout counts measured therein are summarized in figures A.6 and A.7. The figures again show averages and 95% confidence intervals for different link layer handover delays and the three Mobile IPv6 variants under consideration.

The impact of the higher base round-trip time can best be evaluated based on the measured number of consecutive retransmission timeouts per handover. For conservative Mobile IPv6, this number increases with the base round-trip time across all observed link layer handover delays. The reason for this is that forward buffering delays on the transmission path consume a smaller portion of the actual round-trip times when the base round-trip time is high. This gives Mobile IPv6 signaling, which takes the new transmission path without much forward buffering, a smaller advantage over the retransmission timeout period, which does incorporate forward buffering. The increase in the mean number of retransmission timeouts is strongest for small link layer handover delays, simply because the contribution of the round-trip time to the TCP handover delay is then highest according to figure A.3(a).

Optimistic Mobile IPv6 profits slightly from the higher base round-trip time if the link layer handover delay is 400 ms. The longer retransmission timeout period then facilitates a smaller number of consecutive retransmission timeouts per handover on average. In all other cases, the longer base round-trip time does not change the mean number of retransmission timeouts that TCP goes through during a handover.

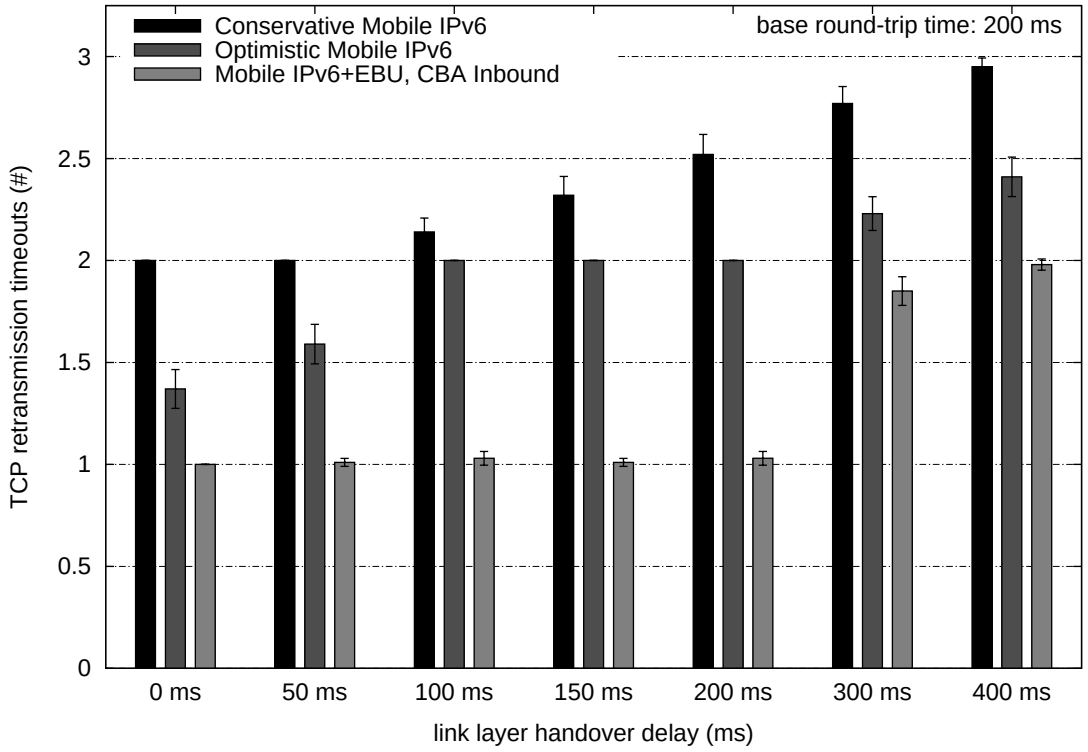



Figure A.7: Mean TCP retransmission timeout counts by link layer handover delay in a symmetric network topology with 200-ms base round-trip times

The mean TCP handover delays of Mobile IPv6 with Early Binding Updates are more stable to the increase in the base round-trip time because they solely depend on the link layer handover according to figure A.3(b). The longer base round-trip time may even reduce the mean number of retransmission timeouts per handover. This happens in the experiments with link layer handover delays of 150 ms or 200 ms, which now actually fit within the increased retransmission timeout period.

The small drop in TCP handover delays for a link layer handover delay of 100 ms, which was observed in figure A.4 for Mobile IPv6 with Early Binding Updates, does not appear in figure A.6 due to the compensating effect of a small number of experiments where TCP goes through two consecutive retransmission timeouts. The much longer TCP handover delay in those few experiments then compensates the effect that the sudden lack of delayed acknowledgments would otherwise have.

# B. Impact of Movement Detection

Mobility management involves protocols in different parts of the network stack; the mobility protocol itself is only one of these. A brief overview of handover-related activities at the IP layer was given in section 2.4, and this has shown that the standard IPv6 protocol suite is inappropriate to provide the efficiency needed for interactive, real-time applications. IP layer handover delays other than those produced by the mobility protocol itself are mainly due to router discovery, movement detection, and IP address auto-configuration.

In the experiments discussed so far, the delays of Stateless Address Autoconfiguration were eliminated by help of Optimistic Duplicate Address Detection, and optimum router discovery and movement detection was achieved with the DNA Protocol. A nice property of Optimistic Duplicate Address Detection is that it functions without support in the network, so a mobile node can use it as a replacement for standard Duplicate Address Detection independent of its environment. The DNA Protocol, however, requires the cooperation of access routes and this may not always be available. The purpose of this appendix is hence to illuminate how much the handover performance may degrade in situations where the application of the DNA Protocol is impossible due to nonexistent access router support. Two alternative movement detection mechanisms are considered for this purpose:

1. *Standard movement detection* — Access routers run the standard Neighbor Discovery protocol with the recommended extensions for mobile environments (see section 2.4.6). They multicast unsolicited Router Advertisement messages in intervals of between 30 ms and 70 ms, each including an Advertisement Interval option with an upper bound on the interval between consecutive advertisements. In this case, the advertised upper bound is 90 ms, which includes an extra 20 ms in order to account for scheduling granularities in mobile nodes and access routers. This increment is required whenever the actual maximum advertisement interval is smaller than 200 ms. Accordingly, a mobile node expects to receive a Router Advertisement message every 90 ms from a particular access router. If advertisements from the access router fail to be received for three times this period, the mobile node can conclude with high confidentiality, despite a potential for packet loss, that it has changed IP connectivity.

2. *Complete Prefix List* — Access routers behave as described above for standard movement detection. However, sophisticated logic on the mobile node now facilitates expedited movement detection based on the reception of a single Router Advertisement message from a new access router.

## B.1 Internet Telephony

Figure B.1 juxtaposes UDP handover delay averages and 95% confidence intervals of an Internet telephony application that were measured for each of the three movement detection protocols. The experiments were repeated for conservative Mobile IPv6, optimistic Mobile IPv6, and Mobile IPv6 with Early Binding Updates. All of them were conducted in a symmetric network topology with a fixed base round-trip time of 100 ms. The mobile node's link layer handover delay is zero. The results clearly reflect the different performance of the three movement detection mechanisms. For the DNA Protocol, the measured handover delays depend only on the round-trip time between the mobile node and the correspondent node. Router discovery and movement detection happen without delay based on a single Router Advertisement message which the mobile node can obtain from an access router instantaneously. Complete Prefix List usually does with a single Router Advertisement message as well, but the mobile node must wait for an unsolicited multicast advertisement. The wait time until the mobile node receives the first multicast advertisement after a handover ranges between 0 and 70 ms: In the best case, the new access router sends a Router Advertisement message right after the mobile node has arrived on the new link. In the worst case, the new access router has sent an advertisement just before the mobile node arrives, and the time until it sends the next advertisement is maximum. The mean interval between successive advertisement transmissions is 50 ms, so the mobile node can expect to receive the first message subsequent to handover after 25 ms. This wait time is responsible for the slightly longer handover delays for Complete Prefix List compared to the DNA Protocol.

