Universität Karlsruhe (TH)
Fakultät für Informatik
Institut für Telematik
76128 Karlsruhe

Netzwerkmanagement und Hochleistungskommunikation

Teil XXII

Seminar SS 2000

Herausgeber:
Roland Bless
Verena Kahmann
Daniel Müller
Klaus Wehrle

Universität Karlsruhe (TH)
Institut für Telematik
http://www.telematik.informatik.uni-karlsruhe.de/

Fakultät für Informatik
Interner Bericht 2000-18
ISSN 1432–7864

Zusammenfassung

Der vorliegende interne Bericht enthält die Beiträge zum Seminar „Netzwerkmanagement und Hochleistungskommunikation“, das im Sommersemester 2000 zum 22. Mal stattgefunden hat.

Die Themenauswahl kann grob in folgende drei Blöcke gegliedert werden:

1. Ein Block ist der Hochgeschwindigkeits-Technologie gewidmet. Im ersten Beitrag wird das Konzepts des *Multiprotocol Label Switchings (MPLS)* vorgestellt, welches gegenüber herkömmlichem Routing einen Geschwindigkeitsvorteil im Bereich einer Größenordnung mit sich bringt. Der zweite Beitrag beschreibt *effiziente Verfahren und Algorithmen zur Klassifikation von IP-Paketen*, welche bei ständig zunehmender Geschwindigkeit der Router zunehmend an Bedeutung gewinnen. Der dritte Beitrag behandelt *Varianten des Transmission Control Protocols TCP*, dessen Mechanismen für höhere Leistung oder mobile Einsatzszenarien erweitert werden müssen.

2. Ein zweiter Block beschäftigt sich mit verschiedenen Themen aus den Bereichen drahtlose Kommunikation, Netzwerkmanagement und Sicherheit. Hier wird zum einen die Protokoll-Architektur des *Wireless Application Protocols (WAP)* vorgestellt, welches die speziellen Anforderungen mobiler Teilnehmer mit kleinsten drahtlos angebundenen Endgeräten, beispielsweise Mobiltelefonen, in Bezug auf die Internet- bzw. Datenkommunikation berücksichtigt. Zum anderen wird im Beitrag zur *automatischen Netzwerk-Konfiguration* auf ein im Zeitalter der ständig wachsenden Netze zunehmend wichtiges Netzwerkmanagement-Thema eingegangen. Schließlich werden im dritten Thema *Zero-Knowledge-Protokolle*, elegante Verfahren zur Authentisierung, vorgestellt, welche etwa im Bereich des elektronischen Zahlungsverkehrs eingesetzt werden können.

3. Der dritte Block umfasst den Themenbereich der Gruppenkommunikation. Hier werden einerseits *neuere Ansätze zum Multicast-Routing* beschrieben und andererseits eine Auswahl der funktional darüber angeordneten, zahlreichen *Multicast-Transportprotokolle*.

Abstract

This Technical Report includes student papers produced within a seminar of ‘Network Management and High Performance Communications’. For the 22nd time this seminar has attracted a large number of diligent students, proving the broad interest in topics of network management and high performance communications.

The topics of this report may be divided into three blocks:

1. One block is devoted to high speed and high performance technology. At first, the concept of *Multiprotocol Label Switchings (MPLS)* is described. Subsequently, *Efficient Methods and Algorithms for Classification of IP Packets* and *Variants of TCP* are presented.

2. A second block deals with various topics such as wireless communications, network management and security. The first article shows advantages of the *Wireless Application Protocol (WAP)* to access Internet information in mobile environments. The second article describes *Automatic Network Configuration Mechanisms* which are of increasing importance. Third, *Zero Knowledge Protocols* for secure authentication are examined and presented.

3. The third block deals with group communication and shows *New Approaches for Multicast Routing* as well as an overview of some *Multicast Transport Protocols*.

Inhaltsverzeichnis

Vorwort

Das Seminar „Netzwerkmanagement und Hochleistungskommunikation“ erfreut sich wie auch in den letzten Jahren weiterhin großer Beliebtheit. Gerade heutzutage sind Stichworte wie „Switching“, „Quality of Service“, „Mobil-Kommunikation“ oder „Internet“ in aller Munde. Daher sind die Forschungsgebiete in diesen Bereichen auch von allgemeinem Interesse, so dass sie eine derartige Vielzahl von innovativen Arbeiten aufweisen können, deren Behandlung in anderen Lehrveranstaltungen so detailliert nicht möglich ist.

Jetzt liegt auch der nunmehr 22. Seminarband als interner Bericht vor. Durch die engagierte Mitarbeit der beteiligten Studenten konnte so zumindest ein Ausschnitt aus dem komplexen und umfassenden Themengebiet klar und übersichtlich präsentiert werden. Für den Fleiß und das Engagement der Seminaristen sei daher an dieser Stelle recht herzlich gedankt.

Die ausgesprochen gute Resonanz bei den Studenten bestätigt uns darin, auch im Wintersemester 2000/2001 ein derartiges Seminar – natürlich mit geändertem aktuellem Inhalt – durchzuführen, so dass bald ein weiterer interner Bericht mit neuen Forschungsergebnissen aus innovativen Seminarbeiträgen erscheinen wird. Doch vorerst sollen im vorliegenden Band die nachfolgend kurz beschriebenen Themengebiete vorgestellt werden.

MPLS

bearbeitet durch Florian Lechner; Betreuer: Roland Bless

Die derzeitige Internet-Technik weist einige Defizite auf, die bei Betrieb von Hochleistungsnetzen offenkundig werden. Zum einen ist die mit dem Asynchronen Transfermodus (ATM) populär gewordene Switching-Technik um eine Größenordnung schneller als die klassische IP-Routing-Technik, so dass deren Einsatz im Backbone-Bereich insbesondere bei dem stetig steigendem „Bandbreitenbedarf“ weiterhin lukrativ erscheint. Zum anderen lassen sich Mechanismen zur Verkehrssteuerung (Traffic Engineering) und qualitätsbasierte IP-Dienste nicht ohne weiteres mit der heutigen IP-Technik realisieren. Der Ansatz des *Multiprotocol Label Switchings (MPLS)* ermöglicht nun, die Vorzüge der Switching-Technik für beliebige Schicht-3-Protokolle, vor allem für das Internet-Protokoll, zu nutzen. Die dazu notwendigen Verfahren und Mechanismen werden in diesem Beitrag vorgestellt.

Effiziente Verfahren/Algorithmen zum Klassifizieren von IP-Paketen

bearbeitet durch Tilman Seifert; Betreuer: Klaus Wehrle

Router sind heute nicht nur zuständig für die schnelle Weiterleitung der Datenpakete. Als Firewall eingesetzt, müssen sie Sicherheit im Netz gewährleisten, indem sie für jedes Paket entscheiden, ob es weitergeleitet werden darf oder nicht. Werden Quality-of-service-Anwendungen eingesetzt, müssen sie zugesicherte Dienstgüten erbringen. Hierfür werden effiziente Verfahren benötigt, die mehrere Felder des IP-Kopfes betrachten. Daher spricht man von *multi field classification*. In jüngerer Zeit wurden verschiedene Ansätze zur Lösung der multi field classification vorgelegt, über die hier ein Überblick gegeben wird.

TCP-Varianten

bearbeitet durch Thomas Lange; Betreuer: Klaus Wehrle

TCP bereitet beim Einsatz auf verschiedenen Netzwerktypen und Übertragungsmedien unterschiedliche Probleme, die von den jeweiligen Besonderheiten des Netzwerks und des Übertragungsmediums herrühren. Außerdem bereitet die ansteigende Größe und Komplexität des

Internet Probleme bei der Flusskontrolle. Es werden verschiedene Erweiterungen von TCP vorgestellt, welche die Fluss- und Staukontrolle verbessern und neue Übertragungsmedien unterstützen sollen. Zur Verbesserung der Flusskontrolle wurden die Erweiterungen Slow-Start, Congestion Avoidance, Fast Retransmit und Fast Recovery bereits eingeführt. In der Mobilkommunikation müssen häufigere Übertragungsfehler, Hand-Offs und eine begrenzte Bandbreite unterstützt werden. Bei der Hochleistungskommunikation müssen die festen Konstanten und zu kleinen Felder des TCP-Protokolls an geänderte Bedingungen angepasst werden, damit eine angemessene Funktion der Flusskontrolle zu gewährleisten. Für transaktionsorientierte Dienste wird vorgeschlagen, den 3-Wege-Handshake zu verbessern.

Das Wireless Application Protocol

bearbeitet durch Tanjev Stuhr; Betreuer: Daniel Müller

Das *Wireless Application Protocol* ist die Entwicklung des WAP-Forums, eines Zusammenschlusses von Firmen und Organisationen, welche damit einen Rahmen für den Zugriff drahtloser und mobiler Systeme auf das Internet stecken wollen. WAP ist nicht nur ein Protokoll sondern eine ganze Protokollarchitektur, welche unterschiedliche Transportsysteme unterstützt und bereits Sicherheitsmechanismen, Transaktionsprotokolle und Anwendungsunterstützung beinhaltet. Neben einer Vorstellung der Architektur wird in diesem Beitrag detaillierter auf die Neuerungen der kürzlich verabschiedeten Version 1.2 des Standards eingegangen werden, etwa auf die *WAP Push Architecture*.

Automatische Netzwerk-Konfiguration

bearbeitet durch Anne Fröhling; Betreuer: Daniel Müller

In heutigen Netzwerken müssen an den einzelnen Komponenten im Allgemeinen zunächst diverse Konfigurationseinstellungen vorgenommen werden, bevor ein Betrieb – insbesondere ein reibungsloser – möglich ist. In Anbetracht der weiterhin zunehmenden Verbreitung von Netzwerken in unterschiedlichste Anwendungsbereiche erscheint es wünschenwert, den manuellen Konfigurationsaufwand zu verringern, und zwar idealerweise bis auf Null, sodass keine Spezialkenntnisse mehr erforderlich sind, um Netzkomponenten in Betrieb zu nehmen. Existierende Ansätze zur Verringerung des Konfigurationsaufwandes sind beispielweise das *Dynamic Host Configuration Protocol* (DHCP) und die Autokonfigurationsfähigkeiten der Version 6 des *Internet Protocol* (IP). Eine relativ neue Arbeitsgruppe der *Internet Engineering Task Force* beschäftigt sich mit völliger Konfigurationsfreiheit (*Zero Configuration*). Im Beitrag wird ein Überblick über erforderliche Konfigurationmaßnahmen gegeben, die genannten Ansätze werden beschrieben und zu den Anforderungen der Zero-Configuration-Arbeitsgruppe in Beziehung gesetzt.

Zero-Knowledge-Protokolle

bearbeitet durch Michael Böhl; Betreuer: Daniel Müller

Zero-Knowledge-Protokolle dienen dazu, anderen gegenüber zu beweisen, dass man ein bestimmtes Wissen besitzt, ohne dieses Wissen aber dabei preiszugeben. Insbesondere kann ein solches Verfahren dazu verwendet werden, sich über ein Netzwerk hinweg zu authentisieren, also einem Kommunikationspartner die eigene Identität nachzuweisen. Im Beitrag wird das Konzept der Zero-Knowledge-Protokolle zunächst anhand von Beispielen vorgestellt und anschließend in allgemeinerer Form beschrieben. Schließlich werden auch praktische Einsatzgebiete von Zero-Knowledge-Protokollen betrachtet.

Neue Vorschläge für Multicast-Routing

bearbeitet durch Lars Kühn; Betreuer: Verena Kahmann

In der Internet-Protokollfamilie wird Multicast-Routing heute über Klasse-D-IP-Adressen durchgeführt. Es existieren verschiedene Routing-Protokolle, von denen die bekanntesten kurz vorgestellt werden sollen. Multicast ist jedoch im Internet nicht sehr weit verbreitet, da die bestehenden Ansätze einige Probleme bereiten. Daher sollen zwei neue Vorschläge für Multicast-Routing, die Multicast auf Unicast abbilden, beschrieben und bewertet werden.

Multicast-Transportprotokolle

bearbeitet durch Tobias Kraft; Betreuer: Verena Kahmann

Multicast stellt auch an die Transportebene bestimmte Anforderungen an Protokolle. Zwar sollen ähnliche Mechanismen implementiert werden, jedoch können Protokolle wie TCP oder UDP nicht einfach übertragen werden. In dieser Arbeit soll evaluiert werden, welche Anforderungen auch durch zukünftige Anwendungen an Multicast-Transportprotokolle gestellt werden, und wie bestehende Ansätze diese erfüllen.

Multi-Protocol Label Switching

Florian Lechner

Kurzfassung

Das herkömmliche Routing im Internet ist für viele der neuen Einsatzgebiete nicht mehr ideal. Insbesondere auf den Gebieten von Qualitätsunterstützung und Traffic Engineering gibt es mehrere Ansätze, die IP um diese fehlende Funktionalität erweitern. Im Folgenden soll das Multi-Protocol Label Switching (MPLS) betrachtet werden. MPLS ist von der IETF standardisiert worden, um bereits bestehende Ansätze für eine schnellere Paketweiterleitung in IP-Netzwerken zu vereinheitlichen. Durch den Einsatz der Switchingtechnologie, lässt sich eine Geschwindigkeitssteigerung bei der Paketweiterleitung um eine Größenordnung erreichen. Des Weiteren lässt sich MPLS sehr gut in bereits bestehende Netzarchitekturen wie z.B. ATM integrieren. Dadurch können die Vorteile von MPLS ohne hohe Zusatzkosten benutzt werden. Dadurch, dass bei ATM die Virtual Circuits erst während der Paketweiterleitung aufgebaut werden, entsteht ein sehr grosser Verwaltungsaufwand für die Initialisierung und den Abbau dieser virtuellen Verbindungen. Dieser Mehraufwand kann bei MPLS vermieden werden, da die so genannten Label Switched Paths schon vor dem ersten Datentransfer angelegt werden können.

1 Einleitung

1.1 Motivation

Durch das explosive Wachstum des Internets in den vergangenen Jahren zeigt sich verstärkt, dass das dominierende IP-Protokoll nicht für alle Probleme eine ideale Lösung bietet. Zuverlässigkeit, Unterstützung von *Quality of Service* (QoS) und eine Unterstützung für eine geschickte Verkehrssteuerung *(Traffic Engineering)*, also z.B. die bessere Verteilung von Netzwerklast, werden mit der herkömmlichen IP-Struktur nur unzureichend unterstützt. Insbesondere im Bereich der Paketweiterleitung, lässt sich durch den Einsatz der Switching-Technik eine Geschwindigkeitssteigerung um den Faktor 10 erreichen. Die bisherigen Ansätze basierten jedoch meist auf Vorschlägen einzelner Unternehmen, deren kommerzielle Interessen natürlich im Vordergrund standen.

1.2 Eine Lösung: MPLS

Die Internet Engineering Task Force (IETF) hat nun ein Verfahren entwickelt, das zum Standard beim IP-Switching werden soll. Es handelt sich hierbei um das *Multi-Protocol Label Switching*, oder kurz: MPLS. Der Name „Multi-Protocol“ sagt eigentlich schon aus, dass das Verfahren vom Prinzip her nicht auf IP beschränkt ist. In der Realität sieht es jedoch so aus, dass MPLS bislang nur Unterstützung für IPv4 und das neuere IPv6 bietet.

Ein Ziel von MPLS ist es, eine Verschmelzung von Routing und dem deutlich schnelleren Switching in einen einzelnen Standard zu erreichen. Das bringt einige Vorteile mit sich. So sind hier insbesondere die Kostenersparnis und eine Leistungssteigerung bei der Kombination von

Routing und Switching zu nennen. Eine gute Netzwerkskalierbarkeit ist ebenfalls ein wichtiges Kriterium, denn bisherige Lösungen für qualitätsorientierte Dienste, wie zum Beispiel ATM, haben besonders in diesem Bereich große Defizite.

Letztendlich wird eine größere Flexibilität beim Einsatz neuer Netzwerkdienste geboten. Dies wird in Zukunft ein immer wichtigerer Aspekt werden, denn mit erweiterten Netzwerkdiensten bieten sich für die Internet Service Provider (ISP) natürlich neue Möglichkeiten, zahlungskräftigen Kunden die gewünschten Dienstleistungen gegen ein entsprechendes Entgelt anzubieten.

Bevor nun auf die einzelnen Merkmale von MPLS eingegangen wird, ist es wichtig, sich die gegenwärtige Routingsituation im Internet noch einmal klar zu machen.

2 Routing im heutigen Internet

Kommt in einem herkömmlichen IP-Netz ein IP-Paket bei einem Router an, so wird zunächst der Paketkopf analysiert. Anhand der dort eingetragenen Zieladresse wird in der Weiterleitungstabelle nach dem längsten passenden Präfix gesucht. Das bedeutet in der Regel, dass die komplette Weiterleitungstabelle durchsucht werden muss. Ist der längste passende Präfix gefunden steht der zugehörige Routerausgang fest. Anschließend wird der Paketkopf entsprechend aktualisiert und das Paket dann an den ausgewählten Ausgang weitergeleitet. Diese Vorgehensweise wiederholt sich jetzt bei jedem Router, bis das Paket schließlich beim Empfänger ankommt.

Die Verteilung von Topologieinformationen, die zum Aufbau der Weiterleitungstabellen benötigt werden, erfolgt im Internet über verschiedene Routingprotokolle. Im globalen IP-Netzwerk bilden dabei die so genannten *Autonomen Systeme* (AS) eine flache Topologie und die oberste Hierarchiestufe. Autonome Systeme zeichnen sich dadurch aus, dass sie ihre innere Struktur verbergen und diese somit ausserhalb des Systems nicht sichtbar ist. Intern benutzen die AS ein *Interior Gateway Protocol* (IGP).

Der Austausch von Routinginformaitonen zwischen verschiedenen AS wird über ein Exterior Gateway Protocol (EGP) geleistet. Die gebräuchlisten IGP sind das *Open Shortest Path First* (OSPF) und *Intermediate System-Intermediate System* (IS-IS). Als EGP wird meistens das *Border Gateway Protocol* (BGP-4) eingesetzt.

Die IGP werden benötigt, um Informationen über alle angeschlossenen Systeme im eigenen Netz bekannt zu machen. Jeder Router hat also zu jedem Zeitpunkt ein komplettes Bild aller angeschlossenen Systeme und Router in seinem Netzwerk. Auf Basis dieser Information kann jeder Router für sich nun den kürzesten Pfad zu allen möglichen Zielsystemen oder -netzen ermitteln. Dies wird zum Beispiel mit dem Kürzesten Pfad Algorithmus von Dijkstra erreicht. Anschließend baut jeder Router seine eigene Weiterleitungstabelle (Forwarding Table) auf. In dieser Tabelle wird eine Zuordnung von IP-Netwerkpräfixen zum nächsten Router (Next-Hop) gespeichert.

2.1 Probleme

Es wird deutlich, dass jeder Router eine autonome Entscheidung über die Weiterleitung des Pakets trifft. Das Hauptproblem, welches sich aus dieser Zuordnung mittels der Weiterleitungstabelle ergibt, ist, dass Verbindungen, die nicht auf dem kürzesten Pfad liegen, bei Routingentscheidungen überhaupt nicht betrachtet werden. So kann es passieren, dass alle Pakete über den gleichen Ausgang des Routers weitergeleitet werden, und es so zu Stausituationen kommt, obwohl andere Ausgänge, die ebenfalls zum Ziel führen, überhaupt nicht ausgelastet sind. Die Möglichkeiten zur Verkehrssteuerung sind dadurch eingeschränkt.

Aus diesem Grund geht man heute dazu über, die Router manuell umzukonfigurieren bzw. Routinggewichtungen zu ändern und Alternativstrecken zu integrieren. Der Administrationsaufwand für solche Arbeiten ist verständlicherweise beträchtlich und die Fehleranfälligkeit entsprechend hoch.

Ein Aspekt, der vollkommen vernachlässigt wird, ist die Unterstützung für Quality of Service. Zwar gibt es hierfür entsprechende Erweiterungen (z.B. Integrated Services und RSVP), doch haben diese Verfahren sehr große Schwierigkeiten bei der Skalierbarkeit und sind deshalb nur begrenzt einsetzbar.

3 Multi-Protocol Label Switching

Die Idee, die hinter MPLS steckt, ist vom Prinzip her aus ATM bekannt. Über ein Signalisierungsprotokoll wird zuerst ein Pfad vom Sender zum Empfänger aufgebaut. An die zu verschickenden Datenpakete wird eine lokal gültige Kennung, das so genannte *Label*, angehängt. Die Entscheidung, an welchen Ausgang das Paket geleitet wird, erfolgt dann nicht mehr aufgrund der Zieladresse des Pakets, sondern auf Basis des Labels.

Der Vorteil dieses Vorgehens ist offensichtlich. Die aufwendigere Betrachtung des IP-Headers entfällt und somit findet die Weiterleitung nicht mehr über die Netzwerkschicht statt. Vielmehr wird eine Weiterleitung der Pakete direkt auf Schicht 2 realisiert, was einen enormen Geschwindigkeitsvorteil einbringt. Man kann davon ausgehen, dass Switches etwa eine Größenordnung schneller sind als Router.

Auf diese Weise erhält man eine verbindungsorientierte Weiterleitung in einem IP-Netzwerk. MPLS unterstützt so also direkt eine der zentralen Anforderungen, nämlich die Verteilung der Pakete entlang eines ausgewählten Pfads. So bietet MPLS dieselben Möglichkeiten wie ATM mit seinem Konzept der Virtual Circuits (VC).

Ein weiterer Vorteil von MPLS ergibt sich aus der festen Länge des Labels. Im Idealfall kann das Label direkt als Index einer Weiterleitungstabelle dienen und der benötigte Eintrag in einem Schritt ermittelt werden. Beim herkömmlichen Routingverfahren ist in der Regel eine aufwendigere Suche nach dem längsten Präfix nötig.

3.1 Architektur

Die Aufgabe eines Routers übernimmt bei MPLS der *Label Switching Router* (LSR). Ein LSR ist in der Lage Pakete auf Schicht-3 weiterzuvermitteln (Forwarding/Routing) und Pakete auf Schicht-2 weiterzuleiten (Switching). Er arbeitet dabei mit herkömmlichen IP-Routingprotokollen, aber leistet zusätzlich eine Zuordnung von Labels zu Zieladresse und die Verteilung der Labels mittels eines speziellen Protokolls.