Handover delays are longest with standard movement detection. The mobile node here waits three times the interval advertised in Advertisement Interval options in Router Advertisement messages until it considers a previous access router unreachable due to a handover. Changes in IP connectivity are therefore detected 270 ms after the last advertisement was received from the old access router. Since this last advertisement may have been sent up to 70 ms prior to the actual link layer handover, the latency of movement detection ranges between 200 ms and 270 ms. On average, the period between reception of the last advertisement from the old access router and the link layer handover is 25 ms, yielding a mean movement detection delay of 245 ms.

## B.2 TCP File Transfers

Figure B.2 shows averages and 95% confidence intervals of the handover delays measured in TCP SACK file transfer experiments per movement detection mechanism and Mobile IPv6 variant. The experiments were conducted in a symmetric network topology with a fixed base round-trip time of 100 ms; the link layer handover delay was again zero. The results indicate that Complete Prefix List and the DNA Protocol are very similar with respect to their impact on the TCP handover delay.

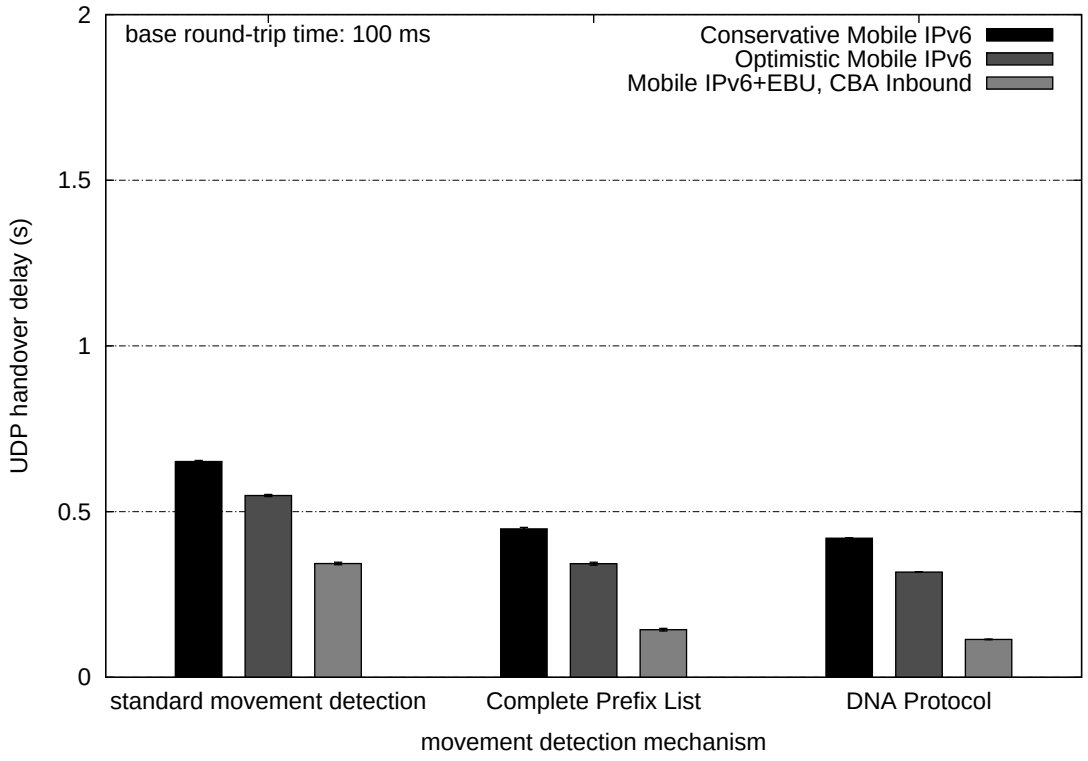



Figure B.1: Mean UDP handover delay as a function of link layer handover delay in a symmetric network topology with 100-ms base round-trip times

Both mechanisms detect movement based on a single Router Advertisement message. They differ only in the amount of time it takes the mobile node to obtain the first advertisement subsequent to handover. The ability of a mobile node to solicit an immediate Router Advertisement message gives the DNA Protocol a slight advantage over Complete Prefix List, where the expected time to receive the first advertisement after a handover is 25 ms. This additional delay in most cases does not increase the number of retransmission timeouts that TCP goes through during a handover, so it only occasionally has an effect on the TCP handover delay. On the other hand, the average movement detection delay of 245 ms in standard movement detection is a multiple of the retransmission timeout period and hence causes notably longer TCP handover delays.

Figure B.3 corroborates the above results with averages and 95% confidence intervals of the number of consecutive retransmission timeouts that the correspondent node goes through during a handover on the mobile-node side. In line with the similar TCP handover delays for Complete Prefix List and the DNA Protocol, the mean number of consecutive retransmission timeouts for Complete Prefix List is only slightly higher than it is for the DNA Protocol. More consecutive retransmission timeouts are on average required across all Mobile IPv6 variants if standard movement detection is used. This explains the relatively long TCP handover delays for standard movement detection compared to Complete Prefix List and the DNA Protocol.

The impact that standard movement detection has on the TCP handover performance is illustrated in figure B.4 for conservative Mobile IPv6 and Mobile IPv6

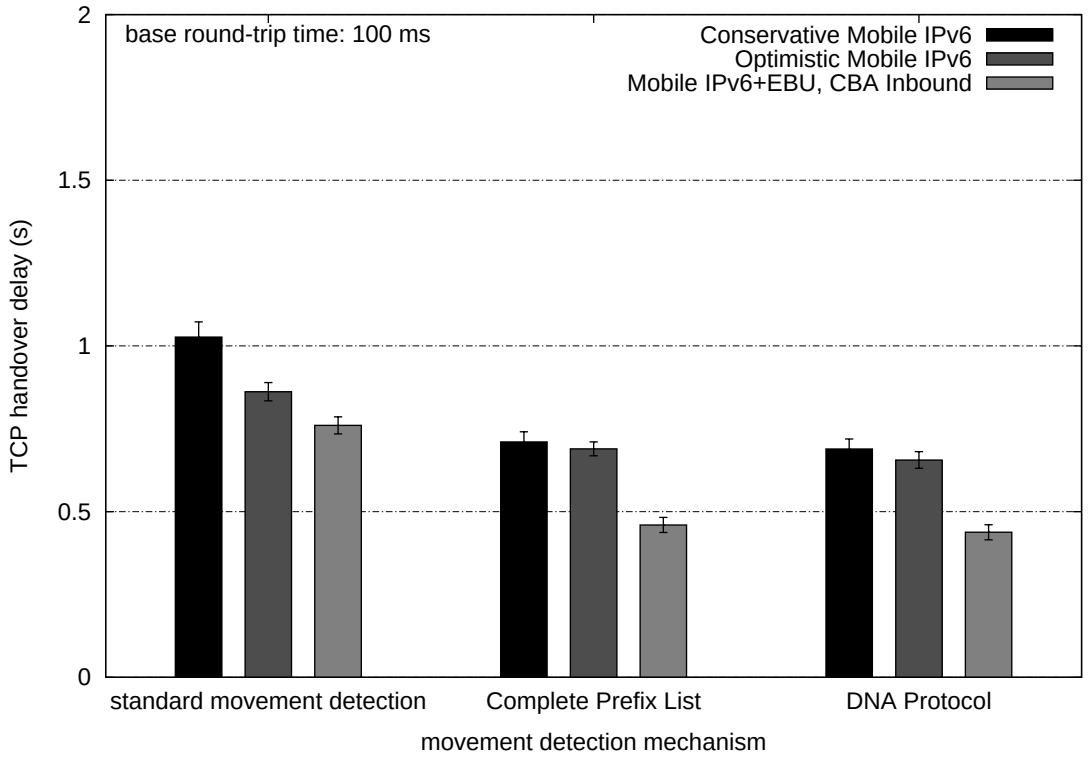



Figure B.2: Mean TCP handover delay as a function of link layer handover delay in a symmetric network topology with 100-ms base round-trip times

with Early Binding Updates. In both cases, the mobile node waits between 200 ms and 270 ms after the link layer handover before it decides that it has changed IP connectivity, even though it typically receives multiple Router Advertisement messages from the new access router in the meantime. Signaling for conservative Mobile IPv6 spans several round-trips as shown in part (a) of the figure, so it may take multiple retransmission timeout periods until the binding update completes on the correspondent node. Part (b) of the figure shows a binding update for Mobile IPv6 with Early Binding Updates. The time difference between the mobile node's transmission of the last TCP acknowledgment prior to handover and the transmission of the early Binding Update message after the handover is now only determined by the movement detection delay. The time gap between the arrival of the last TCP acknowledgment from the old care-of address and the arrival of the early Binding Update message is generally the same because forward buffering delays on the paths from the mobile node to the correspondent node are negligible. The binding update may thus already complete within a single retransmission timeout period despite the use of a slow movement detection mechanism.

The positive effect on handover performance that optimizations to the mobility protocol have independent of the movement detection mechanism is further evidenced by the measurements in figures B.2 and B.3. For standard movement detection, conservative Mobile IPv6 causes three consecutive retransmission timeouts on about one third of all handovers. The overall TCP handover delay then increases to seven times of what a single retransmission timeout period would be. Optimistic Mobile IPv6 completes a binding update before the second retransmission timeout with only

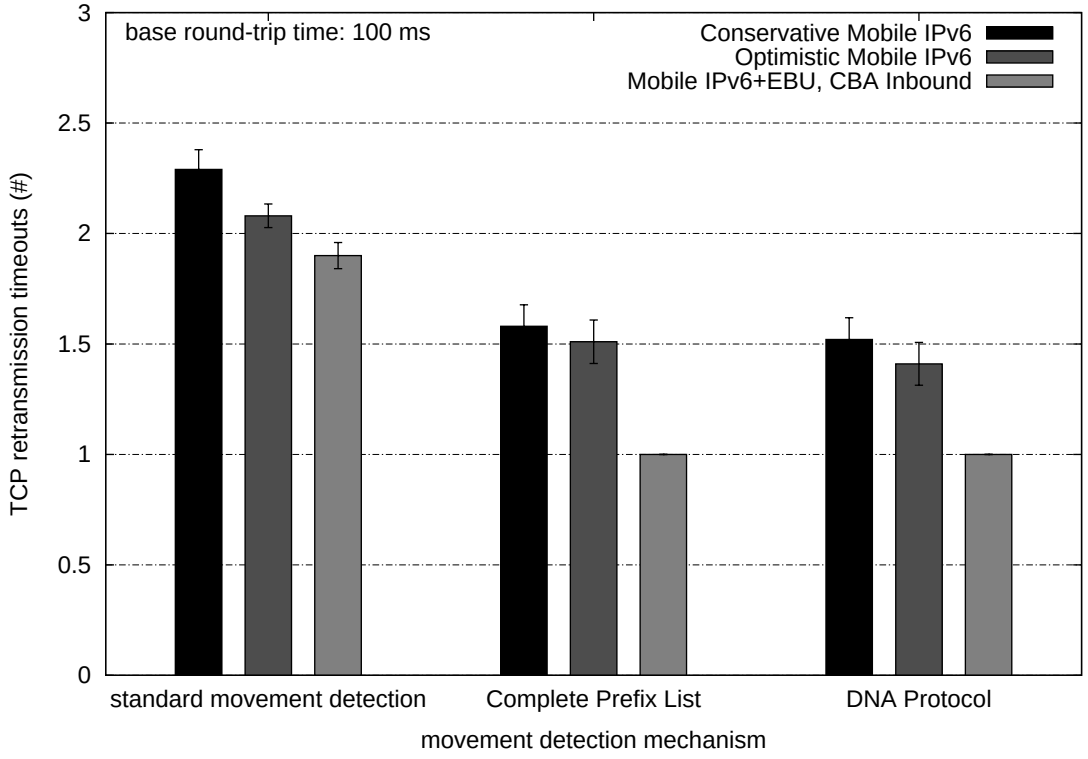
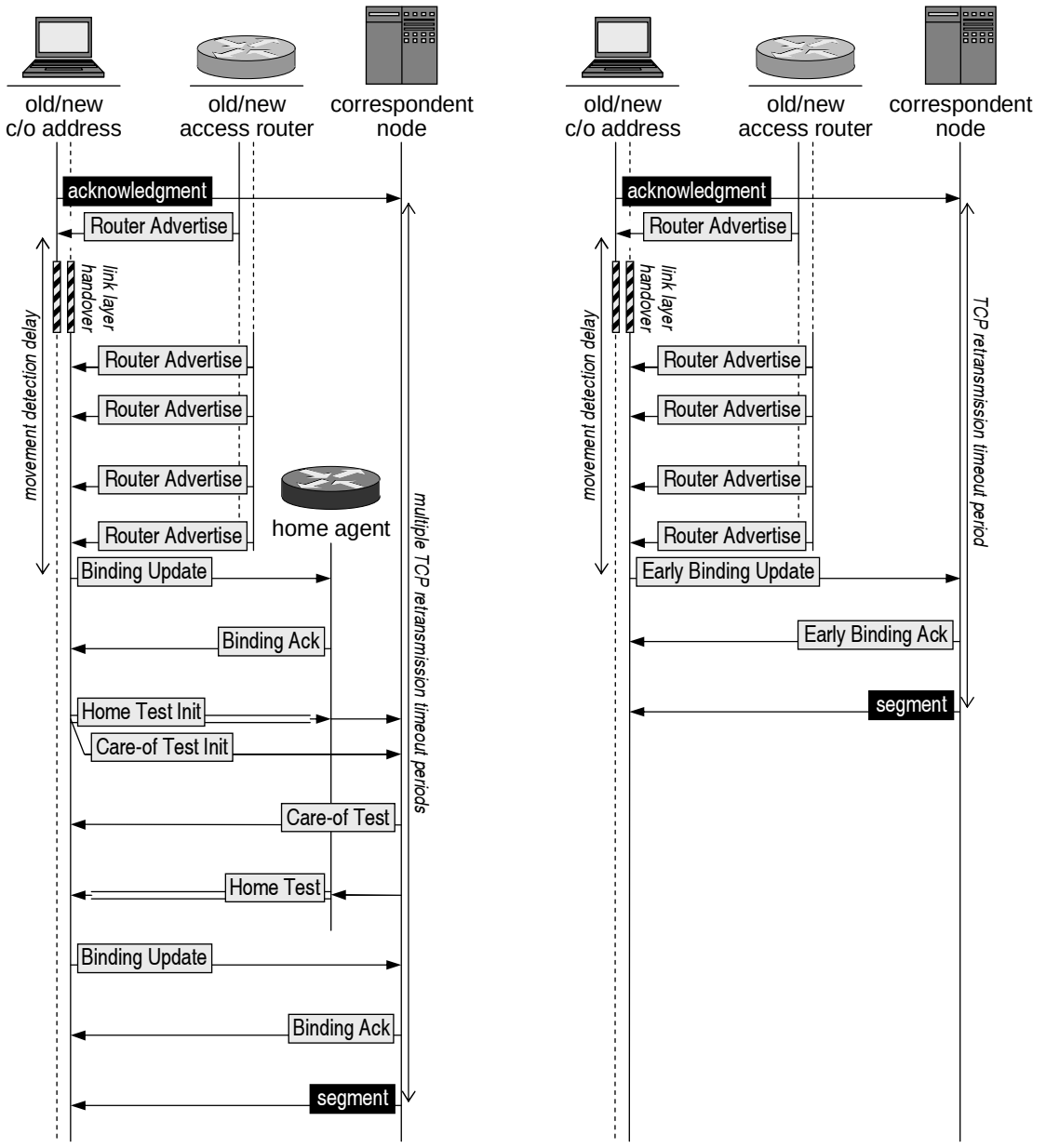