Die Zieladressen gehören zu so genannten *Forwarding Equivalence Classes* (FECs). Unter einer FEC versteht man eine Gruppe von Schicht-3-Paketen (also in der Regel IP-Pakete), die in derselben Art und Weise weitergeleitet werden. Also beispielsweise entlang des gleichen Pfads und mit dem gleichen Weiterleitungsverhalten was die Behandlung von Qualitätsparametern betrifft. FECs sind normalerweise direkt mit einem oder mehreren (Ziel-) Netzwerkadressenpräfix(en) assoziiert. Wie später gezeigt wird, besteht jedoch auch die Möglichkeit, das gleiche Adresspräfix unterschiedlichen FECs zuzuordnen, um so eine gleichmäßige Lastverteilung zu erreichen.

Das Label ist ein kurzer Identifikator mit fester Länge, der an jedes zu vermittelnde Paket angehängt wird. Der Shim-Header (siehe Abbildung Nr. 2) beinhaltet neben dem eigentlichen

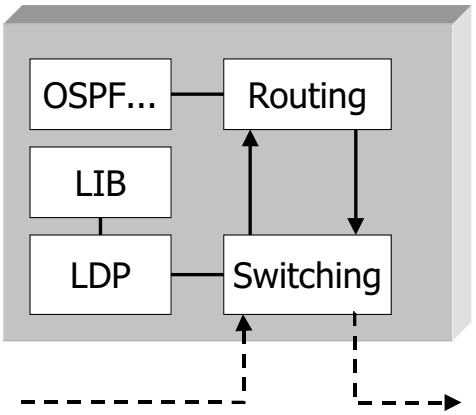

Abbildung 1: Schematischer Aufbau eines LSR. In der Label Information Base (LIB) sind sämtliche benötigte Zuordnungstabellen enthalten. Das Label Distribution Protocol LDP ist direkt an die Switchingkomponente der Schicht-2 gekoppelt. Auf Schicht-3 arbeiten die herkömmlichen Routingprotokolle wie z.B. OSPF.

Adresse	FEC Identifier
129.13.1.1:80	A
1.8.15.1	B
192.*	A

Tabelle 1: Beispiel einer FEC-Zuordnung

Label noch weitere Felder. Im COS-Feld besteht die Möglichkeit, einem Label mehrere Qualitätsmerkmale zuzuordnen. Das Stack Indicator Bit wird gesetzt, wenn das Label Teil eines Labelstacks ist, also mehrere MPLS-Shims kaskadiert sind. Das TTL-Feld schließlich bietet die Möglichkeit, Pakete, die in eine Routingschleife geraten sind, nach Ablauf der TTL zu entfernen.

Jeweils zwei angrenzende LSRs einigen sich auf einen Wert und die Bedeutung des Labels. Jedes Label muss also zwischen zwei LSRs eindeutig bestimmt sein. Nötig ist das, weil jeder LSR eine Zuordnung von Labeln zu den FECs herstellt, ohne die eine schnelle Paketweiterleitung nicht möglich wäre.

Jedes Label kennzeichnet also zwischen zwei LSRs eindeutig eine FEC. Wie bereits erwähnt, besteht aber auch die Möglichkeit, mehrere MPLS-Shims zu kaskadieren. Es ist also ein Labelstack mit jedem Paket verbunden. Auf dem Weg liegende LSRs können auf diesen Labelstack Labels plazieren oder sie entfernen. So ist es möglich, Pakete mehreren FECs zuzuordnen und die entsprechenden Pfade zu bestimmen, die das Paket durchlaufen soll.

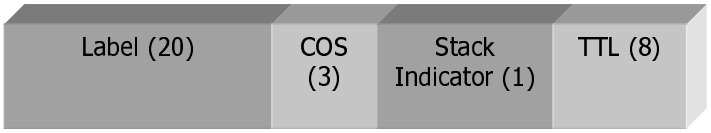

Abbildung 2: Aufbau des MPLS Shim-Headers (Feldlängen in bit)

Prinzipiell arbeitet MPLS nach folgendem Prinzip: Wenn LSR A ein Paket empfängt, das über LSR B hin zum Ziel LSR C verteilt werden muss, so wird zwischen den LSRs A und C ein *Label Switched Path* (LSP) aufgebaut. Entlang dieses Pfads werden dann die Pakete von einem

MPLS-Knoten zum nächsten weitergeleitet. Der Aufbau des Pfads geschieht über das *Label Distribution Protocol* (LDP). Die Weiterleitung des Pakets erfolgt anhand der zugeordneten FEC. LSR A packt also nun ein Label vor das Paket, das LSR B eindeutig erkennen kann und somit weiß, zu welchem FEC das entsprechende Paket gehört. Daraufhin wird das Paket an LSR B weitergeleitet.

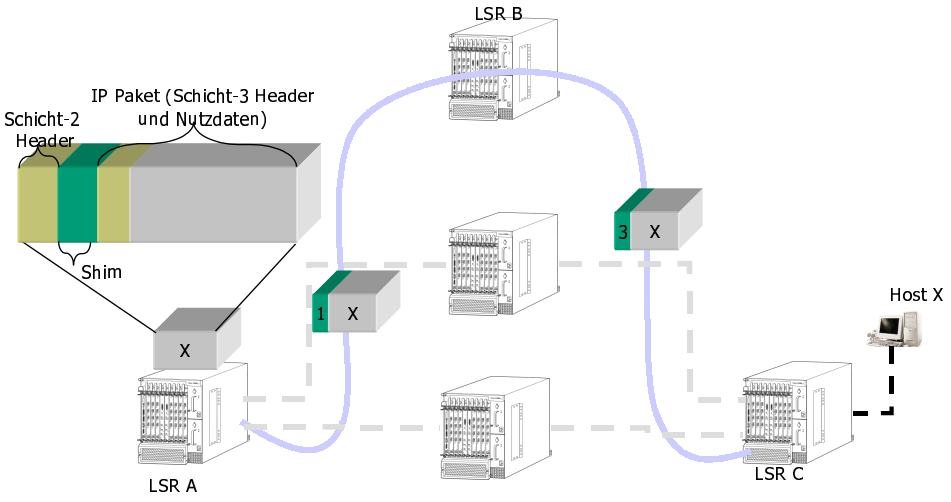

Abbildung 3: Pakettransport mit Labels von LSR A über LSR B und C zu Host X

Für diese eindeutige Zuordnung, sieht MPLS eine Reihe von Tabellen vor.

- Ein Next Hop Label Forwarding Entry (NHLFE) wird benötigt, um Pakete, die mit einem Label versehen sind, richtig weiterleiten zu können. Die wichtigsten Eintragungen, die ein NHLFE dafür beinhaltet, sind
  - Nächster Router für das Paket
  - Auszuführende Operation auf dem Labelstack
    - Austausch des obersten Labels auf dem Stack
    - Entfernen des obersten Labels auf dem Stack
    - Austausch des obersten Labels auf dem Stack. Anschließend werden weitere Labels auf den Stack gelegt.

NHLFE	Next Hop	Label Operation	Outbound Label	Outbound Port
123	LSR B	Push	1	2
456	LSR D	Pop		
500	LSR C	Replace	3	1

Tabelle 2: Beispiel einer Next Hop Label Forwarding Entry (NHLFE)-Tabelle

Es ist möglich, dass der empfangende Router gleichzeitig der Endpunkt eines LSP ist. In diesem Fall wird das Label einfach vom Stack entfernt („Pop“) und das Paket dann über Schicht-3 weiter verteilt.

- Die FEC-to-NHLFE Map (FTN) ist wichtig, falls Pakete empfangen werden, die noch kein Label besitzen. Der LSR ist also der Anfangspunkt eines LSP. In diesem Fall wird die FTN untersucht, in der eine entsprechende Zuordnung zum NHLFE steht.

FEC	NHLFE
A	123
B	456
C	500

Tabelle 3: Beispiel einer FEC-to-NHLFE-Tabelle

- Schließlich gibt es noch die Incoming Label Map (ILM). In der ILM befindet sich eine Zuordnung von Paketen, die bereits mit einem Label versehen sind, zu einem entsprechenden NHLFE.

Label	NHLFE
1	123
2	333
3	444

Tabelle 4: Beispiel einer Incoming Label Map (ILM)

LSR B kann nun aufgrund des Labels und der in seiner diversen Tabllen gespeicherten Informationen sofort ermitteln, dass das Paket weiter zu LSR C geleitet werden muss. LSR B tauscht also das obere Label vom Labelstack gegen ein neues Label aus (s. Abbildung 3). Dieses neue Label wiederum ist zwischen den LSRs B und C eindeutig definiert, so dass C nach Eingang des Pakets sofort den zugehörigen FEC ermitteln kann.

Wenn LSR C das Paket nun erhält und das Label inspiziert, stellt er fest, dass er der letzte LSR auf dem Weg zum Zielpunkt des Pakets ist. Er entfernt daraufhin das Label und fährt dann mit einer herkömmlichen Schicht-3-Verarbeitung des Pakets fort, bis das Paket schließlich beim Empfänger angelangt ist.

3.2 Das Label Distribution Protocol (LDP)

Eine der grundlegenden Anforderungen von MPLS ist, dass sich benachbarte LSRs auf die Bedeutung von auszutauschenden Labels verständigen. Genau für diese Aufgabe ist das Label Distribution Protocol LDP zuständig. Mittels dieses Kontrollprotokolls handeln alle LSRs ihre FEC/Label-Zuordnung miteinander aus, so dass es später möglich ist, den zugehörigen LSP anhand eines Labels zu ermitteln.

3.2.1 Einrichtung eines LSPs

MPLS bietet verschiedene Möglichkeiten, wie die Labelzuordnung, und damit schließlich auch die Einrichtung eines LSP, vorgenommen werden kann:

- *Topologiebasierte Zuordnung.* Bei diesem Verfahren erfolgt die Zuordnung, während der LSR die herkömmlichen Routingprotokolle wie OSPF oder BGP verarbeitet. Die Weiterleitungstabellen werden auf Basis dieser Informationen aktualisiert und die Labels entsprechend zugeordnet. Ein Vorteil dieses Verfahrens ist, dass ein Label zugeordnet ist, sobald eine Route existiert. Dadurch gibt es vor dem eigentlichen Label-Switching keine Verzögerung mehr, da keine Zuordnung mehr erfolgen muss.

- *Anfragegesteuerte Zuordnung.* Bei dieser Art der Zuordnung, werden Labels gemäß der Anforderungen von Protokollen wie RSVP zugewiesen. Wenn ein LSR die RSVP Nachrichten auswertet, können Änderungen an den Weiterleitungstabellen vorgenommen werden und die Labels entsprechend zugeordnet werden. Wie bei der topologiebasierten Zuordnung gilt, dass es beim eigentlichen Label-Switching nicht mehr zu Verzögerungen kommt. Ein Nachteil kann jedoch sein, dass man im Vergleich zur topologiebasierten Zuordnung eine größere Anzahl Labels benötigt.

- *Verkehrsgesteuerte Zuordnung.* Schließlich gibt es mit der verkehrsgesteuerten Zuordnung noch ein Verfahren, das erst bei Ankunft eines Datenpakets den LSR anstößt, eine Labelzuordnung und -verteilung vorzunehmen. Hier kommt es nun zu einer Verzögerung, wenn das Datenpaket ankommt, bis es weitervermittelt werden kann. Es kann daher besser sein, während dieser Initialisierungsphase Pakete auf der Netzwerkschicht weiterzuleiten. Die Verkehrsgesteuerte Zuordnung kann zusätzlich den Verbrauch von Labels reduzieren und kann, besonders bei einer großen Anzahl von Rechnern im Netz, die Nachteile des begrenzten Labelraums umgehen.

3.2.2 Explizite Labelverteilung

Die verschiedenen Möglichkeiten der Labelverteilung sollen anhand des folgenden Beispiels illustriert werden. Ein LSR A leitet Pakete an einen LSR B weiter. A ist in diesem Fall der sendende oder stromaufwärts gelegene (upstream) LSR. B, als empfangender LSR, heißt auch stromabwärts gelegener (downstream) LSR. Die Labelverteilung muss sicherstellen, dass die Bedeutung aller Labels, die zwischen den LSR A und B ausgetauscht werden richtig bekannt gemacht wird. Ein wichtiger Aspekt ist dabei, ob A oder B das Label zuordnen. MPLS sieht als Standardverfahren vor, dass der Downstream-LSR (hier also B) das Label zuordnet. Der Downstream-LSR leitet die Informationen dann an den Upstream-LSR weiter. Das hat den entscheidenden Vorteil, dass der Downstream-LSR das Label so wählen kann, dass er beim Eingang eines Pakets das Label direkt als Index für seine NHLFE-Tabelle benutzen kann. So können in nur einem Schritt sämtliche, zum Weiterleiten benötigten, Informationen ermittelt werden.

Unabhängig davon, welches Verfahren zur Verbreitung der Labelinformationen eingesetzt wird, ist die Hauptsache, dass alle beteiligten LSR sich auf konsistente Labeldefinitionen verständigen. Das Framework [CDFF+99] unterscheidet hierbei folgende Möglichkeiten:

Downstream Label Allocation — Die Downstream Label Allocation ist das bei MPLS gebräuchlichste Verfahren zur Labelverteilung. In diesem Fall ist der Downstream-LSR (hier LSR B) dafür zuständig, Labels zu generieren und diese entsprechend seinen Eingängen und den FECs zuzuordnen. So ordnet er jedem FEC F ein eigenes Label L zu. Diese Zuordnung teilt der stromabwärts gelegene LSR dann dem stromaufwärts gelegenen LSR (im Beispiel LSR A) mit. Dieser weiss damit, welches Label er den Paketen voranstellen muss. Der Downstream-LSR darf das gleiche Label mehrfach verwenden. Um jedoch den korrekten LSP zu ermitteln, muss bei Benutzung des gleichen Labels für mehrere FECs sichergestellt sein, dass die Zuordnung eindeutig ist.

Upstream Label Allocation — Wird die Upstream Label Allocation eingesetzt, so ist der sendende LSR für die Labelzuordnung zuständig. Er gibt die Informationen über seine Zuordnungen an die angeschlossenen Downstream LSR weiter. Dieses Verfahren ist vorgesehen, um eine Möglichkeit zur Optimierung bei Multicast zu haben. Derzeit ist dieses Verfahren aber nicht realisiert.

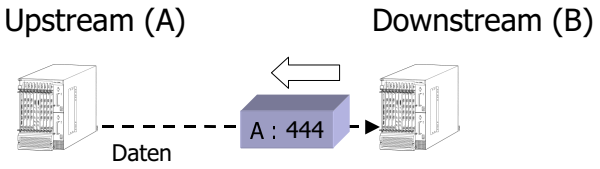

Abbildung 4: Downstream Label Allocation: LSR B sendet eine (Label:FEC)-Zuordnung an alle angeschlossenen Upstream-LSR (hier z.B. LSR A)

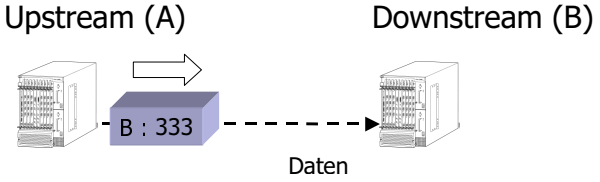

Abbildung 5: Upstream Label Allocation: LSR A sendet Labelinformationen an LSR B

Downstream-On-Demand — Zusätzlich zu den bereits vorgestellten Methoden, sieht MPLS den Downstream-On-Demand vor. In diesem Fall, kann jeder LSR vom nächsten an ihn stromabwärts angeschlossenen Router ein Label für eine spezielle FEC anfordern. Das ist wichtig, falls dem sendenden LSR zum Beispiel aufgrund von Routingänderungen keine gültigen Label/FEC-Zuordnungen mehr vorliegen.

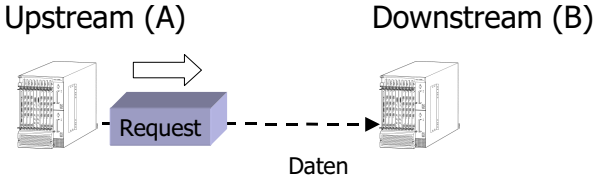

Abbildung 6: Downstream On Demand I: LSR A fordert von LSR B mittels eines Request-Pakets ein Label an.

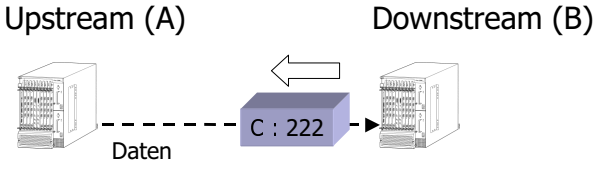

Abbildung 7: Downstream on Demand II: Nach Bearbeitung des Requests schickt LSR B die geforderte Label : FEC Zuordnung an LSR A.

Unsolicited Downstream — Schließlich sieht MPLS mit der „Unsolicited Downstream“ Variante noch vor, dass jeder LSR seine Labelinformationen an alle angeschlossenen Router in einer Art Rundruf bekannt geben kann. Wie die einzelnen LSRs auf diese „nicht angeforderten“ Labelinformationen reagieren, ist aber Implementierungssache. Eine Möglichkeit besteht darin, diese Informationen zu speichern und erst bei notwendigen Routingänderung darauf zurück zu greifen.

Neben der expliziten Labelverteilung sieht MPLS mit der impliziten Labelverteilung eine weitere Variante vor. Hierbei werden die benötigten Informationen per Piggybacking an die bekannten Protokolle wie OSPF, BGP, RSVP etc. angehängt.

3.2.3 Label-Granularität

MPLS sieht keine Einschränkungen bezüglich des Adressraums, der jedem Label zugeordnet ist, vor. So kann ein Label einer Route zu einem bestimmten Endsytem (also dem Weg zu einer einzigen IP-Adresse) zugeordnet sein, oder den Weg zu einem ganzen Adressraum (Netzbereich) abdecken. Die Zuordnung kann aber andererseits bis auf Anwendungsadressebene herunter gehen, also z.B. der HTTP-Port eines Zielrechners (IPsrc, IPsrcPort, IPdest, IPdestPort, protocol). Auch mehrere Datenströme können problemlos dem gleichen Label zugeordnet werden. Insbesondere für den Einsatz von Multicast ist diese Flexibilität sehr wichtig.

3.3 Paketweiterleitung

Wenn die Labelinformationen der einzelnen LSRs im Netz bekannt gemacht wurden, kann MPLS mit der schnellen Paketvermittlung beginnen. Im Folgenden soll daher die Weiterleitung eines Pakets entlang eines LSP genauer betrachtet werden.

3.3.1 Weiterleitung eines Pakets ohne Label

Trifft ein Paket ohne Label auf einen LSR, so ist dieser LSR dafür zuständig, dass dem Paket ein entsprechendes Label angehängt wird. Der LSR betrachtet dazu seine FTN. Das daraus ermittelte Label wird auf den (leeren) Labelstack gelegt. So ist das Paket für den Transport entlang eines LSPs vorbereitet und kann zum entsprechenden Ausgang des LSRs geleitet werden.

3.3.2 Weiterleitung eines Pakets mit Label

Trifft ein Paket mit Label an einem Router ein, so kann anhand seiner ILM sehr schnell der entsprechende NHLFE ermittelt werden, da das Label gegebenenfalls direkt als Index verwendet werden kann. Entsprechend den Eintragungen im NHLFE wird der zugehörige Ausgangsport ermittelt und die notwendige Operation auf dem Labelstack durchgeführt. Im Regelfall bedeutet dies die Entfernung des Labels und das anschließende Erzeugen eines neuen Labels, das wiederum auf den Stack gelegt wird. Dieses Label ist dann für den nächsten Router wieder dazu da, die entsprechenden Zuordnungen vorzunehmen.

3.3.3 Empfang des Labels am Ende eines LSPs

Wenn ein LSR ein Paket mit Label empfängt und anhand des ermittelten NHLFE feststellt, dass er der Router am Ende eines LSPs für dieses Paket ist, wird das Label vom Labelstack entfernt und der Labelstack ist dadurch wieder leer. Ein neues Label wird nicht auf den Stack gelegt.

Zu diesem Zeitpunkt ist aus dem Label-Paket wieder ein herkömmliches IP-Paket geworden, das nun in der altbewährten Methode an die Schicht 3 weiter gereicht wird und von dort seinen endgültigen Empfänger über herkömmliches IP-Routing erreicht.

3.4 Label Switched Path (LSP)

Beim Aufbau des LSP gibt es noch einige Varianten, die eine weitere Effizienzsteigerung von MPLS ermöglichen. Eine davon, ist das so genannte „Penultimate Hop Popping“. Der Hintergrund bei diesem Verfahren ist folgender:

Der letzte LSR entlang des LSPs empfängt normalerweise ein herkömmliches Paket inklusive Label. Dieses Label hat aber für ihn keine Bedeutung mehr, da er als letzter Router entlang des LSPs das Label sowieso nur noch entfernen muss. Dennoch untersucht der Router zuerst seine ILM und betrachtet anschließend den entsprechenden NHLFE. Hier stellt er nun fest, dass er selbst der Empfänger des Paketes ist und braucht dementsprechend nur noch den Labelstack zu leeren und das Paket an Schicht 3 zu übergeben.

Also werden für ein Paket, das sowieso an Schicht 3 übergeben wird, noch einmal sämtliche MPLS-Operationen durchgeführt. Mit dem Einsatz des Penultimate Hop Popping kann man sich diesen Vorgang sparen und das überflüssige Label schon vorher entfernen. Genauer gesagt entfernt der vorletzte LSR auf dem LSP bereits sämtliche Labels, so dass der letzte LSR schon ein herkömmliches IP-Paket empfängt und dies dann ohne weiteren Verzug an seine Schicht 3 übergeben kann.

Natürlich ist beim Einsatz von Penultimate Hop Popping die Haltung von zusätzlichen Informationen erforderlich. So muss der vorletzte LSR auf dem LSP selbstverständlich wissen, dass er der vorletzte LSR ist und das Label zur Effizienzsteigerung entfernen muss. Des Weiteren ist im MPLS-Standard nicht vorgesehen, dass Penultimate Hop Popping von jedem LSR beherrscht werden muss. Daher wird dieses Verfahren nur dann eingesetzt, wenn es explizit angefordert wurde und die entsprechenden LSRs dies unterstützen.

4 MPLS-Dienste

Wie zuvor beschrieben, ist MPLS in der Lage, eine schnelle und effiziente Paketvermittlung zu gewährleisten. Das allein ist aber sicherlich noch kein Grund, eine bestehende Infrastruktur auf MPLS umzurüsten. Der Hauptvorteil von MPLS liegt natürlich in der schnellen Schicht-2-Verteilung der Pakete. Auf diese Weise kann ein ISP seinen Kunden ohne großen Aufwand spezielle IP-spezifische Netzwerkschicht-Dienste anbieten, und so natürlich hochwertigere Produkte verkaufen.