Figure B.3: Mean TCP retransmission timeout count by link layer handover delay in a symmetric network topology with 100-ms base round-trip times

very few exceptions, and Mobile IPv6 with Early Binding Updates may occasionally do with a single retransmission timeout. For Complete Prefix List or the DNA Protocol in conjunction with conservative or optimistic Mobile IPv6, chances are about equal that a binding update completes before the first or only before the second retransmission timeout. Two retransmission timeouts are slightly more probable for the combination of Complete Prefix List and conservative Mobile IPv6, and one retransmission timeout is a bit more likely for the combination of the DNA Protocol and optimistic Mobile IPv6. A binding update of Mobile IPv6 with Early Binding Updates always completes within a single retransmission timeout, both for Complete Prefix List and for the DNA Protocol.

### Increased Round-Trip Times

The impact that different movement detection mechanisms have on the TCP handover performance changes as the base round-trip time on the transmission path increases. There are mainly two reasons for this. First, the movement detection delay becomes relatively smaller compared to TCP's estimated retransmission timeout period, because the former is constant, while the latter scales with the actual round-trip time on the transmission path. Second, the impact of forward buffering delays on the old transmission path reduces as well given that the capacity of forward buffers is constant. Mobile IPv6 signaling, which takes the new transmission path and hence experiences only negligible forward buffering delays, may therefore span more retransmission timeout periods smaller when the base round-trip time is higher, as explained in section 6.4.2. In an effort to illuminate the consequences of




(a) Conservative Mobile IPv6 (b) Mobile IPv6 with Early Binding Updates

Figure B.4: Impact of standard movement detection on TCP handover delay

an increased base round-trip time, additional experiments were run for a symmetric network topology with a higher base round-trip time of 200 ms. Average TCP handover delays and retransmission timeout counts, along with their 95% confidence intervals, are shown in figures B.5 and B.6, respectively.

One impact of the higher round-trip time is that, for each movement detection mechanism, the TCP handover delays measured for the three Mobile IPv6 variants are now further apart. This is most apparent for standard movement detection: Mobile IPv6 with Early Binding Updates causes an average of 1.9 retransmission timeouts at the correspondent node when the base round-trip time between the mobile node and the correspondent node is 100 ms, but the retransmission timeout count goes down to 1.2 for a round-trip time of 200 ms. The reason for the reduced number of

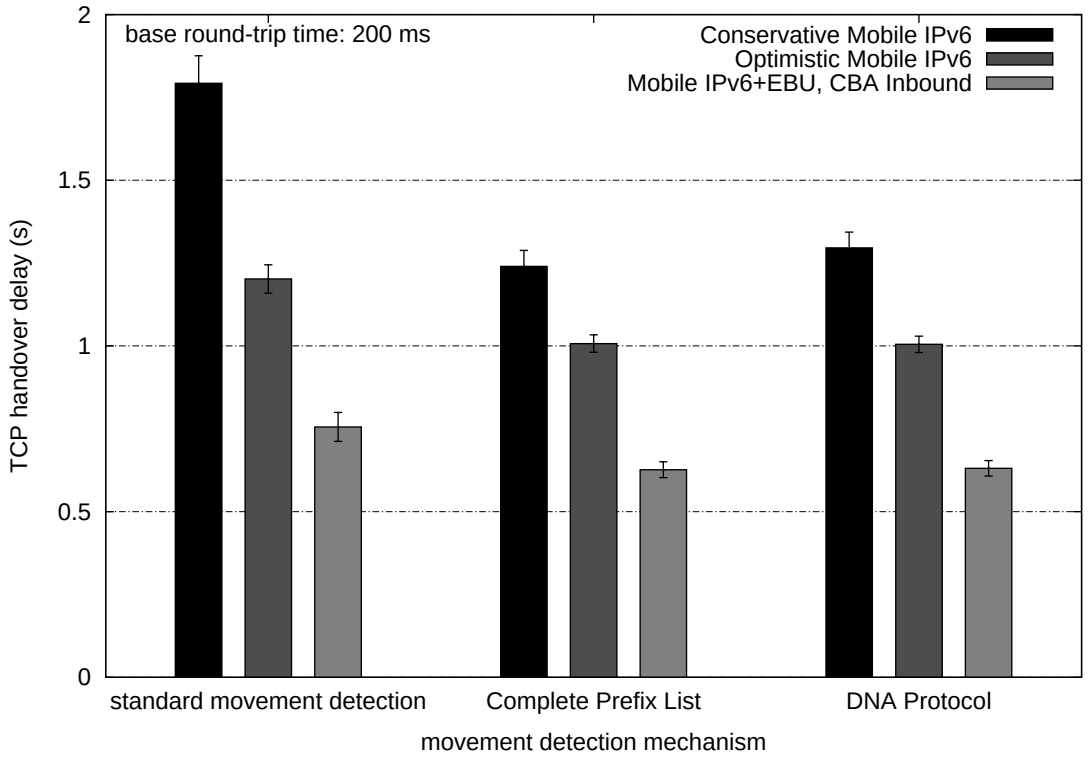



Figure B.5: Mean TCP handover delay as a function of link layer handover delay in a symmetric network topology with 200-ms base round-trip times

retransmission timeouts is that the longer round-trip times increase the retransmission timeout period to beyond the average movement detection delay of 245 ms. The binding update now happens within a single retransmission timeout period with high probability. The consequence of the reduced number of retransmission timeouts is that, although the retransmission timeout period increases with the round-trip time, TCP handover delays for Mobile IPv6 with Early Binding Updates turn out to be about the same for base round-trip times of both 100 ms and 200 ms.

The situation is different for the combination of standard movement detection with conservative Mobile IPv6 where, according to figure B.4, several round-trips of Mobile IPv6 signaling cause additional handover delays. The smaller effect of forward buffering delays in the presence of a higher base round-trip time now reduces the ratio between the retransmission timeout period and the base round-trip time. This leads to an increase in the mean number of retransmission timeouts as the base round-trip time grows from 100 ms to 200 ms and, consequently, to higher TCP handover delays.

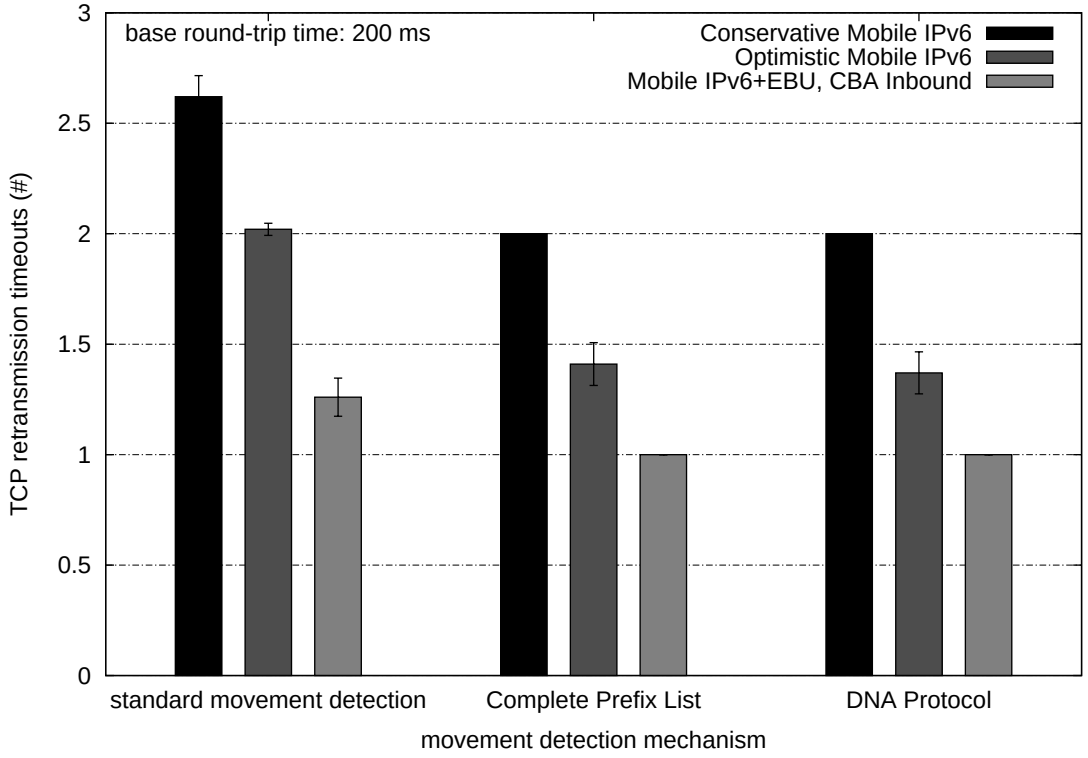



Figure B.6: Mean TCP retransmission timeout count by link layer handover delay in a symmetric network topology with 200-ms base round-trip times

# Bibliography


[1] 3GPP System to Wireless Local Area Network (WLAN) Interworking; System Description (Release 6). 3GPP Technical Specification Group Services and System Aspects, technical specification 23.234, version 6.6.0, September 2005.

[2] Donald E. Eastlake 3rd, Stephen D. Crocker, and Jeffrey I. Schiller. Randomness Recommendations for Security. IETF Request for Comments 1750, December 1994.

[3] Draft IEEE Standard for Local and Metropolitan Area Networks: Media Independent Handover Services. Draft IEEE Standard P802.21/D01.00, March 2006.

[4] Mark Allman. A Web Server's View of the Transport Layer. *ACM SIGCOMM Computer Communication Review*, 30(5):10–20, October 2000.

[5] Mark Allman, Hari Balakrishnan, and Sally Floyd. Enhancing TCP's Loss Recovery Using Limited Transmit. IETF Request for Comments 3042, January 2001.

[6] Ross Anderson. Why Information Security is Hard – An Economic Perspective. In *Proceedings of the Computer Security Applications Conference*, pages 358–365, December 2001.

[7] Stephane Antoine, Ming Wei, and A. H. Aghvami. Impact of Mobile IPv6 Handover on the Performance of TCP: an Experimental Testbed. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(1):31–33, January 2003.

[8] Antonios Argyriou and Vijay Madisetti. The Design and Evaluation of an End-to-End Handoff Management Protocol. *Kluwer Academic Publishers Wireless Networks*, 13(1):61–75, January 2007.

[9] Jari Arkko, James Kempf, Brian Zill, and Pekka Nikander. SEcure Neighbor Discovery (SEND). IETF Request for Comments 3971, March 2005.

[10] Jari Arkko and Iljitsch van Beijnum. Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming. IETF Internet Draft draft-ietf-shim6-failure-detection-07.txt (work in progress), December 2006.

[11] Jari Arkko, Christian Vogt, and Wassim Haddad. Enhanced route optimization for mobile ipv6. IETF Internet Draft draft-ietf-mipshop-cga-cba-03.txt (work in progress), February 2007.

[12] Tuomas Aura. Cryptographically Generated Addresses (CGA). *Lecture Notes in Computer Science*, 2851/2003:29–43, December 2003.

[13] Marcelo Bagnulo, Ignacio Soto, Alberto Garcia-Martinez, and Arturo Azcorra. Random Generation of Interface Identifiers. IETF Internet Draft draft-soto-mobileip-random-iids-00.txt (work in progress), January 2002.

[14] Paramvir Bahl, Atul Adya, Jitendra Padhye, and Alec Walman. Reconsidering Wireless Systems with Multiple Radios. *ACM SIGCOMM Computer Communication Review*, 34(5):39–46, October 2004.

[15] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. IETF Request for Comments 3704, March 2004.

[16] Nilanjan Banerjee, Arup Acharya, and Sajal K. Das. Seamless SIP-Based Mobility for Multimedia Applications. *IEEE Network*, 20(2):6–13, March 2006.

[17] Ralf Beck. Proaktives Mobilitätsmanagement auf Ende-zu-Ende-Basis. Diploma Thesis, Institute of Telematics, Universitaet Karlsruhe (TH), Germany, June 2006.

[18] Steven M. Bellovin. Defending Against Sequence Number Attacks. IETF Request for Comments 1948, May 1996.

[19] Massimo Bernaschi, Filippo Cacace, Giulio Iannello, Stefano Za, and Antonio Pescape. Seamless Internetworking of WLANs and Cellular Networks: Architecture and Performance Issues in a Mobile IPv6 Scenario. *IEEE Wireless Communications*, 12(3):73–80, June 2005.

[20] Ethan Blanton, Mark Allman, Kevin Fall, and Lili Wang. A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP. IETF Request for Comments 3517, April 2003.

[21] Robert Braden. Requirements for Internet Hosts – Communication Layers. IETF Request for Comments 1122, October 1989.

[22] Vladimir Brik, Arunesh Mishra, and Suman Banerjee. Eliminating Handoff Latencies in 802.11 WLANs Using Multiple Radios: Applications, Experience, and Evaluation. In *Proceedings of the ACM/USENIX Internet Measurement Conference*, October 2005.

[23] CERT Coordination Center. CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks. http://www.cert.org/advisories/CA-1996-21.html, September 1996.

[24] CERT Coordination Center. CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks. http://www.cert.org/advisories/CA-1998-01.html, January 1998.

[25] Ranveer Chandra, Paramvir Bahl, and Pradeep Bahl. MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card. In *Proceedings of the IEEE INFOCOM*, volume 2, pages 882–893, March 2004.

[26] John Chapin and Alok Shah. Software Radio Technology and Challenges. In *Proceedings of the International Telemetering Conference*, October 2002.

[27] JinHyeock Choi and Erik Nordmark. DNA with Unmodified Routers: Prefix List Based Approach. IETF Internet Draft draft-ietf-dna-cpl-01.txt (work in progress), April 2005.

[28] Sik Choi, Gyung-Ho Hwang, Taesoo Kwon, Ae-Ri Lim, and Dong-Ho Cho. Fast Handover Scheme for Real-Time Downlink Services in IEEE 802.16e BWA System. In *Proceedings of the Vehicular Technology Conference*, volume 3, pages 2028–2032, May 2005.