4.1 MPLS und ATM

MPLS ist nicht auf eine spezielle Schicht-2 Technologie angewiesen. ATM bietet sich jedoch an, denn ATM ist besonders weit verbreitet und viele der eingesetzten Switches können ohne größeren Aufwand MPLS-fähig gemacht werden. MPLS verspricht dabei eine effizientere Ausnutzung der ATM-Infrastruktur im Vergleich zu Ansätzen wie IP-over-ATM. Dennoch ist MPLS nicht dazu gedacht, die gesamte ATM-Landschaft zu verändern. Viele ISPs, die keine Notwendigkeit für MPLS sehen, werden ihre ATM-Geräte nicht entsprechend umrüsten. Daher war ein Hauptbestreben der MPLS-Arbeitsgruppe, ein perfektes Zusammenspiel von MPLS-over-ATM und herkömmlichem ATM zu ermöglichen. Es war also wichtig, die bestehenden ATM-Protokolle zu unterstützen und einige Methoden zum kombinierten Einsatz von MPLS und ATM zur Verfügung zu stellen.

Eine Möglichkeit, ist das so genannte *Peer-Modell*. Hierbei wird ein ATM-Switch zu einem LSR erweitert. Man spricht dann von einem ATM-LSR. Dieser ATM-LSR unterstützt herkömmliches IP-Routing, Vermittlung von Paketen auf Schicht 3 und Weiterleitung von Zellen auf Schicht 2. Mehrere ATM-LSRs werden direkt miteinander verbunden und tauschen

Routingtabellen-Updates und LDP-Nachrichten aus. Der Vorteil dieser Vorgehensweise im Vergleich zum herkömmlichen IP-over-ATM ist, dass jeder ATM-LSR nur mit seinen direkten Nachbarn Routing-Beziehungen aufbaut und nicht mit jedem Router entlang des gesamten VC.

Eine andere Variante ist der *integrierte Ansatz*. Hierbei kommunizieren ATM-LSRs über herkömmliche ATM-Switches miteinander. Benachbarte ATM-LSRs können dann VCs zwischen sich aufbauen. Der größte Vorteil des integrierten Ansatzes ist, dass MPLS schrittweise in das bestehende ATM-Netzwerk integriert werden kann. Mit einem Satz von MPLS-LSRs in der Mitte eines großen ATM-Netzes, könnte man zu Anfang einen kostengünstigen Einstieg in die Technologie beginnen und übrige ATM-Switches nacheinander ersetzen. Die Grundfunktionalität von MPLS ist auch durch wenige MPLS-LSRs direkt gegeben. Der Nachteil dieser Variante besteht darin, dass weiterhin ATM-VCs aufgebaut werden müssen und dies im Vergleich zu den LSP bei MPLS deutlich langsamer ist.

Die dritte Möglichkeit ist der *Ships-In-The-Night* oder SIN-Ansatz. Hierbei wird das ATM-Netz in zwei separate Topologien aufgeteilt, wobei eine davon das neue MPLS unterstützt. Die Switches arbeiten dann, je nach Anforderung, nach ATM- oder MPLS-Prinzipien. Den wichtigen Teil der Entdeckung von Schleifen im Netz, leistet dabei das MPLS Protokoll.

4.2 Unterstützung von Dienstqualität

Die Bereitstellung von Diensten mit einer bestimmten Dienstqualität (*Quality of Service*) wird in Zukunft weiter an Bedeutung zunehmen. Denn durch die Bereitstellung von besseren Diensten können ISPs höherwertige Produkte verkaufen und so ihre Einnahmen steigern. MPLS unterstützt QoS durch mehrere Möglichkeiten. Es ist zum Beispiel möglich, Qualitätsindikatoren innerhalb des Labels zu integrieren. So können Pakete mit entsprechenden Labels verschiedenen Prioritäten zugeordnet werden. Des Weiteren besteht die Möglichkeit, einem kompletten LSP bzw. FEC eine bestimmte Dienstgüte zuzuordnen. So kann zum Beispiel ein LSP auch direkt auf Basis der zugrundeliegenden ATM-Technik betrieben werden und die in ATM verfügbarn QoS-Merkmale nutzen.

4.3 Multicast

MPLS unterstützt Multicasting-Verkehr wie z.B. Mehr-Teilnehmer-Videokonferenzen, indem Multicast-Pakete direkt einem ganzen LSP-Baum zugeordnet werden können. Die Pakete erreichen dann natürlich aufgrund der MPLS-Technologie schneller den Empfänger. Zusätzlich bietet MPLS aber die Möglichkeit, Multicast-Pakete auf einem LSP von anderen Paketen strikt zu trennen. Das ist wichtig, da Multicast auf Verstopfungen im Verkehrsfluss nicht so reagiert, wie herkömmliche TCP-Anwendungen.

5 Zusammenfassung

Das von der IETF entwickelte Multi-Protocol Label Switching (MPLS) bietet mit seiner schnellen Paketvermittlung eine gute Möglichkeit, dem immer stärker zunehmenden Internetverkehr begegnen zu können. Switches können Pakete etwa zehn mal schneller vermitteln, als herkömmliche Router. Der zusätzlich benötigte Aufwand (z.B. durch Label Verteilung) fällt bei MPLS, was die Paketweiterleitung angeht, nicht ins Gewicht. Mit der Verwendung des kurzen 20-Bit Labels trägt MPLS den Anforderungen, nach einer möglichst einfachen Verarbeitung der benötigten Zusatzinformationen Rechnung.

Da MPLS sich hervorragend in bestehende Architekturen wie ATM eingliedert und eine Umrüstung der Hardware in vielen Fällen sehr einfach möglich ist, kann ein ISP sein Netz kostengünstig erweitern und höherwertige MPLS-Dienste seinen Kunden anbieten, um so seinen Umsatz steigern zu können. ATM Merkmale wie Quality of Service können mit MPLS-over-ATM weiterhin genutzt werden. Hinzu kommen die neuen Merkmale wie zum Beispiel die Möglichkeiten der erweiterten Verkehrssteuerung und -planung (Traffic Engineering).

Bei der Diskussion um MPLS ist natürlich nicht zu vernachlässigen, dass im Gegensatz zu ähnlichen Ansätzen, wie beispielsweise dem Tag-Switching, hinter MPLS nicht die kommerziellen Interessen einer einzelnen Firma stehen, sondern MPLS von der IETF als offener Standard eingeführt wurde. Allein deswegen wird MPLS in Zukunft weiter an Bedeutung gewinnen.

Literatur

[CDFF+99] Ross Callon, Paul Doolan, Nancy Feldman, Andre Fredette, George Swallow und Arun Viswanathan. A Framework for Multiprotocol Label Switching. *IETF Internet Draft*, September 1999.

Effiziente Verfahren/Algorithmen zum Klassifizieren von IP-Paketen

Tilman Seifert

Kurzfassung

Router sind heute nicht nur zuständig für die schnelle Weiterleitung der Datenpakete. Als Firewall eingesetzt, müssen sie Sicherheit im Netz gewährleisten, indem sie für jedes Paket entscheiden, ob es weitergeleitet werden darf oder nicht. Werden Quality-of-service-Anwendungen eingesetzt, müssen sie zugesicherte Dienstgüten erbringen. Hierfür werden effiziente Verfahren benötigt, die mehrere Felder des IP-Kopfes betrachten. Daher spricht man von *multi field classification*. In jüngerer Zeit wurden verschiedene Ansätze zur Lösung der multi field classification vorgelegt, über die hier ein Überblick gegeben wird.

1 Einleitung

Die Verbreitung des Internets scheint keine Grenzen zu kennen; seit Einführung des World Wide Web steigt die Zahl der ans Internet angeschlossenen Teilnehmer, Rechner und Netzwerke in beeindruckendem Tempo. Damit einher geht natürlich auch eine Steigerung des zu bewältigenden Datenflusses. Neue Anwendungen erfordern die Einhaltung von bestimmten Dienstgüte-Kriterien; sie bauen z. B. auf Protokollen wie *Integrated Services / Resource Reservation Protokol (RSVP)* oder *Differentiated Services* auf. Diese Protokolle müssen von den Routern unterstützt werden.

Um diesen neuen Anforderungen gerecht zu werden, müssen drei Schlüsselfaktoren betrachtet werden: Die Übertragungsgeschwindigkeit in den großen Datenleitungen, der Datendurchsatz der Router und die Geschwindigkeit der Paketweiterleitung. Der Unterschied zwischen den beiden letztgenannten ist, dass der Datendurchsatz des Routers davon abhängt, wie er die Pakete intern speichert, wie er die Warteschlangen verwaltet etc., während die Weiterleitungsgeschwindigkeit von den *routing lookup*-Algorithmen abhängt. Für die zwei erstgenannten Probleme gibt es praxistaugliche, schnelle Lösungen, so existieren z. B. optische Leiter für den Gigabit-Bereich.

Diese Arbeit befasst sich mit dem dritten Problem, bei dem eine Routing-Datenbank verwendet werden soll, um für einzelne Pakete zu entscheiden, ob, über welchen Ausgang (also zu welchem Nachbarn) und wie schnell sie weitergeschickt werden sollen.

Zunächst sollen kurz die beiden Adressierungsschemata IPv4 und IPv6 vorgestellt werden. Daraus ergeben sich die Aufgaben, welche von den Routern zu erfüllen sind. Deren prinzipieller Aufbau soll, ebenfalls nur kurz, erläutert werden. Im Abschnitt 4 werden die gängigen Verfahren vorgestellt, die zur Zeit Einsatz finden. Anschließend sollen Verfahren gezeigt werden, die bessere Effizienz aufweisen und besser für den Schritt zu IPv6 gerüstet sind.

2 Adressierungsschemata im Internet

2.1 IPv4

An dieser Stelle sollen nur die in diesem Zusammenhang wichtigen Eigenschaften des IP-Headers erwähnt werden. Neben den jeweils 32 Bit langen Sender- und Empfängeradressen gibt es unter anderem noch zwei jeweils 8 Bit lange Felder, die den *Diensttyp* und das von der darüberliegenden Schicht verwendete *Transportprotokoll* (also z. B. TCP oder UDP) kennzeichnen. In dem Feld für den Diensttyp kann die gewünschte Dienstqualität (Priorität, Geschwindigkeit, Zuverlässigkeit) eingetragen werden. (Allerdings wird vom Protokoll an sich nichts in dieser Hinsicht garantiert.)

Die Protokolle TCP und UDP verwenden Portnummern für die Kennzeichnung einer Verbindung. Da diese Portnummern Aufschluss über den Dienst der Verbindung geben, sind sie ebenfalls sehr wichtig für die Klassifikation von IP-Paketen. Im Abschnitt 3 wird hierauf noch einmal eingegangen.

2.2 Künftiger Standard: IPv6

Mit der Einführung des neuen Protokolls IPv6 sollen diese Probleme angegangen werden; gleichzeitig werden neue Möglichkeiten für neue Anwendungen geschaffen.

Was ist neu an IPv6?

- Als wichtigster Unterschied zu IPv4 ist die Länge der Adresse zu nennen: War eine Adresse in IPv4 noch 32 Bit lang, so wird sie in IPv6 128 Bit lang sein (das sind also ca. $3 \cdot 10^{38}$ Adressen, oder ca. 10^{24} Adressen pro m^2 Erdoberfläche).

  Auch bei ineffizienter Ausnutzung des Adressbereiches wird das ausreichend sein, selbst wenn jedes Mobiltelefon, jeder Personal Assistant, jedes Auto und jeder Kühlschrank seine eigene IP-Adresse bekommt.

- Die Trennung zwischen Netzwerk- und Host-Teil der Adresse wird abgeschafft. Statt dessen wird Adress-Aggregierung unterstützt, so dass das Routing hierarchisch durchgeführt werden kann. Allerdings treten dann immer noch Probleme auf bei *multi-homing* und bei Kunden, die ihren Provider wechseln, aber ihre IP-Adresse(n) behalten möchten.

3 Routing

3.1 Aufgaben des Routers

Router sind die Knotenpunkte im Datennetz (sei es ein firmen- oder campusweites Netz oder sei es das Internet). Bei ihnen laufen verschiedene Datenleitungen zusammen, an denen entweder kleinere Netzsegmente hängen können oder die die Verbindungen zu größeren, überregionalen Netzabschnitten oder Providern darstellen.

Sie müssen entscheiden, welche Pakete wohin weitergeschickt werden sollen. Dabei geht es zunächst um die Entscheidung, über welche Datenleitung das betreffende Paket sein Ziel am schnellsten oder am billigsten erreicht (*forwarding lookup*). Wenn der Router gleichzeitig als Firewall arbeitet, muss er auch noch Sicherheitsüberlegungen mit einbeziehen.

Unter Umständen sind zusätzlich noch Dienstgüte-Anforderungen zu beachten (*Quality of Service, QoS*), also maximale Verzögerungszeiten, minimale Datenraten, maximale Verlust- oder Fehlerraten.

Die Aufgaben als Firewall und der QoS-Anforderungen sind nur lösbar, indem mehrere Felder des IP-Kopfes analysiert werden. Üblicherweise werden hier auch die Portnummern des TCP- bzw. UDP-Protokolls mit einbezogen. Aufgrund einer Menge von Filterregeln wird dann eine Entscheidung getroffen, ob ein Paket weitergeschickt werden darf und wenn ja, wie schnell und über welche Ausgangsleitung. Die Untersuchung mehrerer Felder hinsichtlich einer Menge von Regeln bezeichnet man als *Mehrfeld-Klassifikation* oder *multi field classification*, kurz auch MF-Klassifikation.

3.2 Funktionsweise eines Routers

Ein Router lässt sich grob in drei Teile gliedern (vgl. [Bier00]):

1. Auf der Eingangsseite gibt es mehrere Ports, auf denen IP-Pakete eintreffen. Diese werden nach Filterregeln klassifiziert und mit Hilfe einer Datenbank einem Ausgang zugeordnet. Dieser Teil wird uns in dieser Arbeit beschäftigen.
2. Über ein routerinternes Netz werden die Pakete dann dem entsprechenden Ausgang zugeordnet.
3. Im Ausgangsbereich werden sie an der jeweiligen Ausgangsleitung vom *packet scheduler* weggeschickt. Der packet scheduler ist für die Einhaltung von eventuell vorhandenen QoS-Anforderungen zuständig.

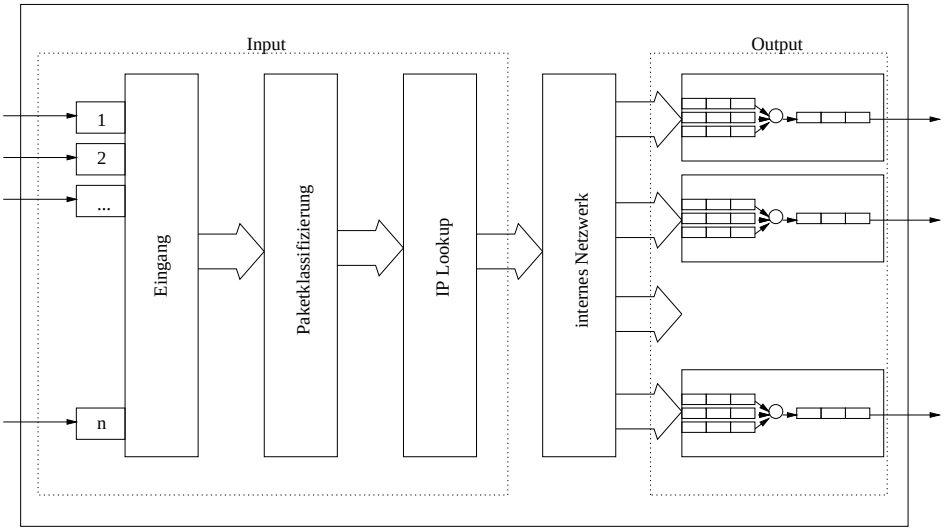

Abbildung 1: Vereinfachtes Schema eines Routers

Betrachten wir den Bereich des IP lookup etwas genauer: Üblicherweise sucht der Router nach dem längsten Präfix in seiner Datenbank, das auf die Adresse des zu behandelnden Pakets passt und ordnet dem Paket die zugehörige Ausgangsleitung zu. Das Problem, das in möglichst kurzer Zeit gelöst werden muss, ist also die Suche nach einem *best matching prefix (BMP)*. Es existieren auch Verfahren, die nicht nur die Zieladresse nach einem solchen längsten Präfix untersuchen, sondern auch noch andere Felder des IP-Headers mit einbeziehen. Sie werden bei der Paketklassifizierung eingesetzt. Zu beiden Möglichkeiten werden eine Reihe von Verfahren vorgestellt. Zwei Algorithmen (je einer zu beiden Ansätzen) sollen etwas detaillierter betrachtet werden.

3.3 Aktuelle Zahlen und Fakten mit Bedeutung für das Routing

Es wird unterschieden zwischen *enterprise router*, die in Unternehmen oder kleineren Campusnetzen eingesetzt werden, und *backbone router*, die in großen, überregionalen Netzen verwendet werden, welche die kleineren Netze miteinander verbinden.

Sie unterscheiden sich in der Zahl der Routingtabellen-Einträge und in der Tatsache, dass Backbone-Router keine *default*-Einträge haben dürfen – sie müssen für jedes bekannte Ziel (-netz) einen Eintrag haben.

Um heutige Routing-Systeme besser einordnen zu können, sollen an dieser Stelle ein paar interessante für einen Backbone-Router typische Zahlen genannt werden (vgl. [Bier00]):

- 75% aller Pakete sind kleiner als 552 byte.
- etwa 50% aller Pakete sind kleiner als 44 byte (z. B. TCP-Acknowledgement-Pakete, http-Anforderungen etc.)
- nur 10% der Pakete sind größer als 1500 byte.
- Es gibt gleichzeitig sehr viele verschiedene Verbindungen (auf höherer Ebene, auf IP-Ebene spricht man von *flows*): oft sind mehrere 10.000 zu beobachten. Dies macht caching von oft angesprochenen Adressen fast unmöglich.
- Die Verteilung der Präfix-Längen in den Routingtabellen weist oft bei den Längen 16 Bit und 24 Bit ein Maximum auf, da dies die Netzadressen von Klasse B- bzw. C-Netzen sind. Fast alle Einträge haben Längen zwischen 10 Bit und 24 Bit.
- Eine einfache Rechnung verdeutlicht, wie zeitkritisch der gesamte Routingvorgang ist: Bei einer Leitung mit einer Kapazität von 1 GBit/s und einer durchschnittlichen Paketgröße von z. B. 500 byte muss der Router seine Arbeit innerhalb von 4 μs erledigt haben, will er nicht zum Nadelöhr werden.

  Setzen wir 10 GBit/s und durchschnittlich 100 byte ein, dann erhalten wir sogar eine obere Grenze von nur 80 ns.

4 Aktuell eingesetzte und neue Techniken

Es wird immer in der Datenbank nach dem längsten Präfix gesucht (*Best Matching Prefix, (BMP)*), das auf die Zieladresse des Paketes passt. Dafür gibt es verschiedene Algorithmen, die meist auf Tries, Grid-Files, Hashing oder Kombinationen davon aufbauen.

Zu berücksichtigen ist dabei sowohl die Performance heute als auch die Skalierbarkeit, d. h.: Wie verhält sich das Verfahren, wenn die Zahl der erreichbaren Rechner steigt? Weiterhin ist bei der Wahl des verwendeten Algorithmus' der Aufwand für die Erzeugung und Aktualisierung der Datenstrukturen von Bedeutung. Dies trifft für Backbone Router noch mehr zu als für Enterprise Router, da die betrachtete Netztopologie sich eher ändern kann (durch Leitungsausfall, „Datenstau“ auf bestimmten Teilstrecken, hinzukommende oder wegfallende externe Teilnetze etc.) als bei überschaubareren Teilnetzen, und da sie schlicht die größeren Routing-Tabellen besitzen.

Nicht zu unterschätzen ist auch der Speicherbedarf, der bei einer großen Anzahl von Einträgen in der Routing-Tabelle durchaus ins Gewicht fällt. Er entscheidet auch darüber, ob Datenstrukturen im schnelleren Cache-Speicher liegen können, oder ob sie in den „normalen“ Hauptspeicher gelegt werden müssen.

Routing-Verfahren unterscheiden sich grundsätzlich dadurch, ob sie nur die Zieladresse betrachten oder auch weitere Felder mit einbeziehen, ob sie Dienstgütezusicherungen unterstützen oder nicht, und ob sie als statisch oder dynamisch bezeichnet werden. Statische Verfahren müssen ihre Datenstrukturen nach Änderungen komplett neu aufbauen, während dynamische Algorithmen ihre Datenstrukturen inkrementell und in vertretbar kurzer Zeit aktualisieren können.

Alle Verfahren können prinzipiell in Hardware oder in Software realisiert werden. Hardwarebasierte Lösungen arbeiten in der Regel etwas effizienter; Softwarelösungen haben aber den Vorteil, wesentlich billiger (solange es nicht um Massenherstellung geht) und zudem noch flexibler zu sein.

Wir werden Algorithmen kennenlernen, die in Software auf einem Standard-PC sehr gute Ergebnisse erzielen.

Dieser Abschnitt gibt einen Überblick über verschiedene gängige Verfahren; die beiden folgenden Abschnitte 5 und 6 beschreiben zwei Verfahren etwas genauer.

4.1 Tries

Das Wort „Trie“ kommt von re*trie*val (vgl. [OtWi93], S. 400ff). Sie werden auch als „alphabetische Suchbäume“ bezeichnet, da die Einträge im Trie durch zeichenweisen Vergleich gefunden werden. Die folgende Abbildung verdeutlicht die Idee.

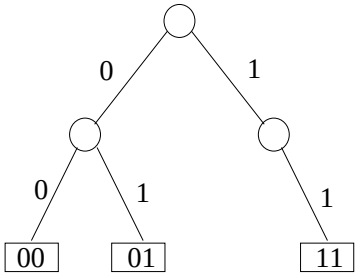

Abbildung 2: Beispiel eines Tries

In diesem Beispiel ist der Weg (1, 1) mit dem Ergebnis „11“ bereits ab der ersten „1“ eindeutig und könnte verkürzt dargestellt werden, so dass das Blatt „11“ bereits nach der ersten „1“ erreicht wird. Diese Variation nennt man *Patricia Trie*.

Der Kernel des BSD-Systems enthält eine Implementation des *radix trie* (eine Variation des *Patricia Trie*), die häufige Verwendung in unserem Zusammenhang findet. Mit W als Länge der Adresse liegt der Aufwand im schlechtesten Fall in $O(W)$, kann also bis zu 32 bzw. 128 (teure) Speicherzugriffe kosten.