[29] Morten Jagd Christensen, Karen Kimball, and Frank Solensky. Considerations for IGMP and MLD Snooping Switches. IETF Internet Draft draft-ietf-magma-snoop-12.txt (work in progress), February 2005.

[30] David D. Clark. Window and Acknowledgement Strategy in TCP. IETF Request for Comments 813, July 1982.

[31] Richard Comerford. No Longer in Denial. *IEEE Spectrum*, 38(1):59–61, January 2001.

[32] Alex Conta, Stephen Deering, and Mukesh Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. IETF Request for Comments 4443, March 2006.

[33] Microsoft Corporation. New Networking Features in Windows Server 2008 and Windows Vista, April 2007.

[34] Les Cottrell. ICFA SCIC Network Monitoring Report. http://www.slac.stanford.edu/xorg/icfa/icfa-net-paper-jan05/, January 2005.

[35] Matt Crawford. Transmission of IPv6 Packets over Ethernet Networks. IETF Request for Comments 2464, December 1998.

[36] Greg Daley, Brett Pentland, and Richard Nelson. Movement Detection Optimizations in Mobile IPv6. In *Proceedings of the IEEE International Conference on Networks*, pages 687–692, September 2003.

[37] Cédric de Launois and Marcelo Bagnulo. The Paths Towards IPv6 Multihoming. *IEEE Communications Surveys & Tutorials*, 8(2):38–51, April 2006.

[38] Stephen E. Deering and Robert M. Hinden. Internet Protocol, Version 6 (IPv6) – Specification. IETF Request for Comments 2460, December 1998.

[39] Detecting Network Attachment (dna). IETF Working Group, http://www.ietf.org/html.charters/dna-charter.html, July 2007.

[40] Ralph Droms, Jim Bound, Bernie Volz, Ted Lemon, Charles E. Perkins, and Mike Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). IETF Request for Comments 3315, July 2003.

[41] Ashutosh Dutta, Tao Zhang, Yoshihiro Ohba Kenichi Taniuchi, and Henning Schulzrinne. MPA assisted Optimized Proactive Handoff Scheme. In *Proceedings of the ACM Conference on Mobile and Ubiquitous Systems*, pages 155–165, July 2005.

[42] Jean-Pierre Ebert, Eckhard Grass, Ralf Irmer, Rolf Kraemer, Gerhard Fettweis, Karl Strom, Guenther Traenkle, Walter Wirnitzer, Reimund Witmann, Hans-Jürgen Reumerman, Egon Schulz, Martin Weckerle, Peter Egner, and Ulrich Barth. Paving the Way for Gigabit Networking. *IEEE Communications Magazine*, 43(4):27–30, April 2005.

[43] Lars Eggert and Wesley M. Eddy. Towards More Expressive Transport-Layer Interfaces. In *Proceedings of the ACM/IEEE Workshop on Mobility in the Evolving Internet Architecture*, pages 71–74, December 2006.

[44] Mihaela Enachescu, Yashar Ganjali, Ashish Goel, Nick McKeown, and Tim Roughgarden. Part III: Routers With Very Small Buffers. *ACM SIGCOMM Computer Communication Review*, 35(3):83–90, July 2005.

[45] Pasi Eronen (Ed.). IKEv2 Mobility and Multihoming Protocol (MOBIKE). IETF Request for Comments 4555, June 2006.

[46] Minoruo Etoh and Takeshi Yoshimura. Wireless Video Applications in 3G and Beyond. *IEEE Wireless Communications*, 12(4):66–72, August 2005.

[47] Paul Ferguson and Daniel Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. IETF Request for Comments 2827, May 2000.

[48] Gerhard Fettweis, Tim Hentschel, and Ernesto Zimmermann. WIGWAM – A Wireless Gigabit System with Advanced Multimedia Support. In *Proceedings of the VDE Kongress*, October 2004.

[49] Sally Floyd, Thomas Henderson, and Andrei Gurtov. The NewReno Modification to TCP's Fast Recovery Algorithm. IETF Request for Comments 3782, April 2004.

[50] Eva Fogelstroem, Annika Jonsson, and Charles E. Perkins. Mobile IPv4 Regional Registration. IETF Internet Draft draft-ietf-mip4-reg-tunnel-04.txt (work in progress), October 2006.

[51] Yashar Ganjali and Nick McKeown. Update on Buffer Sizing in Internet Routers. *ACM SIGCOMM Computer Communication Review*, 36(5):67–70, October 2006.

[52] Bur Goode. Voice Over Internet Protocol (VoIP). *Proceedings of the IEEE*, 90(9):1495–1517, September 2002.

[53] Joshua B. Gross and Mary Beth Rosson. Looking for Trouble: Understanding End-User Security Management. In *Proceedings of the Conference On Human Factors And Computing Systems*, March 2007.

[54] Robert M. Hinden and Stephen E. Deering. Internet Protocol Version 6 (IPv6) Addressing Architecture. IETF Request for Comments 4291, February 2006.

[55] Robert Hsieh, Aruna Seneviratne, Hesham Soliman, and Karim El-Malki. Performance Analysis on Hierarchical Mobile IPv6 with Fast-Handoff over End-to-End TCP. In *Proceedings of the IEEE Global Telecommunications Conference*, November 2002.

[56] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. A Framework for Classifying Denial of Service Attacks. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 99–110, August 2003.

[57] IEEE. Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority, May 2007.

[58] Internet Assigned Numbers Authority. IP Version 6 Parameters. http://www.iana.org/assignments/ipv6-parameters.

[59] Kentaro Ishizu, Yoshia Saito, and Masahiro Kuroda. Additional Information Reports and IEs to Clarify Functionalities of CS and IS. IEEE 802.21 Draft 21-06-0657-00-0000, May 2006.

[60] Suraj Jaiswal and Sukumar Nandi. Simulation-Based Performance Comparison of TCP-Variants over Mobile IPv6-Based Mobility Management Schemes. In *Proceedings of the IEEE Conference on Local Computer Networks*, pages 284–291, November 2004.

[61] Jim H. James, Bing Chen, and Laurie Garrison. Implementing VoIP: A Voice Transmission Performance Progress Report. *IEEE Communications Magazine*, 42(7):36–41, July 2004.

[62] Hee-Jin Jang, Alper Yegin, JinHyeock Choi, and Kuntal Chowdhury. DHCP Option for Home Information Discovery in MIPv6. IETF Internet Draft draft-ietf-mip6-hiopt-00.txt (work in progress), August 2006.

[63] David Johnson, Charles E. Perkins, and Jari Arkko. Mobility Support in IPv6. IETF Request for Comments 3775, June 2004.

[64] Daniel Jungbluth. Software Design and Analysis of an End-to-End Optimization for Mobile IPv6 Route Optimization. Student Thesis, Institute of Telematics, Universitaet Karlsruhe (TH), Germany, July 2005.

[65] Daniel Jungbluth. Anticipation and Optimization of Home-to-Roaming Handovers in Mobile IPv6. Diploma Thesis, Institute of Telematics, Universitaet Karlsruhe (TH), Germany, May 2006.

[66] Stephen Kent. IP Authentication Header. IETF Request for Comments 4302, December 2005.

[67] Stephen Kent and Karen Seo. Security Architecture for the Internet Protocol. IETF Request for Comments 4301, December 2005.

[68] Seok Joo Koh, Moon Jeong Chang, and Meejeong Lee. mSCTP for Soft Handover in Transport Layer. *IEEE Communications Letters*, 8(3):189–191, March 2004.

[69] Rajeev Koodli and Charles Perkins. Mobile IPv4 Fast Handovers. IETF Internet Draft draft-ietf-mip4-fmipv4-03.txt (work in progress), February 2007.

[70] Rajeev Koodli (Ed.). Fast Handovers for Mobile IPv6. IETF Request for Comments 4068, July 2005.

[71] Suresh Krishnan, Nicolas Montavont, Eric Njedjou, Siva Veerepalli, and Alper Yegin. Link-Layer Event Notifications for Detecting Network Attachments. IETF Internet Draft draft-ietf-dna-link-information-06.txt (work in progress), February 2007.

[72] Koojana Kuladinithi, Niko A. Fikouras, Carmelita Goerg, Koltsidas Georgios, and Fotini-Niovi Pavlidou. Filters for Mobile IPv6 Bindings (NOMADv6). IETF Internet Draft draft-nomadv6-mobileip-filters-03.txt (work in progress), October 2005.

[73] John Loughney (Ed.), Jari Arkko, Marc Blanchet, Samita Chakrabarti, Alain Durand, Gerard Gastaud, Jun-ichiro itojun Hagino, Atsushi Inoue, Masahiro Ishiyama, Rajiv Raghunarayan, Shoichi Sakane, Dave Thaler, and Juha Wiljakka. IPv6 Node Requirements. IETF Request for Comments 4294, April 2006.

[74] Xiapu Luo and Rocky K. C. Chang. On a New Class of Pulsing Denial-of-Service Attacks and the Defense. In *Proceedings of the Network and Distributed System Security Symposium*, February 2005.

[75] Xiapu Luo and Rocky K. C. Chang. Optimizing the Pulsing Denial-of-Service Attacks. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks*, pages 582–591, June 2005.

[76] Karim Malki and Hesham Soliman. Simultaneous Bindings for Mobile IPv6 Fast Handovers. IETF Internet Draft draft-elmalki-mobileip-bicasting-v6-06.txt (work in progress), July 2005.

[77] Karim Malki (Ed.). Low Latency Handoffs in Mobile IPv4. IETF Internet Draft draft-ietf-mobileip-lowlatency-handoffs-v4-11.txt (work in progress), October 2005.

[78] Alberto Medina, Mark Allman, and Sally Floyd. Measuring the Evolution of Transport Protocols in the Internet. *ACM SIGCOMM Computer Communication Review*, 35(2):37–52, April 2005.

[79] Jelena Mirkovic and Peter Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, April 2004.

[80] Arunesh Mishra, Minho Shin, and William Arbaugh. An empirical analysis of the IEEE 802.11 MAC layer handoff process. *ACM SIGCOMM Computer Communication Review*, 33(2):93–102, April 2003.

[81] Paul V. Mockapetris. Telephony's Next Act. *IEEE Spectrum*, 43(4):28–32, April 2006.

[82] Nicolas Montavont and Thomas Noël. Handover Management for Mobile Nodes in IPv6 Networks. *IEEE Communications Magazine*, 40(8):38–43, August 2002.

[83] Nicolas Montavont and Thomas Noël. Analysis and evaluation of mobile IPv6 handovers over wireless LAN. *Kluwer Academic Publishers Mobile Networks and Applications*, 8(6):643–653, December 2003.

[84] Gabriel Montenegro and Claude Castelluccia. Crypto-based Identifiers (CBIDs): Concepts and Applications. *ACM Transactions on Information and System Security*, 7(1):97–127, February 2004.

[85] Gabriel Montenegro, Ed. Reverse Tunneling for Mobile IP, Revised. IETF Request for Comments 3024, January 2001.

[86] David Moore, Colleen Shannon, Doug Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet Denial-of-Service Activity. *ACM Transactions on Computer Systems*, 24(2):115–139, May 2006.

[87] Nick 'Sharkey' Moore. Optimistic Duplicate Address Detection (DAD) for IPv6. IETF Request for Comments 4429, April 2006.

[88] Robert Moskowitz and Pekka Nikander. Host Identity Protocol (HIP) Architecture. IETF Request for Comments 4423, May 2006.

[89] Sathya Narayanan, James Kempf, Erik Nordmark, Brett Pentland, JinHyeock Choi, Greg Daley, Nicolas Montavont, and Nick 'Sharkey' Moore. Detecting Network Attachment in IPv6 Networks (DNAv6). IETF Internet Draft draft-ietf-dna-protocol-03.txt (work in progress), October 2006.

[90] Thomas Narten and Harald Tveit Alvestrand. Guidelines for Writing an IANA Considerations Section in RFCs. IETF Request for Comments 2434, October 1998.

[91] Thomas Narten and Richard Draves. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. IETF Request for Comments 3041, January 2001.

[92] Thomas Narten, Erik Nordmark, William A. Simpson, and Hesham Soliman. Neighbor Discovery for IP version 6 (IPv6). IETF Internet Draft draft-ietf-ipv6-2461bis-05.txt (work in progress), October 2005.

[93] Pekka Nikander. End-Host Mobility and Multihoming with the Host Identity Protocol. IETF Internet Draft draft-ietf-hip-mm-04.txt (work in progress), June 2006.

[94] Pekka Nikander, Jari Arkko, Tuomas Aura, Gabriel Montenegro, and Erik Nordmark. Mobile IP Version 6 Route Optimization Security Design Background. IETF Request for Comments 4225, December 2005.

[95] Ville Nuorvala, Henrik Petander, and Antti Tuominen. Mobile IPv6 for Linux (MIPL), July 2007.

[96] Sixto Ortiz Jr. Internet Telephony Jumps off the Wires. *IEEE Computer*, 37(12):16–19, December 2004.

[97] Greg O'Shea and Michael Roe. Child-Proof Authentication for MIPv6 (CAM). *ACM SIGCOMM Computer Communication Review*, 31(2):4–8, April 2001.

[98] Sangheon Pack, Kunwoo Park, Taekyoung Kwon, and Yanghee Choi. SAMP: Scalable Application-Layer Mobility Protocol. *IEEE Communications Magazine*, 44(6):86–92, June 2006.

[99] Basavaraj Patil, Phil Roberts, and Charles E. Perkins. IP Mobility Support for IPv4. IETF Request for Comments 3344, August 2002.

[100] Vern Paxson. Measurements and Analysis of End-to-End Internet Dynamics. Ph.D. Thesis UCB//CSD-97-945, Computer Science Division, University of California, Berkeley, Berkeley, CA, USA, April 1977.

[101] Vern Paxson. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. *ACM SIGCOMM Computer Communication Review*, 31(3), July 2001.

[102] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems. *ACM Computing Surveys*, 39(1), April 2007.

[103] Charles E. Perkins. Securing Mobile IPv6 Route Optimization Using a Static Shared Key. IETF Request for Comments 4449, June 2006.

[104] Henrik Petander and Eranga Perera. Improved Binding Management for Make Before Break Handoffs in Mobile IPv6. IETF Internet Draft draft-petander-mip6-mbb-00.txt (work in progress), October 2005.

[105] John Postel. User Datagram Protocol. IETF Request for Comments 768, August 1980.

[106] Jon Postel. Internet Protocol. IETF Request for Comments 791, September 1981.

[107] Jon Postel. Transmission Control Protocol. IETF Request for Comments 793, September 1981.

[108] Gaurav Raina, Don Towsley, and Damon Wischik. Part II: Control Theory for Buffer Sizing. *ACM SIGCOMM Computer Communication Review*, 35(3):79–82, July 2005.