Selbst im besten Fall und unter Verwendung von binärer Verzweigung können für n Präfixe in der Datenbank $\log_2(n)$ Speicherzugriffe benötigt werden (vgl. [SrVa97] und [Bier00]).

Allerdings ist es möglich, durch ein Kompressionsverfahren die Größe eines solchen Tries so zu reduzieren, dass er in den sehr viel schnelleren Cache passt und damit deutliche Performance-Gewinne erzielen kann.

4.2 Hashing-Verfahren

Hashing-Verfahren scheinen auf den ersten Blick untauglich für die Suche nach dem längsten Präfix, da sie ja eine exakte Suche auf dem ganzen Schlüssel, nicht aber auf einem Teil

des Schlüssels ermöglichen. Wie Hashing doch sehr effizient eingesetzt werden kann, zeigt [WVTP97] und wird in Abschnitt 5 vorgestellt.

4.3 Grid-File

Hashverfahren ermöglichen Zugriff auf ein eindimensionales Feld von Werten. Der *Grid-File*-Algorithmus (vgl. [OtWi93], S. 254ff) erweitert die Idee des Hashing auf mehrere Dimensionen und ermöglicht so auch eine Bereichssuche.

Verfahren, die mehrere Felder eines Paketkopfes untersuchen, um die anzuwendende Filterregel zu finden, können einen solchen Algorithmus enthalten.

Die Grundidee ist, den Raum so in mehrere Rechtecke einzuteilen, die jeweils einen Datenblock enthalten. Für jede einzelne Dimension wird ein eindimensionales Hashverfahren verwendet, um die richtige Region zu ermitteln. Die Einteilung des Raumes in Regionen bewirkt, dass räumlich nahe beieinander liegende Punkte in einem Datenblock gespeichert werden. Dies ist günstig für Bereichsanfragen.

4.4 Hardware-orientierte Techniken

Ein häufig eingschlagener Weg ist es, *content addressable memory (CAM)* einzusetzen, um die BMP-Suche zu implementieren. Allerdings benötigt man dann 32 CAMs für IPv4 und 128 CAMs für IPv6, und das ist relativ teuer. In der Vergangenheit konnten auch CAMs nicht mit der Entwicklung bei RAM-Speicher Schritt halten. Daher ist es gut möglich, dass eine solche Lösung schon bald von einer Softwarelösung übertroffen wird, die auf einem schnelleren Rechner läuft.

Auch Caching-Techniken schneiden (zumindest bei Backbone-Routern) nicht sehr gut ab, da die volle IP-Adresse (nicht nur ein Präfix) abgelegt werden muss. Die Anzahl der Adressen, die gespeichert werden müssten, ist einfach zu groß.

Zu diesen Anmerkungen vgl. auch [SrVa97], Abschnitt 2.

4.5 Protokollbasierte Verfahren

Es existieren *proprietäre Erweiterungen* des IP-Protokolls wie z. B. IP-Switching und Tag-Switching. Die Idee dabei ist, das Problem der BMP-Suche zu vermeiden, indem einem Router von seinem Vorgänger ein Index in seine Routing-Tabelle übergeben wird. Damit kann mit einem einzigen Speicherzugriff die Routing-Entscheidung getroffen werden. Da solche Verfahren auf einer Erweiterung des Protokolls beruhen, nennt man sie *protokollbasierte Verfahren.*

Ein solches Vorgehen bringt Effizienzgewinne, wenn die Zahl der Router, die diese Protokollerweiterung kennen, groß genug ist, und der so optimierte Datenfluss umfangreich genug ist.

Für kurze Datenflüsse wird keine Performance-Steigerung erzielt, da ja zumindest am Rand eines derart ausgestatteten Teilnetzes und zu Beginn eines Datenflusses auch innerhalb des Teilnetzes immer noch „klassisch“ (also durch IP lookup) geroutet werden muss. Ein weiterer Nachteil ist, in die Vielfalt der schon parallel eingesetzen Protokollvarianten noch weitere einzuführen. Das könnte zu Unübersichtlichkeit und in der Folge zu Instabilität führen (vgl. hierzu [SrVa97], Abschnitt 3, und [WVTP97], Abschnitt 2).

5 Ein Verfahren zum reinen IP-Lookup

Das Papier [WVTP97] stellt einen sehr interessanten Algorithmus vor, der auch in anderen Arbeiten zu diesem Thema zitiert wird, und der nach Angaben der Autoren auch in kommerziellen Produkten eingesetzt werden soll.

Seine Vorteile liegen in der Möglichkeit, es sowohl in Hardware als auch in Software effizient implementieren zu können, und in sehr gutem Verhalten im durchschnittlichen und akzeptablem Verhalten im schlechtesten Fall.

5.1 Beschreibung des Verfahrens

Das Verfahren beruht auf der Verknüpfung von drei wesentlichen Ideen: *Hash-Tabellen* erlauben einen schnellen Zugriff (in $O(1)$), *binäre Suche* ist ein sehr schnelles Suchverfahren, und *Vorberechnungen* vermeiden Backtracking und erlauben eine Suche in $O(\log_2 W)$.

5.1.1 Binäre Suche auf Hash-Tabellen

Für jede Präfix-Länge L existiert eine Hash-Tabelle $H[L]$. Zur Veranschaulichung kann man sich einen Trie vorstellen, in dem jeder Knoten der Tiefe L ein Präfix der Länge L enthält. Die Präfixe aller Knoten einer Ebene der Tiefe L sind in der Hashtabelle $H[L]$ eingetragen.

Im Folgenden sind mit dem Begriff *obere Hälfte* des Tries oder der Hashtabelle alle kürzeren Präfixe, mit *untere Hälfte* alle längeren Präfixe gemeint. Abbildung 3 skizziert diese Struktur.

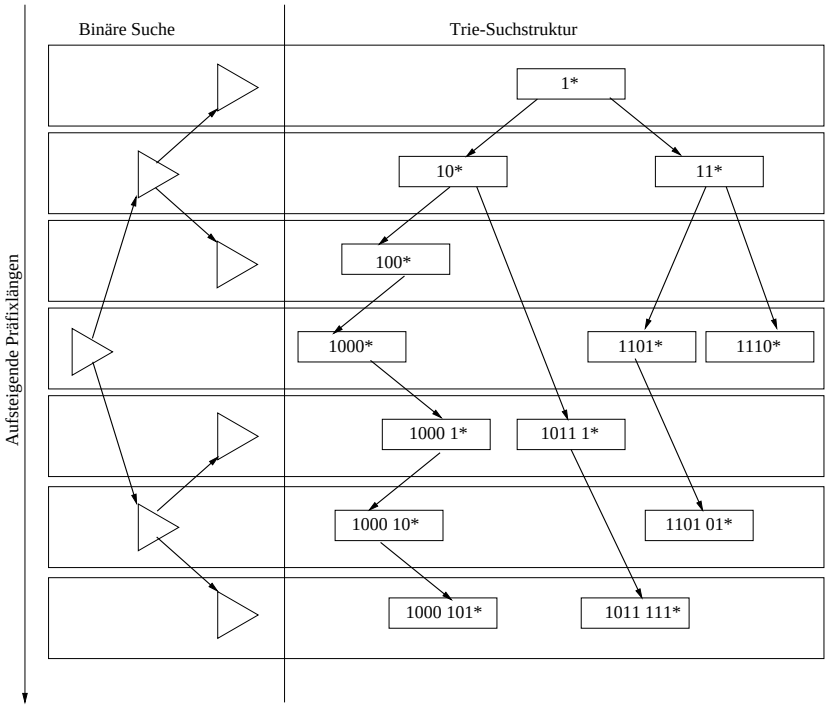

Abbildung 3: Beispiel für Binäre Suche über Trie-Ebenen

Ein naiver Ansatz zur Suche des längsten Präfixes der Adresse D wäre, mit der längsten Länge max zu beginnen, die ersten max bits aus D nach D' zu extrahieren und in $H[max]$ nach D' zu suchen. Wenn ein Eintrag gefunden wird, dann ist dies das längste Präfix. Andernfalls suchen wir in der Tabelle des nächstkürzeren Präfixes $H[l]$ nach einem Eintrag und fahren so fort, bis alle Längen durchsucht sind.

Zwar ermöglicht das Hashing Zugriff auf die Tabelle in konstanter Zeit, doch führt die lineare Suche zu einer Laufzeit von $O(W)$. Deshalb soll statt der linearen Suche binäre Suche über die Längen verwendet werden.

Wir beginnen also beim Median der Längen mit der Suche. Wird ein Eintrag gefunden, so wird in der Hälfte der längeren Einträge weitergesucht, sonst in der Hälfte der kürzeren Präfix-Längen. Die binäre Suche führt zu einer Laufzeit von $O(\log_2 W)$ statt $O(W)$.

Damit das Verfahren allerdings funktioniert, müssen in die Hashtabellen noch weitere Elemente eingeführt werden, und zwar sogenannte *Marker*. Um das zu veranschaulichen, sei ein Beispiel genannt: Gegeben seien die Präfixe $P_1 = 0, P_2 = 00, P_3 = 111$. Gesucht werden soll nach dem längsten Präfix für die Adresse $D = 1111$. Die binäre Suche startet bei dem mittleren Eintrag, P_2, findet nichts passendes und würde demzufolge bei P_1 weitersuchen, obwohl das richtige Ergebnis bei P_3 liegt.

Um dieses Problem zu lösen, muss bei der Länge 2 ein *Marker* eingefügt werden, der für alle Einträge mit Länge größer als 2 die binäre Suche in die richtige Richtung weiterleitet, in diesem Fall also ein Eintrag 11 mit dem Hinweis auf vorhandene längere Präfixe, die mit 11 beginnen. Wenn jetzt die binäre Suche bei P_2 beginnt, findet sie den Marker 11 und sucht bei P_3 erfolgreich weiter.

Nun könnte der Eindruck entstehen, dass die Zahl der benötigten Marker sehr groß sei. Dem ist aber nicht so, denn Marker müssen nicht in allen Hashtabellen für kürzere Präfixe stehen, sondern nur in denen, die von der binären Suche auch tatsächlich besucht werden. Dadurch ist die Zahl der benötigten Marker durch $O(\log_2 W)$ begrenzt und bleibt in der Praxis oft sogar noch darunter, da identische Marker von mehreren längeren Präfixen nur einmal eingetragen werden.

5.1.2 Vorberechnung zur Vermeidung von Backtracking

Der oben angegebene Algorithmus arbeitet aber noch nicht korrekt. Betrachten wir dazu eine Variation des obigen Beispiels mit $P_1 = 1, P_2 = 00, P_3 = 111$ und der Adresse $D = 1100$. Es sei bei $H[2]$ ein Marker 11 eingefügt. Die binäre Suche würde zunächst P_2 und dann P_3 durchsuchen, bei P_3 jedoch nicht fündig werden.

Würde Backtracking eingesetzt werden, um diesen Fehler zu korrigieren, wäre die Laufzeit aber wieder linear und nicht mehr logarithmisch. Das kann durch Vorberechnung wie folgt vermieden werden. Zu jedem Marker wird beim Aufbau der Datenstruktur das längste Präfix *für diesen Marker* ermittelt und gespeichert. Wenn die binäre Suche durch einen Marker in die untere Hälfte verwiesen wird, merkt man sich das längste Präfix dieses Markers. Bricht die binäre Suche ab, ohne selber ein längstes Präfix zu finden, so ist das des zuletzt benutzten Markers das gesuchte längste Präfix.

Dieser Algorithmus funktioniert korrekt; seine Zeitkomplexität liegt in $O(\log_2 W)$. Für IPv6 und $W = 128$ bit sind also maximal 7 Zugriffe in den Hauptspeicher nötig, für IPv4 mit $W = 32$ bit sind es nur 5 Zugriffe.

5.1.3 Verfeinerungen des Algorithmus

Die binäre Suche beschreibt eine Reihenfolge, in der die Hashtabellen durchsucht werden. Man kann sich das vorstellen als einen Weg durch einen binären Baum, bei dem im Knoten eine Präfix-Länge vermerkt ist; wird bei dieser Länge ein passender Eintrag gefunden, der für ein Präfix steht, so ist die Suche beendet; wird ein Marker gefunden, wird beim rechten Sohn die Suche fortgesetzt, sonst beim linken Nachfolgeknoten.

Die Entscheidung, ob links oder rechts weitergesucht werden soll (in unserem Problem heißt das also: ob bei kürzeren oder längeren Präfixen weitergesucht werden soll), beruht nur darauf, ob ein Eintrag gefunden wurde oder nicht.

Mutating binary search

An dieser Stelle kann eine weitere Optimierung des Verfahrens ansetzen. In die Entscheidung kann noch die Information mit einbezogen werden, welches das bisher längste gefundene Präfix ist. Dann kann gezielt bei einer Präfix-Länge weitergesucht werden, die Präfixe enthält, welche mit dem bisher gefundenen längsten Präfix beginnen. So wird die Zahl der noch zu untersuchenden Präfix-Längen weiter reduziert.

Die Suche führt also nicht mehr nur durch einen einzigen binären Suchbaum, sondern durch ein Netzwerk von miteinander verbundenen Bäumen. Jeder Marker M enthält also nicht nur das für den Marker beste Präfix M', sondern auch einen Verweis auf einen neuen binären Suchbaum, der die Suche nur noch zu solchen Hashtabellen führt, in denen Präfixe oder Marker enthalten sind, die mit M beginnen.

Wenn eine Hashsuche erfolgreich ist, dann wird dem Verweis auf den neuen Suchbaum gefolgt und dort die Suche fortgesetzt. Wo die Suche fortgesetzt wird, ist also abhängig vom bisher besten Präfix und damit von der Adresse D, nach deren *BMP* gesucht wird.

Wenn eine Hashtabelle keinen passenden Eintrag enthält, so wird nicht der Suchbaum gewechselt, sondern beim linken Sohn weitergesucht.

Die Autoren nennen diese Variation *mutating binary search*, da die Verzweigungsentscheidung nicht statisch vorhersagbar in einem Baum erfolgt, sondern die Suchbäum im Laufe der Suche gewechselt werden.

Hier wird die Tatsache ausgenutzt, dass in einer Hashtabelle zusammengesetzte Einträge gespeichert werden können, im Gegensatz zur üblichen binären Suche, die nur über einfachen, sortiert abgelegten Schlüsseln sucht.

Allerdings ist für die Erstellung der Datenstrukturen Vorberechnung notwendig, so dass die Zeit zum Einfügen eines neuen Präfixes erhöht wird. Des weiteren steigt der Speicherbedarf für die zusätzlichen binären Suchbäume.

"Rope search"

Betrachtet man die *mutating binary search*, so fällt auf, dass nicht die kompletten binären Suchbäume benötigt werden. Denn wenn ein passender Eintrag in einer Hashtabelle gefunden wird, dann wird ja die Suche bei einem anderen Baum fortgesetzt.

Es ist also nur die Folge der Knoten interessant, die bei wiederholten Fehlversuchen besucht werden soll. Eine solche Folge nennen die Autoren "Rope" (Seil), da man sich so von Suchbaum zu Suchbaum schwingen kann. Ein solches *Rope* kann sehr effizient gespeichert werden. Es enthält Zeiger auf höhere Ebenen, also kürzere Präfix-Längen; es kann also höchstens $\log_2 W$ Zeiger enthalten. Für jede Präfix-Länge wird festgelegt, ob sie durchsucht werden soll oder nicht, es reicht also ein einziges Bit für jede Länge aus. Ein ganzes *Rope* kann also in maximal $\log_2 W$ Bit gespeichert werden. Der daraus resultierende Algorithmus ist schnell, und er ist elegant zu formulieren (s. [WVTP97], Abschnitt 4.2).

Erster Zugriff über ein Array

Als weitere Optimierung kann der erste Zugriff auf eine Hashtabelle über ein indiziertes Array erfolgen. Hierfür können zum Beispiel die ersten 16 Bit der Adresse als Index für ein Array verwendet werden, in dem jeder Eintrag i sowohl das beste Präfix für i als auch ein *Rope* enthält, mit dem die Suche begonnen werden soll.

Ein solcher Zugriff ist nicht nur schneller als ein Hash-Zugriff, sondern er reduziert auch die durchschnittliche Zahl der darauf folgenden Zugriffe (auf etwa 0,5 laut [WVTP97]).

5.1.4 Aufbau der Datenstruktur

Die für den Aufbau de Datenstruktur benötigte Zeit liegt im einfachen Fall in $O(N \log_2 W)$. Für die „Rope Search“ ist der Aufbau komplizierter und erfordert zwei Durchläufe, da jeder Weg einzeln optimiert werden muss. Er benötigt $O(N \max(W, \log_2(W)))$. Die Autoren streben jedoch eine Verbesserung dieser Zeit auf $O(N \log_2(W))$ an.

5.2 Analyse und Bewertung

Die Verfahren wurden mit einer gängigen Router-Datenbank getestet. In der Variante „Rope Search“ mit erstem Zugriff über ein Array geben die Autoren eine Zugriffszeit von durchschnittlich 80ns, maximal 450ns an (zum Vergleich die Zugriffszeiten mit einer Implementation, die sich auch den *Radix Trie* stützt: 1400ns bzw. 2000ns). Der Speicherplatzbedarf sei in beiden Varianten gleich mit 1,2 MB.

Es sei nochmal darauf hingewiesen, dass dieses Verhalten erreicht wird, ohne Annahmen über den Inhalt der Daten zu machen oder Protokolländerungen zu verlangen (die ja über einen definierten Bereich hinaus nicht garantiert werden können).

6 Paket-Klassifikationsverfahren

Die Aufgaben eines Routers sind aber nicht allein auf IP-Lookup beschränkt. Router, die gleichzeitig als Firewall fungieren, müssen entscheiden, welche Pakete in welcher Richtung durchgelassen werden und welche nicht. Dazu liegt eine Reihe von Regeln vor, die verschiedene Felder sowohl des IP- als auch des TCP-Paketkopfes betrachten.

Manche Anwendungen können nur funktionieren, wenn bestimmte Dienstgütequalitäten (engl. *Quality of Service, QoS*) vereinbart und eingehalten werden. Auch hier reicht reines IP-Routing nicht aus; es müssen mehrere Felder betrachtet und ausgewertet werden.

Dazu bildet man *Filterregeln*[1] über die zu betrachtenden Felder. Die Felder sind in der Regel Quell- und Ziel-IP-Adressen, Quell- und Zielports, Protokollflags und eventuell noch mehr. Eine Filterregel enthält zu einem, mehreren oder allen Feldern entweder konkrete Werte oder Wertebereiche bzw. Präfixe. Die Zahl der betrachteten Felder nennt man auch *Dimension* des Filterproblems.

Eine Filterdatenbank, die aus einer Menge von Filtern besteht, zu durchsuchen, erfordert im generellen Fall entweder einen Speicheraufwand von N^d, oder der Zeitaufwand liegt in $O(N)$ mit N als Zahl der Filter (vgl. [SrSV98]). Allerdings kann man die spezielle Struktur solcher Datenbanken ausnutzen, um zu besserem Verhalten zu gelangen. Das in [SrSV98] präsentierte Verfahren wird in Abschnitt 6.2 vorgestellt.

[1] Die Begriffe Filter, Regel und Filterregel werden hier und im Folgenden synonym verwendet.

Im [BoSS98] wird ein Verfahren vorgestellt, das auf einer Kombination von Binomial-Bäumen (vgl. dazu auch [OtWi93], S. 428ff) und Tries (vgl. [OtWi93], S. 400ff) beruht. Interessant an diesem Paper sind vor allem die genaue Dokumentation der Testumgebung, die sehr realitätsnah gewählt scheint, sowie die Analyse der Testergebnisse.

6.1 Einige Verfahren im Überblick

Gupta und McKeown (vgl. [GuMc99] stellen zwei dynamische Klassifikationsalgorithmen vor, d. h. Algorithmen, die während der Laufzeit eine Anpassung der Datenstrukturen vorsehen, die Datenstrukturen also nicht immer wieder komplett neu aufgebaut werden müssen. Sie verwenden *Heap-on-Trie (HoT)* und *Binarysearchtree-on-Trie (BoT)*

Sie analysieren die beiden Algorithmen „nur“ theoretisch und legen keine Versuchsergebnisse vor. Also können sehr gut Aussagen über das worst-case Verhalten gemacht werden, aber keine über das durchschnittliche Verhalten.

Srinivasan, Suri, Varghese: “Packet Classification using Tuple Space Search”. Die typischerweise große Zahl von Filterregeln wird hier auf „Tupel“ abgebildet; danach ist sogar eine einfache lineare Suche über diesen tupel space deutlich schneller als naive Suche über alle Filter. Es wird in Abschnitt 6.2 vorgestellt.

Speziell für eine Umgebung, in der Dienstgütezusicherung eine große Rolle spielt, wird in [BoSS98] ein Verfahren vorgeschlagen, das auf Tries und Binomial-Bäumen basiert. Es untersucht mehrere Felder der Paketköpfe und teilt die Pakete so in Dienstklassen (*service classes*) ein. In diesem Paper wird sehr schön die sehr praxisnahe Testumgebung beschrieben.

Das Verfahren *PathFinder* (vgl. [SrSV98]) vermeidet lineare Suche über alle Filterregeln, indem nur solche Regeln erlaubt werden, die Jokerzeichen am Ende haben. Für Filter der Form *(Ziel-Adresse D, Quell-Adresse S, Protokoll Prot, Zielport DestPort, Quellport SrcPort)* werden nur Filter wie *(D, S, *, *, *)*, nicht aber Filter wie *(D, *, Prot, *, SrcPort)* erlaubt. Mit dieser Einschränkung können die Filter in einen Trie in Verbindung mit Hashtabellen an Stelle der Knoten abgelegt werden, und die Suchzeit liegt in ($O(M)$ (M ist die Zahl der Filterfelder).

Caching und lineare Suche ist ab einer bestimmten Größe der Filterdatenbank keine erfolgversprechende Lösung, da selbst eine Trefferquote im Cache von 80 oder gar 90 Prozent zu einer recht langsamen Laufzeit führt.

Der Ansatz *Multiplane Grid-of-Tries* zerlegt das mehrdimensionale Problem in mehrere zweidimensionale Ebenen und kann mit einer Kombination der Verfahren Grid-File und Trie diese zweidimensionalen Probleme lösen. Allerdings skaliert diese Lösung nicht sehr gut, wenn die Zahl der Felder wächst: Eine typische Firewall-Datenbank würde bis zu acht solcher Ebenen benötigen, und jedes Teilproblem benötigt bis zu acht Speicherzugriffe. Das bedeutet bis zu 64 (teure) Zugriffe in den Hauptspeicher, und das ist zu langsam.