[109] Kishore Ramachandran, Sampath Rangarajan, and John C. Lin. Make-Before-Break MAC Layer Handoff in 802.11 Wireless Networks. In *Proceedings of the IEEE International Conference on Communications*, volume 10, pages 4818–4823, June 2006.

[110] Ramachandran Ramjee, Kannan Varadhan, Luca Salgarelli, Sandra R. Thuel, Shie-Yuan Wang, and Thomas La Porta. HAWAII: A Domain-based Approach for Supporting Mobility in Wide-Area Wireless Networks. *IEEE/ACM Transactions on Networking*, 10(3):396–410, June 2002.

[111] Michael J. Riezenman. Extending Broadband's Reach. *IEEE Spectrum*, 40(3):20–21, March 2003.

[112] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. SIP: Session Initiation Protocol. IETF Request for Comments 3261, June 2002.

[113] Constantin Schimmel. Implementierung und Analyse eines Mechanismus zur sicheren und effizienten Bestimmung des empfangenen Datenvolumens eines Kommunikationspartners. Student Thesis, Institute of Telematics, Universitaet Karlsruhe (TH), Germany, September 2005.

[114] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. IETF Request for Comments 3550, July 2003.

[115] Henning Schulzrinne and Elin Wedlund. Application-Layer Mobility Using SIP. *ACM SIGCOMM Mobile Computing and Communications Review*, 4(3):47–57, July 2000.

[116] Jeffrey Semke, Jamshid Mahdavi, and Matthew Mathis. Automatic TCP Buffer Tuning. *ACM SIGCOMM Computer Communication Review*, 28(4):315–323, October 1998.

[117] Eunsoo Shim and Richard D. Gitlin. Fast Handoff Using Neighbor Information. IETF Internet Draft draft-shim-mobileip-neighbor-00.txt (work in progress), November 2000.

[118] Sangho Shin, Andrea G. Forte, Anshuman Singh Rawat, and Henning Schulzrinne. Reducing MAC Layer Handoff Latency in IEEE 802.11 Wireless LANs. In *Proceedings of the International Workshop on Mobility Management and Wireless Access Protocols*, volume VOLUME, pages 19–26, October 2004.

[119] Kame-Shisa. Mobile IPv6 for FreeBSD 5.4, November 2005.

[120] Alex C. Snoeren and Hari Balakrishnan. An End-to-End Approach to Host Mobility. In *Proceedings of the International Conference on Mobile Computing and Networking*, pages 155–166, August 2000.

[121] Hesham Soliman, Claude Castelluccia, Karim El Malki, and Ludovic Bellier. Hierarchical Mobile IPv6 Mobility Management (HMIPv6). IETF Request for Comments 4140, August 2005.

[122] Miniwatts Marketing GroupInternet World Stats. World Internet Users and Population Statistics. http://www.internetworldstats.com/stats.htm, July 2007.

[123] Lawrence Stewart and James Healy. Tuning and Testing the FreeBSD 6 TCP Stack. Technical Report 070717B, Centre for Advanced Internet Architectures, Faculty of Information and Communication Technologies, Swinburne University, Melbourne, Australia, July 2007.

[124] Randall Stewart, Qiaobing Xie, Michael Tuexen, Shin Maruyama, and Masahiro Kozuka. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. IETF Internet Draft draft-ietf-tsvwg-addip-sctp-20.txt (work in progress), April 2007.

[125] Randall R. Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns Juergen Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang, and Vern Paxson. Stream Control Transmission Protocol. IETF Request for Comments 2960, October 2000.

[126] Cisco Systems. Unicast Reverse Path Forwarding. Cisco IOS Documentation, May 1999.

[127] Paul Tan. Recommendations for Achieving Seamless IPv6 Handover in IEEE 802.11 Networks. IETF Internet Draft draft-paultan-seamless-ipv6-handoff-802-00.txt (work in progress), February 2003.

[128] Fumio Teraoka, Kazutaka Gogo, Koshiro Mitsuya, Rie Shibui, and Koki Mitani. Unified L2 Abstractions for L3-Driven Fast Handover. IETF Internet Draft draft-irtf-mobopts-l2-abstractions-01.txt (work in progress), September 2006.

[129] Susan Thomson and Thomas Narten. IPv6 Stateless Address Autoconfiguration. IETF Request for Comments 2462, December 1998.

[130] Susan Thomson, Thomas Narten, and Tatuya Jinmei. IPv6 Stateless Address Autoconfiguration. IETF Internet Draft draft-ietf-ipv6-rfc2462bis-08.txt (work in progress), May 2005.

[131] Dirk Trossen, Govind Krishnamurthi, Hemant Chaskar, Robert C. Chalmers, and Eunsoo Shim. A Dynamic Protocol for Candidate Access-Router Discovery. IETF Internet Draft draft-trossen-seamoby-dycard-01.txt (work in progress), March 2003.

[132] András G. Valkó. Cellular IP: A New Approach to Internet Host Mobility. *ACM SIGCOMM Computer Communication Review*, 29(1):50–65, January 1999.

[133] Jon-Olov Vatn. An Experimental Study of IEEE 802.11b Handover Performance and Its Effect on Voice Traffic. Technical Report TRITA-IMIT-TSLAB R 03:01, Telecommunication Systems Laboratory, Department of Microelectronics and Information Technology, KTH, Royal Institute of Technology, Stockholm, Sweden, July 2003.

[134] Rolland Vida, Luis Henrique Maciel Kosmalski Costa, Serge Fdida, Steve Deering, Bill Fenner, Isidor Kouvelas, and Brian Haberman. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. IETF Request for Comments 3810, June 2004.

[135] Christian Vogt. A Comprehensive Delay Analysis for Reactive and Proactive Handoffs with Mobile IPv6 Route Optimization. Technical Report TM-2006-1, Institute of Telematics, Universitaet Karlsruhe (TH), Germany, January 2006.

[136] Christian Vogt and Jari Arkko. A Taxonomy and Analysis of Enhancements to Mobile IPv6 Route Optimization. IETF Request for Comments 4651, February 2007.

[137] Christian Vogt, Ralf Beck, Daniel Jungbluth, Max Laier, and Constantin Schimmel. Early Binding Updates and Credit-Based Authorization for the Kame-Shisa Mobile IPv6 Software. http://www.tm.uka.de/∼chvogt/ebucba/, July 2007.

[138] Christian Vogt, Roland Bless, Mark Doll, and Gregory Daley. Analysis of IPv6 Relocation Delays. Technical Report TM-2005-4, Institute of Telematics, Universitaet Karlsruhe (TH), Germany, April 2005.

[139] Christian Vogt, Roland Bless, Mark Doll, and Gregory Daley. Analysis of IPv6 Relocation Delays. Presentation at the 63th Meeting of the Internet Engineering Task Force, DNA Working Group Session, August 2005.

[140] Christian Vogt and Mark Doll. Efficient End-to-End Mobility Support in IPv6. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, volume 1, pages 575–580, April 2006.

[141] Christian Vogt and Martina Zitterbart. Efficient and Scalable, End-to-End Mobility Support for Reactive and Proactive Handoffs in IPv6. *IEEE Communications Magazine*, 44(6):74–82, June 2006.

[142] Damon Wischik and Nick McKeown. Part I: Buffer Sizes for Core Routers. *ACM SIGCOMM Computer Communication Review*, 35(3):75–78, July 2005.

[143] Wei Xing, Holger Karl, Adam Wolisz, and Harald Müller. M-SCTP: Design and Prototypical Implementation of an End-to-End Mobility Concept. In *Proceedings of the Workshop on the Internet Challenge: Technology and Applications*, October 2002.

[144] Hubert Zimmermann. OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, 28(4):425–432, April 1980.

# Index