Für *Hardware-Lösungen* gilt, was auch oben schon gesagt wurde: Sie können durch maßgeschneiderte Hardware Parallelität einsetzen und damit Geschwindigkeit gewinnen, sind aber immer dem Risiko ausgesetzt, zu schnell zu veralten.

6.2 Tuple Space Search

Srinivasan, Suri und Varghese beobachten in [SrSV98], dass trotz einer unter Umständen hohen Zahl an Filterregeln einige Regelmäßigkeiten ausgenutzt werden können, um sowohl Speicherplatz als auch Laufzeit der Paket-Klassifikation zu reduzieren.

Sie betrachten im Einsatz befindliche Firewall-Klassifikations-Datenbanken und stellen fest, dass es bei allen Regeln nur relativ wenige unterschiedliche Kombinationen von Feldlängen gibt. Die unterschiedlichen Regeln lassen sich also einteilen in eine Menge von *Tupeln*, wobei ein solches Tupel eine Kombination von Feldlängen ist. Ein Beispiel hierzu: Das Präfix „129.13.*“ der Ziel-IP-Adresse hat die Länge 16. Ein Filter der Form *(129.13.*, 129.13.64.*, *, *, *)* wird dem Tupel *(16, 24, 0, 0, 0)* zugeordnet. Die Menge der Tupel, die benötigt werden, um alle Regeln der Filterdatenbank zu beschreiben, wird *Tuple Space* genannt.

Für die Einteilung von Portnummern kann nicht die Präfix-Länge verwendet werden. Statt dessen werden Bereiche angegeben, die sich nicht überlappen dürfen; aber ein Bereich darf vollständig in einem anderen enthalten sein. Jeder Bereich bekommt eine eindeutige ID zugewiesen, und es ergibt sich eine bestimmte *Verschachtelungstiefe*. Dann kann im Tupel an Stelle der Länge die Verschachtelungstiefe angegeben werden, und im Filter wird die ID des Bereiches eingetragen.

Ein Tupel kann also betrachtet werden als Suchmaske für eine Reihe von Filtern.

Die Überprüfung, ob ein Paket auf ein solches Tupel passt, ist sehr effizient mit Hilfe von Hashing möglich, wenn zu jedem Tupel T eine Hashtabelle vorliegt, in der alle Filter abgelegt sind, die durch das Tupel T beschrieben werden. Das Tupel T wird als Maske auf den Paketkopf angewendet. Aus den daraus resultierenden Bits wird der Hashschlüssel errechnet.

Man beachte, dass dieses Verfahren aufgrund der Eindeutigkeit der Tupel schnelle Updates der Datenbank erlaubt.

In dem in [SrSV98] genannten Beispiel wird die Filterdatenbank mit 278 Regeln auf 41 Tupel reduziert. Selbst eine einfache lineare Suche über diese Tupel bringt erhebliche Zeitersparnis.

6.2.1 Verbesserungen für Tuple Space Search

Um die Suche zu beschleunigen, wird die Menge der Tupel reduziert, die jeweils durchsucht werden muss. Um das zu erreichen, sind verschiedene Möglichkeiten denkbar.

Eine Idee ähnelt dabei der im Abschnitt 5.1.2 vorgestellten Verbesserung der Suche über die Hashtabellen beim IP-Lookup.[2] In jede Hashtabelle werden Marker eingeführt, die eine Einschränkung für die weitere Suche erlauben.

Eine weitere Möglichkeit ist, eine optimale Reihenfolge bei den zu durchsuchenden Tupeln im Vorhinein zu ermitteln oder durch eine Heuristik festzulegen. Im ersten Fall wird ein Entscheidungsbaum erstellt, in dem abhängig vom Suchergebnis (Filter gefunden oder nicht) verzweigt wird; zu jedem Knoten gehört dabei ein bestimmtes Tupel, das dann betrachtet wird. Allerdings benötigt die Erstellung eines solchen Entscheidungsbaumes exponentielle Zeit, und im schlechtesten Fall wird nichtmal Zeit bei der Suche gewonnen.

Srinivan, Suri und Varghese beobachten, dass in den von ihnen untersuchten Filterdatenbanken nur wenige verschiedene Präfixe zu einer gegebenen Adresse existieren. Diese Eigenschaft kann ausgenutzt werden, um die Zahl der Tupel dramatisch zu verkleinern. Die Autoren nennen diese Heuristik *Tuple Pruning*. Im genannten Beispiel, das 278 Filterregeln auf 41 Tupel abbilden konnte, reduziert diese Optimierung die Tupelmenge auf 11 „verkürzte“ Tupel.

7 Ausblick

Die hier vorgestellten Verfahren zeigen, dass die Router den aktuellen und kommenden Anforderungen gewachsen sein werden und nicht zum Flaschenhals im Netz werden müssen.

[2] Bei beiden Verfahren ist G. Varghese einer der Autoren.

Dennoch ist klar, dass es immer noch mehr Anwendungen geben wird, die immer noch mehr Übertragungskapazität und -qualität verlangen werden. Deshalb muss in diesem Bereich natürlich noch weitergeforscht werden, um damit Schritt halten zu können.

Literatur

[Bier00] Prof. Dr. Ernst Biersack. Algorithms for Forwarding Lookup and Filtering in High Performance Routers. Networking 2000 Tutorials, Februar 2000.

[BoSS98] Niklas Borg, Emil Svanberg und Olov Schelén. *Efficient Multi-field Packet Classification for QoS Purposes.* Telia Research AB, Luleå, Schweden, Computer Science and Electrical Engineering, LuleåUniversity of Technology, Luleå, Schweden. 1998.

[GuMc99] Pankaj Gupta und Nick McKeown. *Dynamic Algorithms with Worst-case Performance for Packet Classification.* Computer Systems Laboratory, Stanford University, Stanford, CA. 1999.

[OtWi93] Thomas Ottmann und Peter Widmayer. *Algorithmen und Datenstrukturen, 2. Auflage.* Mannheim, Leipzig, Wien, ZÜrich: BI Wissenschaftsverlag. 1993.

[SrSV98] V. Srinivasan, S. Suri und G. Varghese. *Packet Classification using Tuple Space Search.* Deptartment of Computer Science, Washington University, St. Louis, MO. 1998.

[SrVa97] V. Srinivasan und G. Varghese. *Faster IP Lookups using Controlled Prefix Expansion.* Deptartment of Computer Science, Washington University, St. Louis, MO. 1997.

[WVTP97] Marcel Waldvogel, George Varghese, Jon Turner und Bernhard Plattner. *Scalable High Speed IP Routing Lookups.* Proceedings of SIGCOMM 97, vol. (October). 1997.

TCP-Varianten

Thomas Lange

Kurzfassung

TCP bereitet beim Einsatz auf verschiedenen Netzwerktypen und Übertragungsmedien unterschiedliche Probleme, die von den jeweiligen Besonderheiten des Netwerks und des Übertragungsmediums herrühren, und die ansteigende Größe und Komplexität des Internets bereitet Probleme bei der Flußkontrolle. Es werden verschiedene Erweiterungen von TCP vorgestellt, welche die Fluss- und Staukontrolle verbessern und neue Übertragungsmedien unterstützen sollen. Zur Verbesserung der Flusskontrolle wurden die Erweiterungen Slow-Start, Congestion Avoidance, Fast Retransmit und Fast Recovery bereits eingeführt. In der Mobilkommunikation müssen häufigere Übertragungsfehler, Hand-Offs und eine begrenzte Bandbreite unterstützt werden. In der Hochgeschwindigkeitskommunikation müssen die festen Konstanten und zu kleinen Felder des TCP-Protokolls zur besseren Flusskontrolle angepasst werden. Für transaktionsorientierte Dienste wird vorgeschlagen, den 3-Wege-Handshake zu verbessern.

1 Einleitung

Als auf der Transportschicht angesiedeltes Protokoll stellt TCP eine transparente Ende-zu-Ende Verbindung zwischen den Endsystemen her. Die Übertragung der Daten muss dabei zuverlässig erfolgen und eventuelle Übertragungsfehler müssen vom Protokoll eventuell durch wiederholte Übertragung behandelt werden.

Aufgrund des ISO-OSI-Schichtenmodells ist TCP unabhängig von den darunterliegenden Schichten implementiert. Neue technische Entwicklungen wie Mobil- und Hochgeschwindigkeitskommunikation sowie gestiegene Anforderungen an Staukontrolle und Dienstgüteunterstützung machen jedoch Anpassungen von TCP an die veränderte Netzwerkarchitektur notwendig, um trotzdem eine effiziente Datenübertragung zu ermöglichen. Eine komplette Abkehr von TCP ist wegen der weiten Verbreitung nicht sinnvoll.

Die Erweiterungen von TCP lassen sich in folgende, hier vorgestellte Gruppen einteilen:

- Erweiterungen zur Fluss- und Staukontrolle
- Erweiterungen für die Mobilkommunikation
- Erweiterungen für die Hochgeschwindigkeitskommunikation
- Erweiterungen für transaktionsorientierte Dienste

Um Kompatibilität mit Kommunikationspartnern, die eine bestimmte Erweiterung von TCP nicht kennen, sicherzustellen, werden die zu verwendenden Erweiterungen beim Aufbau der Verbindung mit SYN-Paketen ausgehandelt.

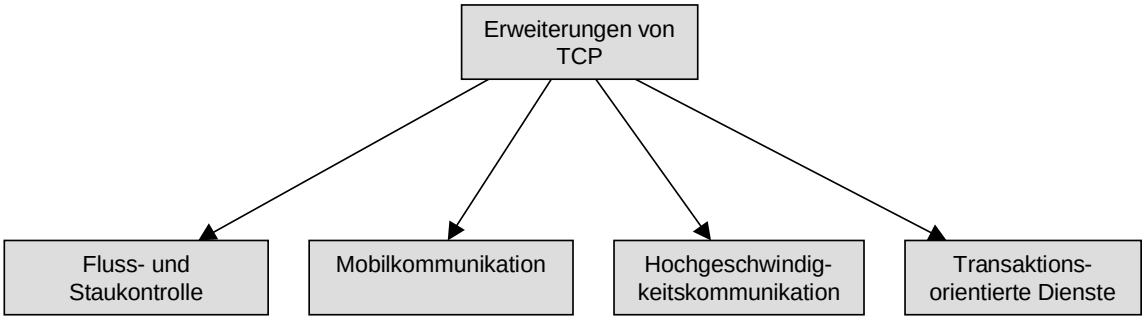

Abbildung 1: Erweiterungen von TCP

2 Fluss- und Staukontrolle

Ein wesentliches Merkmal des Protokolls TCP ist die eingebaute Staukontrolle [Stev97]. Diese sorgt dafür, dass nicht mit maximaler Geschwindigkeit Datenpakete in das Netz eingespeist werden, sondern dass die Sendeleistung an die zur Verfügung stehende Bandbreite und an Bandbreitenschwankungen angepasst wird. Dabei wird auch berücksichtigt, dass es weitere Datenströme auf dem selben Pfad durch das Netzwerk geben kann. Die Bandbreite muss dann gerecht zwischen allen Datenströmen aufgeteilt werden.

Ursprüngliche Versionen von TCP achten nur darauf, dass der Empfänger der Daten nicht überlastet wird. Beim Aufbau der Verbindung teilt der Empfänger dem Sender die Sendefenstergröße mit. Der Sender darf dann so viele Datenpakete senden, wie durch die Sendefenstergröße erlaubt wurde. Nach jedem Paket, von dem vom Empfänger eine Ermpfangsbestätigung erhalten wurde, darf ein weiteres Paket an den Empfänger gesendet werden. Es können also nie mehr Pakete unterwegs sein, als der Größe des Sendefensters entspricht. Wenn beim Empfänger ein Paket ankommt, das noch nicht erwartet wurde, wird erneut eine Bestätigung für das letzte richtige Pakete an den Sender gesendet. Dieser beginnt die Übertragung dann erneut bei diesem Paket.

Aktuelle Implementierungen von TCP basieren auf folgenden vier grundlegenden Algorithmen zur Staukontrolle: Slow Start, Congestion Avoidance, Fast Retransmit und Fast Recovery.

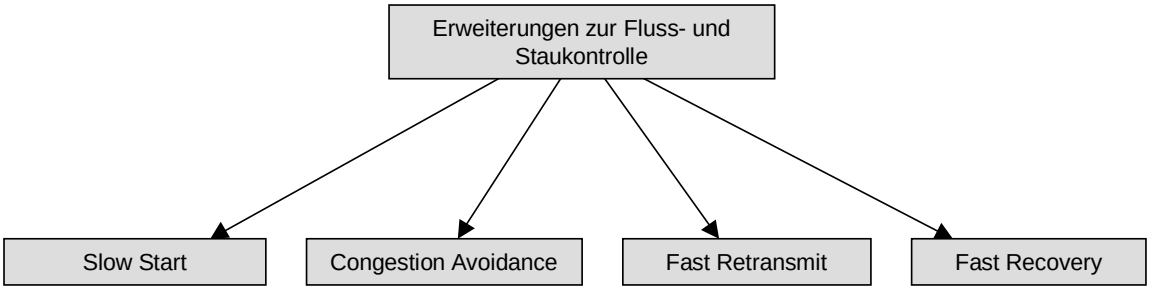

Abbildung 2: Erweiterungen zur Fluss- und Staukontrolle

- Slow Start

Wenn der Sender einfach soviele Datenpakete in das Netzwerk einspeist, wie das vo Empfänger zugebilligte Sendefenster ermöglicht, kann es zu Staus kommen, wenn die Datenpakete über Router oder langsamere Verbindungen weitergeleitet werden müssen. Die Router können die Datenpakete nicht so schnell loswerden, wie sie ankommen. Stattdessen speichert ein Router sie in einem Puffer zwischen. Wenn dieser Puffer voll ist, kann der Router keine weiteren Datenpakete mehr annehmen. Er verwirft sie einfach. Dies führt zu Paketverlusten und daraus resultierenden Übertragungsfehlern. Daraus ergibt sich eine nur geringe Übertragungsrate.

Der „Slow Start"-Algorithmus verhindert dies, indem er langsam mit dem Senden anfängt, und dann die Rate exponentiell erhöht, bis die vom Empfänger zugebilligte Größe des Sendefensters erreicht ist.

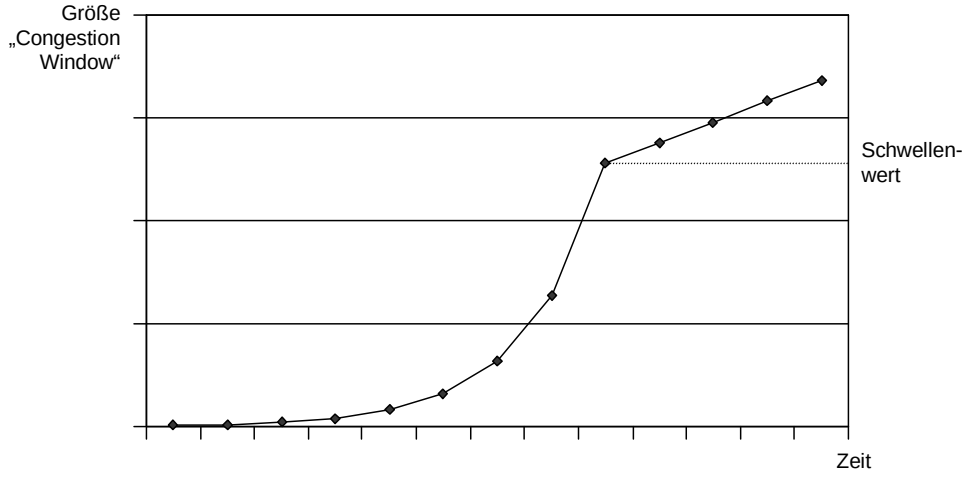

Abbildung 3: Die Größe des Congestion Windows wächst bei Slow Start zuerst exponentiell und dann linear

Dazu wird das *congestion window* eingeführt. Es wird am Anfang mit der Größe eines Segments (Datenpakets) initialisiert. Jedesmal wenn eine Bestätigung eines vom Empfänger empfangenen Pakets beim Sender eintrifft, wird die Größe des *congestion windows* um ein Segment erhöht. Dies führt zu einem exponentiellem Wachstum des congestion windows in der Zeit. Der Sender sendet möglichst so viele Pakete, wie das congestion window zulässt, aber nie mehr, als durch das Sendefenster erlaubt sind. Nach Überschreiten des Schwellenwerts erfolgt die Steigerung linear.

Wenn es unterwegs zu einem Datenstau kommt, und Pakete verworfen werden, ist das *congestion window* zu groß geworden und muss reduziert werden.

Der Slow-Start Algorithmus ist immer zusammen mit der Congestion Avoidance implementiert.

- Congestion Avoidance

  Der Algorithmus geht davon aus, dass es kaum zu Paketverlusten durch Übertragungsfehler kommt, sondern dass Paketverluste meist durch Staus und Pufferüberläufe in Routern entstehen.

  Wenn ein Stau auftritt, der dann entweder durch eine doppelte Bestätigung oder einen Timeout (die Bestätigung bleibt ganz aus) signalisiert wird, wird die Größe des congestion windows halbiert und zwischengespeichert. Wenn der Stau durch einen Timeout angezeigt wurde, wird sie auf den Wert 1 zurückgesetzt. Beim Senden der weiteren Pakete, wird die Größe des congestion windows mit dem Slow Start Algorithmus wieder

laufend verdoppelt, bis der zwischengespeicherte sichere Wert erreicht ist. Danach wird die Größe des congestion windows durch den congestion avoidance Algorithmus nur noch linear vergrößert.

- Fast Retransmit

  Diese 1990 vorgeschlagene Erweiterung von TCP versucht, einzelne verlorengegangene Pakete einfach nachzusenden. Dazu wird, wenn beim Empfänger ein Paket ankommt, obwohl noch nicht alle vorangegangenen Pakete angekommen sind, eine doppelte Bestätigung des Pakets zurückgeschickt, mit dem Hinweis, welches Paket erwartete wurde. Wenn dies ein paar mal nacheinander geschieht, geht der Sender davon aus, dass die Pakete nicht einfach nur unterwegs vertauscht worden sind, sondern dass tatsächlich ein Paket verlorengegangen ist. Dieses wird dann nocheinmal geschickt. So wird vermieden, dass erst weitergesendet wird, wenn ein Time-out für das betreffende Paket geschieht.

- Fast Recovery

  Nachdem ein Fast Retransmit nach einem Paketverlust ausgeführt wird, wird nicht Slow Start angewandt, sondern Congestion Avoidance, da davon ausgegangen werden kann, dass ein einzelner Paketverlust nicht durch einen Stau, sondern einen tatsächlichen Übertragungsfehler verursacht worden ist.

3 Erweiterungen für die Mobilkommunikation

Bei der Mobilkommunikation gibt es mehrere Besonderheiten. Zum einen treten bei drahtloser Übertragung mehr Übertragungsfehler auf, als bei leitungsgebundener Übertragung. Zum anderen können die Endgeräte beweglich sein. Das heisst, es kann zur Vermittlung der Datenpakete nicht immer der gleiche Pfad durch das Netzwerk verwendet werden.

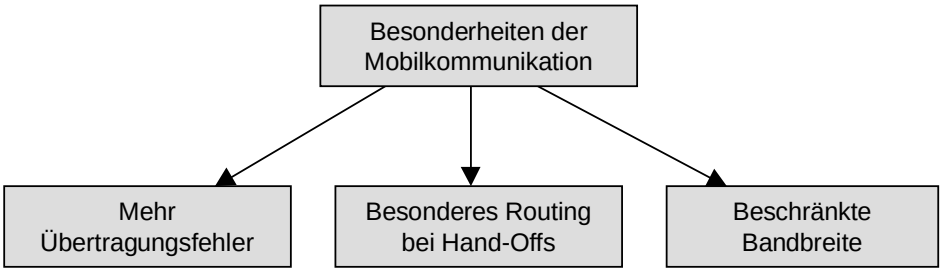

Abbildung 4: Besonderheiten der Mobilkommunikation

Beide Besonderheiten bereiten TCP Probleme. Klassischerweise geht TCP bei einem Paketverlust davon aus, dass es unterwegs zu einem Datenstau mit Pufferüberlauf gekommen ist, und dass das Paket deshalb verworfen wurde. Die Übertragungsrate wird daraufhin reduziert. Bei drahtloser Übertragung kann es jedoch häufig zu Paketverlusten durch Störungen kommen. Wenn ein herkömmliches TCP daraufhin wiederholt die Übetragungsrate reduziert, kommt es zu drastischen Leistungseinbrüchen. Bei einem Hand-Off, wenn das Mobile System seine Basisstation wechselt, kann es zu längeren Pausen in der Datenübertragung kommen, ausserdem können Daten verlorengehen, die zu diesem Zeitpunkt noch auf dem Weg zur bisherigen Basisstation waren.

Das andere Problem besteht im Routing bei der Vermittlung der Datenpakete. Um die Routingtabellen klein zu halten und den Aufwand bei der Vermittlung zu reduzieren, ist der Adressraum von IP hierarchisch geordnet. Ein Teilnetz kennt jeweils nur wenige Übergabestellen an andere unter- oder übergeordnete Netze. Wenn nun ein mobiles Endsystem seine räumliche Position ändert und den Bereich seines Subnetzes verlässt, muss es in ein anderes Subnetz eingegliedert werden. Wenn sich dabei jedoch die IP-Adresse ändert, muss die neue IP-Adresse erst allen potentiellen Kommunikationspartnern bekannt gemacht werden, was mit dem gegenwärtigen Namensdienst „DNS“ nur sehr langsam funktioniert. Bleibt die IP-Adresse gleich, müssen sehr viele Routingtabellen geändert werden, was ebenfalls sehr aufwendig und langsam ist.

Ein drittes Problem ist, dass bei drahtloser Übertragung die Bandbreite aufgrund der physikalischen Eigenschaften des Mediums und staatlicher Regulierung begrenzt ist.

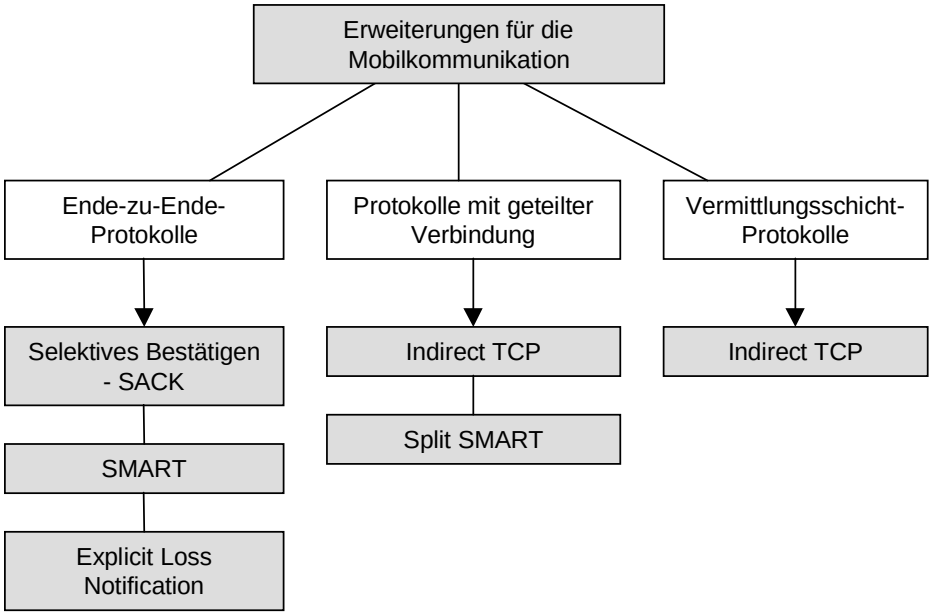

Abbildung 5: Erweiterungen für die Mobilkommunikation

Es gibt jedoch Erweiterungen von TCP, um alle drei Probleme zu lösen:

3.1 Übertragungsfehler

Um das Problem der verlustbehafteten Übertragung zu lösen gibt es mehrere Ansätze: Ende-zu-Ende-Protokolle, Protokolle mit geteilter Verbindung und Vermittlungsschicht-Protokolle [BPSK97].

3.1.1 Ende-zu-Ende-Protokolle

Bei den Ende-zu-Ende-Protokollen werden zwei Verfahren vorgeschlagen, um aus Übertragungsfehlern resultierende Paketverluste zu behandeln. Zum einen wird sogenanntes „selektives Bestätigen“ (SACK) vorgeschlagen, um auch nach einem Verlust von mehreren Paketen innerhalb eines Sendefensters weitermachen zu können, ohne auf einen Timeout warten

zu müssen. Zum anderen wird zwischen Paketverlust durch Übertragungsfehler und Paketverlust durch Stau unterschieden. Das wird mittels einem Verfahren zur expliziten Verlust-Benachrichtigung erreicht.

- SACK - Selektives Bestätigen

  Eine mögliche Lösung stellt eine Erweiterung von TCP zur Übermittlung selektiver Bestätigungen (SACKs, „selective acknowledgements“) dar. Weil bei TCP immer mehrere Pakete auf einmal bestätigt werden, hat der Sender oft nicht genügend Informationen, um auf mehrere Paketverluste innerhalb eines Sendefensters richtig zu reagieren.

  Der Unterschied zwischen normalen Bestätigungen und selektiven Bestätigungen ist, dass bei SACK zusätzlich zu den normalen Bestätigungen noch zusätzliche selektive Bestätigungen gesendet werden, wenn Pakete empfangen wurden, die noch garnicht erwartet wurden. Der Sender kann dann die fehlenden Pakete nochmals senden. Bei normalen Bestätigungen wird jeweils nur das letzte in der richtigen Reihenfolge empfangene Paket bestätigt.

  Die Benutzung von SACK wird beim Verbindungsaufbau ausgehandelt. SACK wurde erstmals in RFC 1072 vorgeschlagen und dann noch einmal in RFC 2018, konnte sich aber nicht als Erweiterung von TCP etablieren.

- SMART

  Eine weitere Implementierung stellt SMART dar. Die von SMART verwendeten Bestätigungen enthalten neben der Bestätigung für das letzte in der richtigen Reihenfolge empfangene Datenpaket noch die Sequenznummer des Pakets, das zum Senden der Bestätigung geführt hat. Der Sender verwendet diese Information, um festzustellen, welche Pakete richtig am Ziel angekommen sind und welche noch nicht. Wenn der Sender eine Lücke in den vom Empfänger bestätigten Datenpaketen feststellt, nimmt er an, dass diese verlorengegangen sind und sendet sie noch ein zweites Mal.

  Ein Nachteil von SMART ist, dass es Pakete, die unterwegs vertauscht worden sind und deshalb in falscher Reihenfolge ankommen, nicht erkennt. Diese Pakete werden dann auch erneut angefordert.

- Explicit Loss Notification

  Dieses Verfahren erweitert die Bestätigung des Empfängers um ein Feld zur expliziten Verlustbenachrichtigung. Wenn der Empfänger erkennt, das ein Paket aufgrund eines Übertragungsfehlers verlorengegangen ist, setzt er dieses Feld bei der nächsten Bestätigung über alle in der richtigen Reihenfolge empfangenen Pakete. Der Sender kann dann dieses Paket erneut senden, ohne dass die Mechanismen zur Staukontrolle benutzt werden müssen.

3.1.2 Protokolle mit geteilter Verbindung

Bei den Protokollen mit geteilter Verbindung wird die drahtlose Übertragungsstrecke völlig vor den Endsystemen verborgen, indem die TCP-Verbindung nur bis zum Anfang der drahtlosen Strecke geht. Sie wird dann in ein Verbindungsprotokoll übersetzt, das besser auf Paketverluste ausgelegt ist. Dabei werden häufig Protokolle mit selektiver Wiederholung (SRP) eingesetzt. Ein Nachteil geteilter Verbindungen ist, dass die Ende-zu-Ende-Semantik der TCP-Bestätigungen verletzt wird, da nun eine Bestätigung beim Sender ankommen kann, noch bevor das Paket den Empfänger erreicht hat.

- Indirect TCP

  Bei Indirect-TCP wird TCP nicht nur für den Teil der Verbindung im Festnetz eingesetzt, sondern auch über die drahtlose Strecke. Dennoch sind die beiden Teile der Verbindung voneinander getrennt, so dass eine Fehlerkorrektur nur auf der drahtlosen Strecke durchgeführt wird. Aufgrund der bekannten Unzulänglichkeiten von TCP führt diese Lösung jedoch nicht zu einer optimalen Übertragungsrate, wenn nicht spezielle Erweiterungen von TCP für die drahtlose Übertragungsstrecke verwendet werden [BaBa95].

- Split SMART

  Hier wird die SMART-Erweiterung für TCP auf der dragtlosen Strecke verwendet. Der Nachteil von SMART, dass vertauschte Pakete nicht erkannt werden, kommt hier nicht so stark zum tragen, da die Wahrscheinlichkeit, dass Pakete auf der drahtlosen Strecke, die oft nur aus einer Verbindung besteht vertauscht werden relativ gering ist.

3.1.3 Vermittlungsschicht-Protokolle

Die Vermittlungsschicht-Protokolle liegen zwischen den beiden anderen Ansätzen. Auch sie versuchen, Paketverluste vor der TCP-Verbindung zu verbergen. Dabei werden verlorengegangene Datenpakete auf der Vermittlungsschicht lokal neu übermittelt. Der Hauptvorteil dieser Protokolle ist, das sie sehr gut in das Schichtenmodell der Netzwerkprotokolle passen. Sie funktionieren unabhängig von Protokollen der höheren Schichten und brauchen keine Zustandsinformationen über die Verbindungen. Darin liegt jedoch auch der Nachteil, denn es kann sein, dass gleichzeitig auch Fehlerkorrekturmechanismen höherer Schichten, z.B. durch Timeouts aktiviert werden. So werden Daten unter Umständen unnötigerweise doppelt übermittelt.

- Snoop Protokoll

  Eine erweiterte Version des Vermittlungsschicht-Protokolls stellt das Snoop Protokoll dar. Vor der drahtlosen Strecke werden alle TCP-Pakete, die über die drahtlose Strecke übermittelt werden sollen, zwischengespeichert und analysiert. Das Snoop-Protokoll erkennt die doppelten Bestätigungen, die bei Paketverlusten gesendet werden. Wenn das Snoop-Protokoll eine doppelte Bestätigung erkennt, wird das betreffende Paket aus dem Zwischenspeicher erneut gesendet und die doppelte Bestätigung wird entfernt. Der Rest des Übertragungsweges durch das Netz wird so entlastet. Das Snoop Protokoll ist kein reines Protokoll der Vermittlungsschicht, da es sich seine Kenntnisse über das übergeordnete TCP-Protokoll der Transportschicht zunutze macht [Bala98].

- Link-Layer SMART

  Link-Layer SMART sendet zur zuverlässigen Übertragung der Daten über die Vermittlungsschicht auf SMART basierende TCP-Bestätigungen. Diese Bestätigungen werden nur dazu verwendet, eine zuverlässige Übertragung über eine einzelne Übertragungsstrecke zwischen zwei Netzwerkknoten sicherzustellen. Die Bestätigungen bleiben vor dem Protokoll der Transportschicht verborgen.

3.2 Routing und Hand-Offs

Da TCP keine Hand-offs unterstützt, müssen diese vor den Endsystemen verborgen werden. Oft geschieht dies durch Tunneln der Verbindung oder durch geteilte Verbindungen.

- Indirect TCP

  Hand-offs zwischen verschiedenen Basisstationen werden durch das spezielle I-TCP Protokoll ermöglicht. Änderungen an den Systemen oder den TCP-Implementierungen im Festnetz sind dabei nicht nötig. Die Verbindung zwischen mobilem System und festem System wird am sogenannten „mobile support router“ (MSR) geteilt. Bei einem Hand-Off wird die Verbindung zwischen den jeweiligen MSRs übergeben.

3.3 Beschränkte Bandbreite

Um Bandbreite zu sparen, können die Daten komprimiert werden. Eine Möglichkeit dazu ist die Kompression der Header des TCP/IP Protokolls. Ipv4-Header haben eine Größe von 20 Bytes. Ipv6-Header werden eine Größe von 40 Bytes haben. Bei der Datenübertragung zu mobilen Systemen werden die Datenströme oft getunnelt. Dabei werden die Pakete jeweils in weitere Pakete eingepackt. Dies lässt den Anteil der Header-Daten noch weiter ansteigen. Eine Möglichkeit zur Header Kompression wird in [DENP] beschrieben. Die Schlüsselbeobachtung, die effiziente Headerkompression erlaubt, ist die Tatsache, dass in Datenströmen, die aus aufeinanderfolgenden Paketen, die von der gleichen Quelle zum gleichen Ziel übermittelt werden und die das gleiche Transportprotokoll benutzen, fast alle Felder gleich sind. Bei einer Verbindung mit komprimierten Headern wird nur der erste Header unkomprimiert übertragen. Es wird ein Schlüssel erstellt, der die entsprechenden Headerdaten eindeutig bestimmt. Bei den darauffolgenden Paketen werden nur noch dieser Schlüssel und die Felder, die sich in jedem Paket ändern können, mitgeschickt. Besonders bei drahtloser Übertragung muss sichergestellt sein, dass durch fehlerhaft übertragene Daten entstandene Fehler richtig erkannt werden.

4 Erweiterungen für Hochgeschwindigkeitskommunikation

Hochgeschwindigkeitskommunikation bezieht sich auf die Kommunikations über Übertragungsstrecken mit hoher Bandbreite. Diese Übertragungsstrecken können schon bei geringer Verzögerung eine sogenannte Pipeline bilden. Das bedeutet, dass große Datenmengen abgesendet werden können, bevor das erste Datenpaket am anderen Ende ankommt. Man spricht hier von einer hohen Pfadkapazität. Dies führt bei Verwendung von TCP über diese Strecken zu verschiedenen Problemen.

RFC 1072, RFC 1185 und RFC 1323 beschäftigen sich mit diesem Thema.

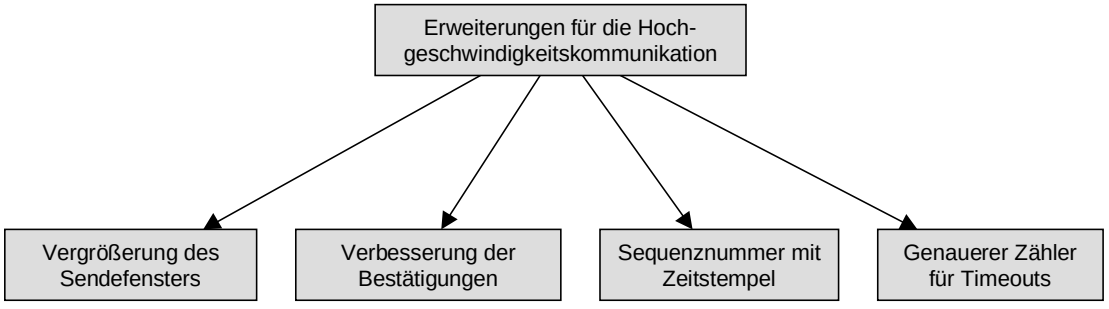

Abbildung 6: Erweiterungen für die Hochgeschwindigkeitskommunikation

- Maximale Größe des Sendefensters zu klein

  Bei TCP wird die Größe des Sendefensters in einem 16 Bit-Feld gespeichert. Die maximale Größe des Sendefensters kann also 64 KB betragen. Diese Datenmenge kann bei Übertragungsstrecken mit hoher Bandbreite sehr schnell gesendet werden. Bis dann aufgrund der Round-Trip-Verzögerung die erste Bestätigung eintrifft, kann so viel Zeit vergehen, dass eine der maximalen Größe des Sendefensters entsprechende Datenmenge bereits gesendet ist und der Sender auf die Bestätigung warten muss, um ein weiteres Paket senden zu können. So kann die zur Verfügung stehende Bandbreite nicht ausgenutzt werden. Zur Lösung dieses Problems wird vorgeschlagen, die Größe des Sendefensters durch einen Faktor zu skalieren. Dann kann die Größe des Sendefensters einen Wert annehmen, der einer Feldgröße von bis zu 32 Bit entspricht. Der zu verwendende Faktor wird beim Aufbau der Verbindung ausgehandelt.

- Kumulierte Bestätigungen

  Da in der ursprünglichen Version von TCP nur die Pakete bestätigt werden, die in der richtigen Reihenfolge ankommen und auch erwartet werden, kommt es bei einem Paketverlust zu einem Timeout, und alle Pakete die danach schon gesendet wurden, werden eventuell noch einmal gesendet. Um diesen Schwachpunkt zu verbessern, wird vorgeschlagen, selektive Bestätigungen zu verwenden. Der Empfänger kann so den Sender genau informieren, welche Pakete empfangen wurden, und welche noch nicht. Bis zur Einführung von Fast Retransmit und Fast Recovery im Jahr 1990 wurde das Problem noch dadurch verschärft, dass nach einem einzelnen Paketverlust alle nachfolgenden nochmals gesendet wurden und zusätzlich der Slow Start-Algorithmus durchgeführt wurde. Bei mehreren Paketverlusten innerhalb eines Sendefensters, was bei zunehmender Größe des Sendefensters immer wahrscheinlicher wird, versagt jedoch auch Fast Retransmit. Einen anderen Weg gehen die Autoren von RFC 1106. Mit negativen Bestätigungen (NAKs) werden dort vom Empfänger Pakete angefordert, die verloren gegangen sind.

- Bestimmung der Round-Trip Time

  Zur Bestimmung der Zeit, nach der ein Timeout erfolgen soll, wird die Round-Trip Time der Pakete gemessen. Die Timeout Zeit wird dann als ein kleines Vielfaches der Round-trip Time festgelegt. Für einen effizienten Datentransport muss diese Round-Trip Time relativ exakt bestimmt werden. Die Messung ist bei Hochgeschwindigkeitsübertragungsstrecken mit großen Sendefenstern und hoher Auslastung aufgrund der kumulierten Bestätigungen sehr ungenau. Man versucht die Genauigkeit der Messungen zu verbessern, indem man jedem Paket einen Zeitstempel mitgibt. Dann braucht man keine komplizierten Kalkulationen mehr durchführen.

- Doppelte Benutzung der selben Sequenznummer

  Für die Sequenznummer eines TCP-Pakets ist lediglich ein Feld mit 32 Bit Länge vorgesehen. Bei Übertragungsstrecken mit extrem hohen Übertragungsraten können sämtliche möglichen Nummer innerhalb kurzer Zeit verbraucht sein, und es müssen Nummern innerhalb des festgelegten sicheren Bereiches von zwei Minuten wiederholt werden. Verzögert sich die Zustellung eines Pakets, kann es sein, dass ein altes Paket mit dem eigentlich richtigen verwechselt wird, was dann zu unerkannten Übertragungsfehlern führt. Außerdem kann es sein, dass verirrte Pakete von früheren Verbindungen mit den gleichen Parametern zum selben Fehler führen können. Besonders akut wird dieses Problem, wenn die maximale Größe des Sendefensters wie oben beschrieben mit einem Faktor erhöht wird. Auch dieses Problem wird mit dem Zeitstempel in jedem Paket umgangen. Anhand des Zeitstempels kann man sehen, ob das Paket noch aktuell oder veraltet ist.

- Staukontrolle

  Um Staus besser zu behandeln, wurden wie oben erläutert zunächst die Mechanismen Fast Retransmit und Fast Recovery im Jahr 1990 eingeführt. Dieser Mechanismus kann bei kleinen Sendefenstern wegen der groben Auflösung des Timeout Zählers von 500 ms jedoch zu großen Stockungen im DatenFluss führen, was sich besonders bei interaktiven Verbindungen wie Telnet-Sitzungen bemerkbar macht. Einfach einen genaueren Zähler zu verwenden, führt zu dem Problem, dass geringe Schwankungen im Datendurchsatz des Netzwerkes nicht mehr herausgefiltert werden und auch zu Timeouts führen. Dieses Problem soll der nicht sehr weit verbreitete Mechanismus der expliziten Staubenachrichtigung (ECN - Explicit Congestion Notification) lösen. Explizite Staubenachrichtigungen können auf der Übertragungsstrecke liegende Router an den Sender abschicken, wenn mehr Datenpakete ankommen, als verarbeitet werden können. Besonders aktuelle Router können technisch dazu in der Lage sein. Die bis 1994 einzige Implementierung dieses Verfahrens stellt „ICMP Source Quench“ dar. Hier werden die Staubenachrichtigungen über das Signalisierungsprotokoll ICMP an den Sender geschickt. Ein Nachteil dieses Verfahrens ist allerdings, dass besonders bei Staugefahr das Netz durch zusätzliche Datenpakete belastet wird. Deshalb ist es sinnvoll, die Rate, mit der die Staubenachrichtigungen ausgesendet werden zu begrenzen [Floy94].

  Ein anderes Verfahren, um Staus zu vermeiden wird in [Brak95] beschrieben. Dieses Verfahren, das vom Autor TCP-Vegas, in Anlehnung an die ursprünglichen Implementierungen von TCP namens Tahoe und Reno genannt wird, ist rein auf die Senderseite beschränkt und benötigt sonst keine Änderungen an TCP. Der Ansatz beruht auf drei Mechanismen.

  - Übertragungswiederholung

    Der Fast Retransmit Algorithmus wird verbessert, indem immer dann, wenn die Timeout Zeit überschritten ist, ein Paket schon bei der ersten doppelten Bestätigung neu gesendet wird. Normalerweise müsste ja auf die dritte doppelte Bestätigung gewartet werden, was aber bei einem kleinen Sendefenster oder großen Verlusten zu einer Blockierung führen kann, da der Sender das Sendefenster bereits gefüllt hat, der Empfänger aber keine Datenpakete mehr erhält, die er - möglicherweise als verloren - bestätigen könnte. Auf diese Weise werden auch viele normale Timeouts verhindert, die sonst aufgetreten wären. Außerdem wird bei diesem Vorgang nicht mehr der grobe 500ms Zeitgeber sondern die Systemuhr zur Bestimmung der Timeout Zeit verwendet. Nach einem Paketverlust reicht sogar schon eine einfache Bestätigung, die nicht innerhalb der Timeout Zeit ankommt, für ein erneutes Senden des entsprechenden Pakets.

  - Stauvermeidung

    Herkömmliche Implementierungen von TCP haben keine Möglichkeit, Staus zu vorherzusehen, bevor sie passieren. Erst bei Paketverlusten wird der Stau erkannt. TCP Vegas setzt hier an und vergleicht fortwährend den erzielten Datendurchsatz mit dem aufgrund des Sendefensters erwarteten Durchsatz. Das Sendefenster wird dann entsprechend angepasst. Dabei wird aber trotzdem immer ein bisschen mehr gesendet, um kurzfristige Verbesserungen der Bandbreite ausnutzen zu können. Dieses Verfahren wird jedoch nicht bei Slow-Start eingesetzt.

  - Slow-Start

    Aufgrund der selbsttaktenden Eigenschaft von TCP, gibt es keine Möglichkeit, die zur Verfügung stehende Bandbreite zu messen, ohne dass Daten gesendet und Bestätigungen empfangen werden. Der Slow Start Algorithmus erfüllt die Aufgabe, die richtige Sendegeschwindigkeit zu finden recht gut. Allerdings muss es stets erst zu einem Stau kommen, bevor die richtige Datenrate festgestellt ist, oder

Änderungen in der Bandbreite erkannt werden. Abhilfe schafft bei TCP Vegas ein veränderter Slow Start. Hier wird die Senderate nur jede zweite Round-Trip Time verdoppelt und in der jeweils dazwischenliegenden Zeit die erzielte Datenrate mit der erwarteten verglichen. Ist die erzielte Datenrate geringer, so wird von einem sich anbahnenden Stau in einem Router ausgegangen und die Datenrate wieder reduziert. So wird der eigentliche Stau vermieden, aber die Bandbreite dennoch genau bestimmt.

Bei Leistungstests stellte sich heraus, dass TCP Vegas 37-71 Prozent besseren Datendurchsatz als das herkömmliche TCP Reno erzielt und dabei genauso fair bei der Ressourcenverteilung ist.

5 Erweiterungen für transaktionsorientierte Dienste

Während TCP gut zur Übertragung von Datenströmen geeignet ist, ist die Leistung bei transaktionsorientierten Diensten nicht optimal. Transaktionsorientierte Dienste sind üblicherweise Anfragen eines Clients an einen Server mit einmaliger Antwort des Servers. Unnötigen Ballast stellt hier zum einen der 3-Wege Handshake von TCP dar. Bevor die eigentlichen Daten übertragen werden können, vergeht so eine ganze Round-trip Time. Zum anderen befindet sich die Verbindung nach Beendigung der Datenübertragung noch 120 Sekunden im Time-Wait Status und der Port bleibt solange für weitere Verbindungen gesperrt. Da insgesamt nur 2 hoch 16 Ports zu Verfügung stehen, kann auch dies zur Begrenzung der Kapazität des Servers führen. Transaction/TCP [Brad94] ist eine Erweiterung von TCP aus dem Jahr 1994, die diese Nachteile beseitigt. Es wird die Time-Wait Zeit verringert und der 3-Wege Handshake umgangen. Dies geschieht, indem schon beim ersten SYN vom Client an den Server die Daten für die Anfrage mitgeschickt werden. Bei der Bestätigung des Servers wird dann gleich die Antwort mitgeschickt. Sollte der Server länger brauchen, als die Time-Out Zeit erlaubt, wird eine separate Bestätigung abgeschickt. Mit Hilfe dieses Verfahrens erhält der Client seine Antwort schon nach der Round-Trip Time zuzüglich der Bearbeitungszeit des Servers.

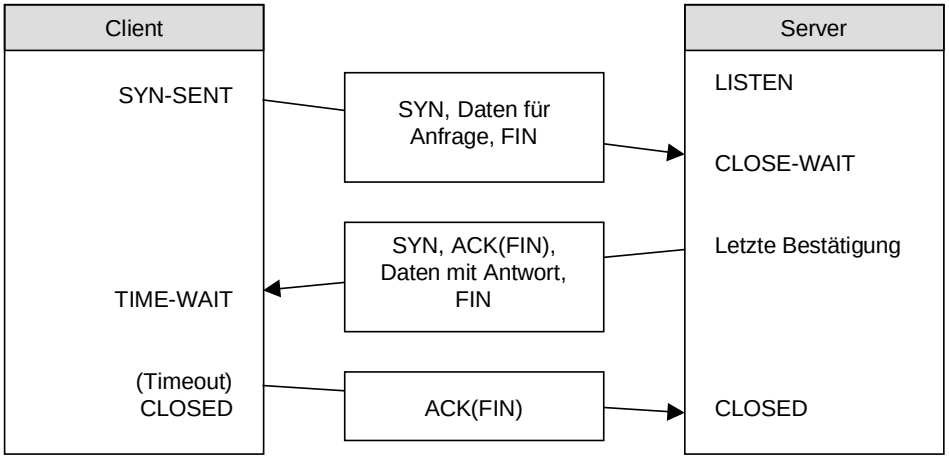

Abbildung 7: Daten werden schon während des 3-Wege-Handshakes ausgetauscht

6 Zusammenfassung

Es wurden verschiedene Erweiterungen des TCP-Protokolls vorgestellt. Diese versuchen, die Probleme zu lösen, die TCP beim Einsatz auf verschiedenen Netzwerktypen bereitet.

Die ersten Erweiterungen von TCP dienten zur Verbesserung der Fluss- und Staukontrolle. Die Verfahren Slow-Start, Congestion Avoidance, Fast Retransmit und Fast Recovery wurden bereits 1988 und 1990 eingeführt und sind überall verbreitet.

In der Mobilkommunikation gibt es bei der drahtlosen Datenübertragung drei Besonderheiten: Es gibt mehr Übertragungsfehler; bei einem Hand-Off muss sich der Pfad der Daten durch das Netz ändern, und die Bandbreite ist begrenzt. Zum lösen dieser Probleme werden Verfahren vorgeschlagen, die das Auftreten von Übertragungsfehlern und den Verlust einzelner Pakete besser behandeln. Dazu gehören Selektives Bestätigen, SMART und Explicit Loss Notification. Andere Ansätze versuchen, die drahtlose Übertragungsstrecke von der leitungsgebundenen zu trennen und jeweils ein Übertragungsverfahren einzusetzen, das die jeweiligen Besonderheiten berücksichtigt. Dazu gehören Indirect TCP, Split SMART, das Snoop-Protokoll und Link-Layer SMART. Zur Unterstützung von Hand-Offs wird meist eine gleichbleibende Anlaufstelle für die ankommenden Daten verwendet. Diese schickt die Daten dann weiter an das mobile System. Auch dazu gehört das Verfahren Indirect TCP. Der beschränkten Bandbreite versucht man durch Header-Kompression zu begegnen.

Zentrale Probleme in der Hochgeschwindigkeitskommunikation sind die nicht an das schnelle und breitbandige Medium angepassten festen Konstanten des TCP-Protokolls und die zu kleinen Feldgrößen. Man versucht, dieses Problem mit erweiterten Feldern und einer genaueren Bestimmung der Round-trip Time und Flusskontrolle zu lösen.

Für transaktionsorientierte Dienste wird vorgeschlagen, den 3-Wege-Handshake zu beseitigen, um so die Antwort des Kommunikationspartners schon in kürzerer Zeit zu erhalten.

Seit 1990 konnte sich im Internet keine Erweiterung von TCP mehr in größerem Umfang etablieren. Das liegt wohl an dem großen Aufwand zur Standardisierung und Implementierung neuer Protokolle und an der Tatsache, dass die Datenübertragung im Internet mit dem bestehenden TCP-Protokoll dennoch recht gut funktioniert.

Literatur

[BaBa95] A. Bakre und B. R. Badrinath. Handoff and System Support for Indirect TCP/IP. In *Proc. Second Usenix Symp. On Mobile and Location-Independent Computing*. April 1995.

[Bala98] H. Balakrishnan. TCP Behaviour of a Busy Web Server: Analysis and Improvements. In *Proc. IEEE INFOCOM*. März 1998.

[BPSK97] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Sehsan und Randy H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. In *IEEE/ACM Transactions on Networking*. Dezember 1997.

[Brad94] R. Braden. RFC 1644 T/TCP – TCP Extensions for Transactions. Juli 1994.

[Brak95] Lawrence S. Brakmo. TCP Vegas: End to End Congestion Avoidance on a Global Internet. In *IEEE Journal on Selected areas in Communications Vol 13 No. 8*. Oktober 1995.

[DENP] Mikael Degermark, Mathias Engan, Björn Nordgren und Stephen Pink. Low-loss TCP/IP Header Compression for Wireless Networks.

[Floy94] Sally Floyd. TCP and Explicit Congestion Notification. Oktober 1994.

[Stev97] W. Stevens. RFC 2001 TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms. Januar 1997.

Wireless Application Protocol

Tanjev Stuhr

Kurzfassung

Im Zeitalter des Internet und der Mobilkommunikation war es unweigerlich der nächste Schritt, via Mobiltelefon ins Internet zu gelangen. Doch ergaben sich einige Probleme, wie etwa die geringe Bandbreite bei der Mobilkommunikation oder die geringen Ressourcen, die ein Mobiltelefon mit sich bringt, wie beispielsweise Speicher, Prozessorleistung oder die Display-Größe. Es wäre sehr schwierig, HTML-Text auf einem Handy-Display anzuzeigen, da dieses i.A. relativ klein ist. Um diese Probleme in den Griff zu bekommen, schlossen sich 1997 verschiedene Firmen aus den Bereichen Hardware, Software und Mobilkommunikation zum so genannten WAP-Forum zusammen. WAP steht dabei für Wireless Application Protocol. Bereits im darauffolgenden Jahr wurde ein Standard verabschiedet, der versucht, diese Probleme zu lösen – WAP 1.0.

1 Einleitung

In diesem Abschnitt wird eine kurze Einführung über WAP gegeben. Im Anschluss daran wird die noch junge Geschichte von WAP betrachtet.

1.1 Was ist eigentlich WAP?

WAP steht für Wireless Application Protocol und ist seit der CeBit 2000 auch ein Begriff für die Öffentlichkeit. Es wurde vom WAP-Forum, einer Vereinigung vieler Unternehmen der Hardware, Software und des Mobilfunks, entwickelt, mit dem Ziel einer einheitlichen Spezifikation zur Entwicklung von Anwendungen über drahtlose Kommunikationsnetzwerke.

WAP beschreibt nicht nur ein einziges Protokoll, sondern eine ganze Protokoll-Familie. Ähnlich wie bei der bekannten Internet-Protocol-Familie (IP) gibt es auch hier viele verschiedene Protokolle, die jedes für sich ein eigenes Aufgabengebiet abdecken. Es handelt es sich hierbei um Protokolle, die drahtlose Kommunikation ermöglichen. Genauer gesagt ist es mittels der WAP-Technologie möglich, mit mobilen Endgeräten wie zum Beispiel mit einem WAP-fähigen Handy, einem PDA, einem Palmtop oder einem Laptop ins Internet zu gelangen. Die Übertragung wird durch verschiedene Protokolle gewährleistet. Mittels einer noch detaillierter betrachteten Technik (Gateways) wird eine „normale“ Internet-Seite so angepasst, dass sie auf einem Handy angesehen werden kann und durch sie hindurch navigiert werden kann. Es ist ein eigenes Format für die Darstellung von Internet-HTML-Seiten (HTML=Hypertext Markup Language) auf WAP-fähigen Endgeräten vom WAP-Forum entwickelt worden – WML (Wireless Markup Language).

1.2 Die Geschichte des WAP

Unter den verschiedenen Entwicklungen in der Informationstechnologie, die allesamt relativ jung sind, ist WAP noch mit die jüngste. Auch wenn das Internet erst seit Mitte der 90-ger Jahre eine weite Verbreitung in der Öffentlichkeit findet, wurden erste Ansätze bereits um 1969 vom Department of Defence (DoD) der USA gemacht. WAP hingegen ist erst im Jahre 1997 vom WAP-Forum, dem bei der Gründung im Juni 1997 die Firmen Ericsson, Motorola, Nokia und Unwired Planet angehörten, ins Leben gerufen worden. Dabei wurde bei der Zusammensetzung des Forums, dem heute schon über 200 Unternehmen angehören, darunter mittlerweile auch AT&T, Intel, VTT oder HP, genau darauf geachtet, dass keine der drei Fraktionen Hardware, Software und Mobilfunkanbieter die Überhand bekam. Ein weiterer großer Vorteil des Forums war und ist die Kooperation mit anderen Normierungsgremien wie European Standards Institute (ETSI), Cellular Telecommunications Industry Association (CTIA), World Wide Web Consortium (W3C), Telecommunications Industry Association (TIA), Internet Engineering Task Force (IETF).

Bereits im April 1998 wurde die erste Version der WAP-Spezifikationen – eine Sammlung von Protokollen aus verschiedenen Schichten – veröffentlicht. Damit war die Basis für Hard- und Softwarehersteller gegeben. Die Ziele eines Standards waren und sind dabei folgende:

- Problemlose Darstellung von Inhalten auf mobilen Endgeräten
- Ein übergreifender Standard für alle mobilen Netzwerktechnologien
- Aufbau auf bestehenden Standards, so weit als möglich

Von der Version 1.0 zu 1.1 wurde dann ein großer Sprung gemacht, so dass die Abwärtskompatibilität nicht gewährleistet wurde. Dies sorgte in der Industrie verständlicherweise für Unruhe. Deshalb wurde bei der Einführung der neuesten Version (Version 1.2) darauf geachtet, dass die Abwärtskompatibilität gewahrt blieb und lediglich einige Erweiterungen hinzugekommen sind. Welche Funktionalitäten genau hinzugekommen sind, entnehme man bitte dem Abschnitt 4.

Kaum ist diese Version von „Proposed“ auf „Approved“ heraufgesetzt worden, hört man schon von einer neuen Version – Version 1.3. Sie steht quasi schon in den Startlöchern.

1.3 Gliederung

Der Abschnitt 2 beschäftigt sich nun mit den oben bereits erwähnten Protokollen. Im Anschluss daran wird in Abschnitt 3 die Technik vorgestellt, mit der Internet-Seiten auf WAP-fähigen Endgeräten angezeigt werden können. Im Abschnitt 4 wird abschließend noch auf Neuerungen der neuen Version eingegangen und in Abschnitt 5 eine Zusammenfassung sowie ein Ausblick in die Zukunft gegeben.

2 WAP

Die WAP-Architektur ist – ähnlich wie das ISO-/OSI-Modell – ein Schichtenmodell. Sie besteht aus sechs Schichten (vgl. Abbildung 1). Auch ähneln die einzelnen Schichten denen des ISO-/OSI-Modells. In den folgenden Abschnitten werde ich auf die einzelnen Schichten und die zugehörigen Protokolle näher eingehen (siehe auch [Schi00]).

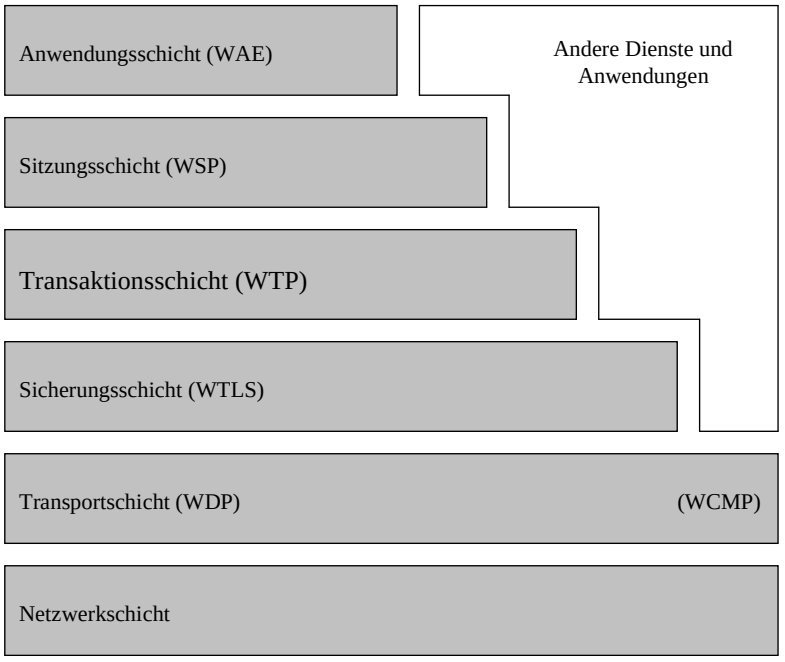

Abbildung 1: WAP-Architektur

2.1 Netzwerkschicht

Die unterste Schicht ist die Netzwerkschicht. Hier gibt es verschiedene Träger (engl. bearer) wie zum Beispiel GSM, die verschiedene Dienste bieten. WAP spezifiziert lediglich die Schnittstellen zu diesen Diensten, nicht aber die Dienste selbst. Allerdings benutzt es existierende Datendienste und wird in Zukunft noch weitere Dienste integrieren. Beispiele für diese Dienste sind SMS über GSM, das wohl jedem geläufig ist, HSCSD (High-Speed Circuit Switched Data) von GSM und GPRS (General Packet Radio Data) von GSM. Viele weitere Träger werden von WAP unterstützt, darunter CDPD, IS-136, PHS.

2.2 Transportschicht

Die Transportschicht mit dem Wireless Datagram Protocol (WDP) und dem Wireless Control Message Protocol (WCMP) bietet den darüberliegenden Schichten einen konsistenten und vom Träger unabhängigen datagrammorientierten Dienst. Damit wird dem Träger gegenüber jegliche Kommunikation transparent gemacht. Die gemeinsame Schnittstelle, die unabhängig von dem zugrundeliegenden Netzwerk von den höheren Schichten benutzt wird, ist der T-SAP (Transport Layer Service Access Point). Die Anpassungen, die in der Transportschicht benötigt werden, um diesen konsistenten Dienst zu liefern, können abhängig vom Dienst des Trägers stark schwanken. Dabei gilt: Je dichter der Trägerdienst an IP angelehnt ist, desto geringer sind die Anpassungen. Bietet der Träger bereits IP, wird UDP, das Datagramm-Protokoll des Internet, als WDP benutzt. Man kann dann also sagen, dass WDP mehr oder weniger die gleichen Dienste anbietet, die auch UDP anbietet.

WDP bietet Quell- und Zielports, die zum Multiplexen bzw. Demultiplexen genutzt werden. Die Dienstprimitive, die von dieser Schicht benutzt werden, sind T-DUnitdata.req, T-DUnitdata.ind und T-DError.ind. Dabei wird zum Senden einer Nachricht das Dienstprimitiv T-DUnitdata.req benutzt (vgl. Abbildung 2). Die Parameter, die dabei übergeben werden, sind die Quell- und Zieladresse, die eindeutig sind (z.B. eine Telefonnummer oder eine IP-Adresse), der Quell- und Zielport und die Nutzdaten. Zum Anzeigen einer Nachricht wird

das Dienstprimitiv T-DUnitdata.ind verwendet, das nur noch die Parameter Quelladresse, Quellport und Nutzdaten hat. Zieladresse und Zielport sind optional verwendbar. Wenn WDP einen angeforderten Dienst einer darüber liegenden Schicht nicht bieten kann, wird das Dienst-Primitiv T-DError.ind, das als Parameter einen Error Code übergibt, genutzt. Es werden damit aber nur lokale Fehler wie etwa eine zu große Nutzdaten-Größe angezeigt.

Um Fehler zu signalisieren, die beim Senden von Datagrammen von einer WDP-Instanz zu einer anderen auftauchen, gibt es das Wireless Control Message Protocol (WCMP). Es bietet Fehlerbehandlungsmechanismen und beinhaltet Steuerungsnachrichten, die denen des Internet Control Message Protocol (ICMP) für IPv4 ähneln. Im Falle von IP-basierten Netzwerken kann ICMP als WCMP verwendet werden. Die Nachrichten, die in WCMP verwendet werden, können auch zu Diagnose- und Informationszwecken eingesetzt werden.

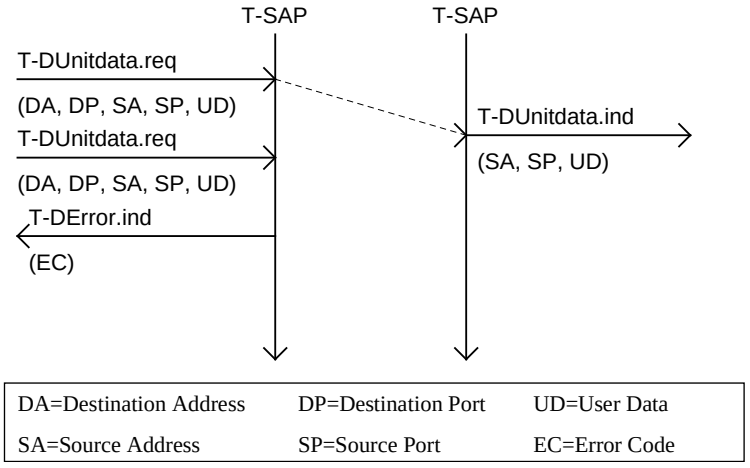

Abbildung 2: WDP-Dienstprimitive

2.3 Sicherungsschicht

Die nächste Schicht ist die Sicherungsschicht mit dem Sicherungsprotokoll Wireless Transport Layer Security (WTLS). Sie ist optional und kann, wenn es eine Anwendung anfordert, in die WAP-Architektur integriert werden. Sie hat die Aufgabe, eine sichere Verbindung und entsprechende Sicherheitstechniken, die auch in Standardnetzwerken eingesetzt werden, wie etwa Authentizität, Vertraulichkeit und Datenintegrität, bereitzustellen. WTLS wurde für geringe Bandbreiten und Netzwerke mit hohen Verzögerungszeiten optimiert. Des Weiteren berücksichtigt es mobile Endgeräte mit geringer Prozessorleistung und beschränkter Speicherkapazität für Verschlüsselungsalgorithmen. WTLS unterstützt sowohl datagrammorientierte als auch verbindungsorientierte Protokolle. Neu im Gegensatz zu GSM ist die Sicherheit zwischen zwei Teilnehmern und nicht mehr nur zwischen mobilem Endgerät und der Basisstation. WTLS hat viele Merkmale des TLS (Transport Layer Security), dem früheren SSL (Security Socket Layer) übernommen.

Bevor Daten über WTLS ausgetauscht werden können, wird eine sichere Verbindung aufgebaut (unter der Sicherungsschicht befindet sich ja nur ein verbindungsloses Protokoll). Beim so genannten „full handshake“ geschieht dies mittels folgender Schritte (dabei kann jeder Teilnehmer zu jeder Zeit die Verbindung abbrechen, etwa wenn die vorgeschlagenen Parameter nicht akzeptabel sind): Zunächst sendet der Initiator ein SEC-Create.req mit den Parametern Quell- und Zieladresse, Quell- und Zielport sowie einige die Verschlüsselung betreffende Parameter (vgl. Abbildung 3). Beim Empfänger wird das Dienstprimitiv SEC-Create.ind ausgeführt, womit ihm der Aufbau einer gesicherten Verbindung angezeigt wird. Daraufhin

antwortet der Partner mit einem SEC-Create.res und übergibt seinerseits einige Parameter für die Verschlüsselung und einen Session Identifier, einer einzigartigen Kennung für jede Verbindung. Als nächstes führt der Partner das Dienstprimitiv SEC-Exchange.req aus, mit dem er seinen Wunsch anzeigt, dass er eine Public-Key-Authentisierung durchführen möchte (z.B. fragt der Partner beim Initiator nach einem Zertifikat). Auf der Seite des Initiators werden nun die Dienstprimitive SEC-Create.cnf, das die Antwort des Partners anzeigt, und SEC-Exchange.ind, das die Anfrage des Partners nach Authentifizierung anzeigt, ausgeführt. Der Initiator antwortet jetzt zum einen mit SEC-Exchange.res, wobei er den Parameter Certificate übergibt, und zum anderen sendet er SEC-Commit.req, das dem Partner anzeigt, dass die Verbindung steht. Der Partner führt nun noch SEC-Commit.ind aus, woraufhin der Initiator SEC-Commit.cnf ausführt, womit die Verbindung steht.

Nun können auf einer sicheren Verbindung Daten mittels SEC-Unitdata.req und SEC-Unitdata.ind übertragen werden, wobei die Parameter wieder die Quell- und Zieladresse, der Quell- und Zielport sowie die Nutzdaten sind.

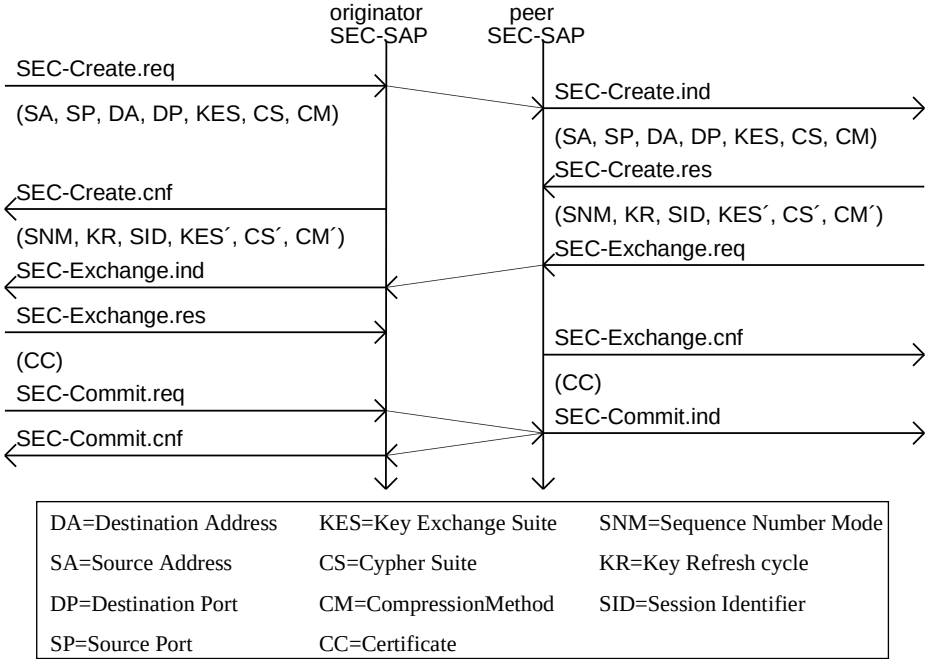

Abbildung 3: Aufbau einer sicheren Verbindung

2.4 Transaktionsschicht

Die Transaktionsschicht mit dem Wireless Transaction Protocol (WTP) befindet sich oberhalb der Transportschicht oder – wenn Sicherheit verlangt wird – oberhalb der Sicherungsschicht. Sie wurde für den Betrieb auf sehr kleinen Clients (wie z.B. Handys) entwickelt und bietet den höheren Schichten verschiedene Vorteile wie z.B. eine verbesserte Zuverlässigkeit bei datagrammorientierten Diensten, eine verbesserte Effizienz bei verbindungsorientierten Diensten und eine Unterstützung bei transaktionsorientierten Diensten wie beispielsweise dem Web-Browsen. In diesem Zusammenhang bedeutet „transaktionsorientiert“ die Verbindung von Anfrage und Antwort bei einer Web-Seite.

Das Protokoll bietet drei verschiedene Klassen von Transaktionsdiensten:

- Klasse 0 ist ein unzuverlässiger Nachrichtenaustausch ohne eine Ergebnisnachricht. Da in dieser Klasse kein Acknowledgement stattfindet, ähnelt WTP Klasse 0 dem WDP.

- Klasse 1 ist ein zuverlässiger Nachrichtenaustausch ohne eine Ergebnisnachricht. Dabei gibt es zwei Varianten, die durch eine unterschiedliche Belegung des Parameters A unterschieden werden: Zum einen kann das Dienstprimitiv TR-Invoke.req mit A=Acknowledgement durch die WTP-Einheit (automatic acknowledgement) ausgeführt werden. Dabei bestätigt die WTP-Einheit des Empfängers implizit die empfangenen Daten. Zum anderen kann TR-Invoke.req mit A=Acknowledgement durch den Benutzer (user acknowlegement) ausgeführt werden. Hier muss der Empfänger den Erhalt der Daten explizit mit dem Dienstprimitiv TR-Invoke.res bestätigen.

- Klasse 2 ist ein zuverlässiger Nachrichtenaustausch mit genau einer zuverlässigen Ergebnisnachricht (also der typische Fall von Anfrage und Antwort). Hier gibt ebenfalls bei der Ausführung von TR-Invoke.req die Möglicheit einer Bestätigung durch die WTP-Einheit oder durch den Benutzer. Hinzu kommt noch die Möglichkeit, den Empfang der vom Initiator gesendeten Daten implizit durch die Antwort (TR-Result.req) zu bestätigen. Der Initiator bestätigt seinerseits den Empfang der Ergebnisnachricht explizit durch TR-Result.res.

Das WTP erreicht seine Zuverlässigkeit durch das Löschen von Duplikaten, durch wiederholte Übertragung, durch Acknowledgements (nur Klasse 1 und 2) und durch eindeutige Transaktionskennungen. WTP benutzt drei Dienstprimitive: TR-Invoke zum Initiieren einer neuen Transaktion, TR-Result, um das Ergebnis einer früher initiierten Transaktion zurückzusenden und TR-Abort, das eine bestehende Transaktion abbricht. Die PDUs, die bei diesen Vorgängen ausgetauscht werden, sind die Invoke-PDU, die Ack-PDU und die Result-PDU. Abbildung 4 zeigt ein Beispiel einer Transaktion Klasse 2 (user acknowledgement).

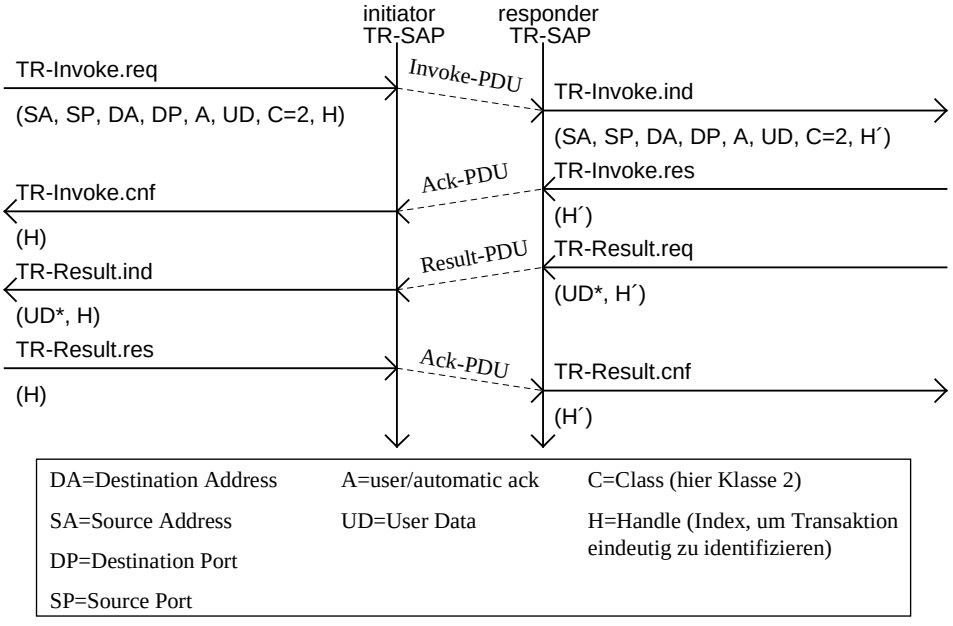

Abbildung 4: Transaktion Klasse 2 mit user acknowledgement

2.5 Sitzungsschicht

Die Sitzungsschicht mit dem Wireless Session Protocol (WSP) bietet dem WAE-Benutzeragenten (WAE=Wireless Application Environment, siehe Abschnitt 2.6) grundsätzlich zwei verschiedene Sitzungsdienste an: Im verbindungsorientierten Modus über das WTP stehen sowohl dem Anfragenden als auch dem Dienstanbieter Request und Response zur Verfügung.

Damit sollte der Anfragende unter normalen Umständen eine Bestätigung seiner Anfrage erhalten. Im verbindungslosen Modus über das WDP erfolgt keine Bestätigung einer Nachricht, weder auf der Seite des Anfragenden noch beim Dienstanbieter. Es handelt sich also um ein so genanntes unzuverlässiges Protokoll. Beiden Möglichkeiten kann, wenn erforderlich, die Sicherungsschicht vorgeschaltet werden. Die Abschnitte 2.5.1 und 2.5.2 beschreiben diese beiden möglichen Modi.

Zuvor wird noch auf einen anderen wesentlichen Aspekt eingegangen, nämlich den Begriff des Zustands. Das zu WSP äquivalente Protokoll im Festnetz-Internet ist das Hypertext Transfer Protocol (HTTP). HTTP ist zustandslos, was einige Probleme mit sich bringt (für Abhilfe sorgen so genannte Cookies, die aber keine elegante Lösung sind). WSP geht hier einen anderen Weg, damit bei einer eventuellen Unterbrechung des Netzwerks nicht dieselben Seiten wieder und wieder übertragen werden müssen.

WSP bietet einige Merkmale, die benötigt werden, um Inhalte zwischen Clients und Servern auszutauschen:

- Sitzungsverwaltung: WSP kann Sitzungen einleiten, die sowohl vom Client als auch vom Server gestartet werden können und die langlebig sind. Sitzungen können auch in einer ordentlichen Art und Weise beendet werden. Des Weiteren von Interesse ist die Möglichkeit einer Aussetzung und späteren Wiederaufnahme einer Sitzung. Die Dauer einer Sitzung ist unabhängig von der Dauer der Transportverbindung (im Falle von WTP) oder der kontinuierlichen Verbindung auf dem Träger.
- Parameterabwicklung: Client und Server können sich beim Aufbau einer Sitzung auf ein gemeinsames Maß an Protokollfunktionalität einigen. Beispiele für Parameter, die dabei vereinbart werden, sind die maximale SDU-Größe (SDU=Service Data Unit) des Client und des Servers sowie die maximale Anzahl ausstehender Anfragen und andere Protokollparameter.
- Darstellung des Inhalts: WSP definiert außerdem die effiziente Binärverschlüsselung des Inhalts, den es überträgt. WSP bietet Inhaltstypisierung und die Zusammensetzung von Objekten an.

Während WSP sehr allgemein gehalten ist, hat das WAP-Forum noch WSP/B (Wireless Session Protocol/Browsing) spezifiziert, das Dienste und Protokolle umfasst, die an Browsing-Anwendungen angepasst sind. Dazu gehören folgende Merkmale:

- HTTP/1.1 Funktionalität: WSP/B unterstützt die Funktionen, die auch HTTP/1.1 unterstützt.
- Austausch von Session Headers: Client und Server können Request/Reply Headers austauschen, die während der Dauer einer Sitzung konstant bleiben. Diese Header können content type, Sprache, Fähigkeiten des Endgeräts, etc. beinhalten.
- Push- und Pull-Datentransfer: Neben dem Pull-Datentransfer, dem traditionellen Mechanismus des Web, werden von WSP/B drei verschieden Push-Mechanismen unterstützt: ein bestätigter Push-Datentransfer bei einer bestehenden Sitzung, ein unbestätigter Push-Datentransfer bei einer bestehenden Sitzung und ein unbestätigter Push-Datentransfer, bei dem keine Sitzung bestehen muss.
- Asynchrone Anfragen: Optional unterstützt WSP/B einen Client, der mehrere Anfragen simultan an einen Server schicken kann. Dabei wird die Anfrageeffizienz gesteigert und Antworten können nun zu wenigen Nachrichten zusammenschmelzen. Des Weiteren wird die Verzögerung verringert, da jede Antwort zum Client gesendet werden kann, sobald sie zur Verfügung steht.

2.5.1 WSP/B über WTP

Wird WSP/B auf WTP aufgesetzt, so kann es alle drei Dienstklassen von WTP, die in dem Abschnitt 2.4 beschrieben wurden, benutzen. Somit entsteht eine große Zahl an Szenarien. In diesem Abschnitt wird jedoch nur exemplarisch auf einige eingegangen.

Wird eine Sitzung aufgebaut, so kann dies beispielsweise mit WTP Klasse 2 geschehen, also mit einem zuverlässigen Dienst (vgl. Abbildung 5). Dabei führt der Client ein S-Connect.req aus und übergibt die Parameter Server-Adresse, Client-Adresse, den optionalen Parameter Client Header und Requested Capabilities, womit Details für die Sitzung ausgehandelt werden. Die Connect-PDU wird übertragen und auf der Server-Seite wird das Dienstprimitiv S-Connect.ind mit den gleichen Parametern wie bei S-Connect.req ausgeführt. Akzeptiert der Server die neue Sitzung, antwortet er mit einem S-Connect.res und übergibt dabei die Parameter Server Header und Negotiated Capabilities, der die ausgehandelten Details beinhaltet. Die ConnReply-PDU wird übertragen, und auf der Client-Seite wird S-Connect.cnf mit den vom Server übertragenen Parametern Server Header und Negotiated Capabilities ausgeführt.

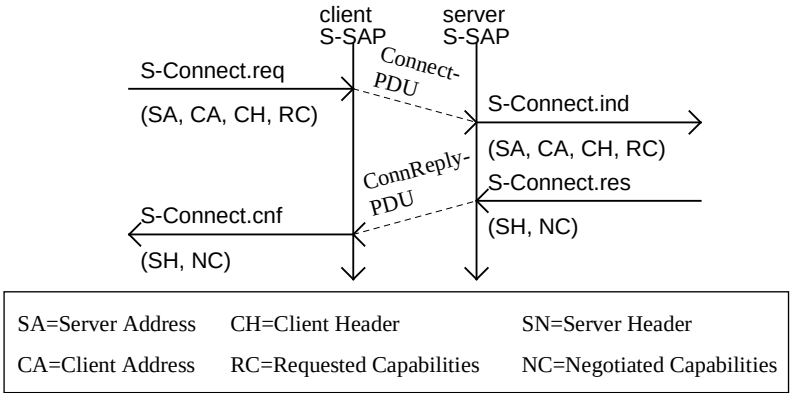

Abbildung 5: WSP/B über WTP-Sitzungsaufbau

Für den Fall, dass ein mobiler Client für kurze Zeit nicht erreichbar ist (z.B. wenn er von einem Netzwerk in ein anderes gelangt oder einfach nur wenn er mit dem Fahrzeug in einen Tunnel fährt), bietet WSP/B über WTP die Dienste Suspend und Resume. So kann beispielweise der Dienst Suspend mit der Klasse 0 realisiert werden (vgl. Abbildung 6): Der Client führt ein S-Suspend.req aus, die Suspend-PDU wird zum Server übertragen und sowohl beim Client als auch beim Server wird das Dienstprimitiv S-Suspend.ind mit dem Parameter Reason ausgeführt.

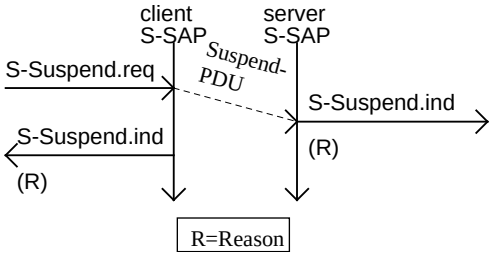

Abbildung 6: WSP/B über WTP-Aussetzung einer Verbindung

Der Dienst Resume kann mit WTP Klasse 2 realisiert werden (vgl. Abbildung 7): Der Client führt S-Resume.req mit den Parametern Server-Adresse und Client-Adresse aus, die Resume-PDU wird zum Server übermittelt und auf der Server-Seite wird S-Resume.ind mit den Parametern Server-Adresse und Client-Adresse ausgeführt. Nun führt der Server das Dienstprimi-

tiv S-Resume.res aus, eine Reply-PDU wird zum Client übertragen und auf der Client-Seite wird S-Resume.cnf ausgeführt – die Verbindung steht wieder.

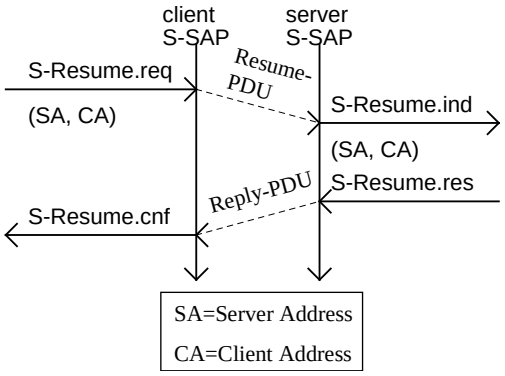

Abbildung 7: WSP/B über WTP-Wiederaufnahme einer Verbindung

Ein etwas umfangreicheres Beispiel für den Einsatz von WTP Klasse 2 ist das folgende Szenario: Um serverseitig eine Operation ausführen zu können, führt der Client das Dienstprimitiv S-MethodInvoke.req aus. Daraufhin wird eine Method-PDU zum Server gesendet. Diese Method-PDU kann entweder eine Get-PDU oder eine Post-PDU, wie sie aus HTTP/1.1 bekannt sind, sein. Beim Server wird nun S-MethodInvoke.ind ausgeführt. Der Server führt S-MethodInvoke.res aus woraufhin der Client-SAP S-MethodInvoke.cnf ausführt. Hat der Server die angeforderte Operation ausgeführt, löst er S-MethodResult.req aus. Eine Reply-PDU wird zum Client übertragen und dort S-MethodResult.ind ausgeführt. Der Client antwortet mit S-MethodResult.res. Abschließend führt der S-SAP des Servers S-MethodResult.cnf aus. Um bei mehreren solchen Anfragen des Clients und einer Reihenfolgevertauschung auf der Server-Seite (z.B. durch eine hohe Zugriffszeit auf eine Disk bei Anfrage k) trotzdem noch zu wissen, welche Antwort zu welcher Anfrage gehört, gibt es die Dienstprimitive S-MethodInvoke_i.req, S-MethodInvoke_i.ind, S-MethodInvoke_i.res, S-MethodInvoke_i.cnf sowie S-MethodResult_i.req, S-MethodResult_i.ind, S-MethodResult_i.res, S-MethodResult_i.cnf.

Die letzten zwei Beispiele, die hier beschrieben werden, handeln nicht von gewöhnlichen Pull-Diensten, bei denen sich der Client Daten vom Server besorgt, sondern von Push-Diensten. Hier schickt der Server ohne Anfrage des Clients diesem Daten.

Zum einen kann ein unzuverlässiger Dienst mittels WTP Klasse 0 in Anspruch genommen werden. Dabei führt der Server S-Push.req mit den Parametern Push Header und Push Body aus und überträgt anschließend eine Push-PDU zum Client. Dort wird nun S-Push.ind mit den Parametern Push Header und Push Body ausgeführt.

Zum anderen kann ein zuverlässiger Dienst mittels WTP Klasse 1 in Anspruch genommen werden (vgl. Abbildung 8). Dabei führt der Server das Dienstprimitiv S-ConfirmedPush.req mit den Parametern Server Push Identifier, Push Header und Push Body aus. Nun wird eine ConfPush-PDU zum Client übertragen. Dort wird S-ConfirmedPush.ind mit den Parametern Client Push Identifier, Push Header und Push Body ausgeführt. Der Client antwortet mit S-ConfirmedPush.res und dem Parameter Client Push Identifier. Als letztes führt nun der Server S-ConfirmedPush.cnf aus, womit die Daten zum Client übertragen wurden.

2.5.2 WSP/B über WDP

Von Zeit zu Zeit ist der Overhead, der beim Verbindungsaufbau und Verbindungsabbau sowie bei Transaktionsaufrufen entsteht, zu groß. Für diese Fälle bietet WSP einen Dienst, der auf

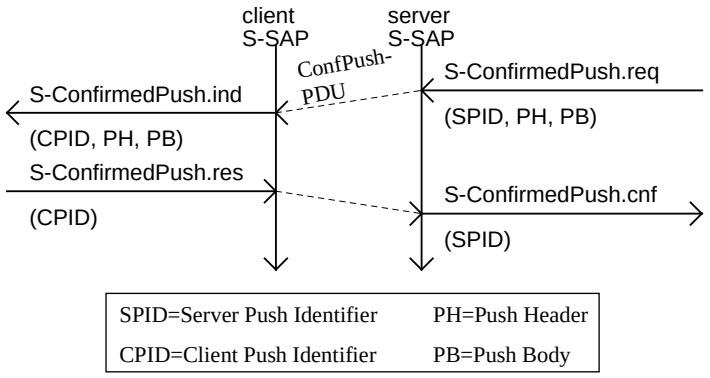

Abbildung 8: WSP/B über WTP-bestätigter Push

dem verbindungslosen, unzuverlässigen WDP-Dienst aufbaut. Eine Beispielanwendung für diesen Dienst, sind mittels Push periodisch zum Client gesendete Wetter-Daten.

Es stehen drei Dienstprimitive für eine verbindungslose Sitzung zur Verfügung: S-Unit-MethodInvoke.req, um eine Operation auf dem Server auszuführen, S-Unit-MethodResult.req, um das Ergebnis zum Client zu übertragen, und S-Unit-Push.req, um Daten im Push-Dienst zum Client zu senden.

Ein Beispiel sieht dann wie folgt aus (vgl. Abbildung 9): Der Client führt S-Unit-MethodInvoke.req mit den Parametern Server-Address, Client-Address, Transaction Identifier, Method und Requested URI aus. Daraufhin wird eine Method-PDU zum Server übertragen. Auf der Server-Seite wird dies mittels S-Unit-MethodInvoke.ind und den Parametern Server-Address, Client-Address, Transaction Identifier, einer einzigartigen Kennung für eine Transaktion, Method, für die auszuführenden Methode, und Requested URI angezeigt. Sobald das Ergebnis beim Server berechnet wurde, wird beim Server S-Unit-MethodResult.req mit den Parametern Client-Address, Server-Address, Transaction Identifier, Status, einer Statusmeldung, Response Header und Response Body ausgeführt. Nun wird eine Reply-PDU zum Client gesendet, der den Empfang der PDU mit S-Unit-MethodResult.ind mit den Parametern Client-Address, Server-Address, Transaction Identifier, Status, Response Header und Response Body anzeigt. Der Server kann aber auch mittels S-Unit-Push.req und den Parametern Client-Address, Server-Address, Push Identifier, Push Header und Push Body eine Push-PDU zum Client übertragen, der seinerseits den Empfang mit S-Unit-Push.ind und den Parametern Client-Address, Server-Address, Push Identifier, Push Header und Push Body anzeigt.

2.6 Anwendungsschicht

Im Folgenden wird kurz auf die Anwendungsschicht mit der Wireless Application Environment (WAE) eingegangen. Weitere Betrachtungen werden im Abschnitt 3 durchgeführt, wo die Datenübertragung vom und zum mobilen Endgerät unter die Lupe genommen wird.

Die Wireless Application Environment spezifiziert einen Rahmen für drahtlose Endgeräte, wie z.B. Mobiltelephone, Pager und PDAs. Seine Aufgabe ist die Bereitstellung von Mitteln zur Entwicklung von Anwendungen und Diensten, die über alle Datenstandards der mobilen Kommunikation hinweg portierbar sind. Anders gesagt: WAE ist also die Umgebung, innerhalb der die Applikationen auf mobilen Endgeräten ablaufen sollen.

WAE erweitert und steigert andere WAP-Schichten, wie das Wireless Transaction Protocol, das Wireless Session Protocol, aber auch andere Internet-Technologien, wie XML, URLs und Scripten. Im Wesentlichen besteht WAE aus folgenden Komponenten:

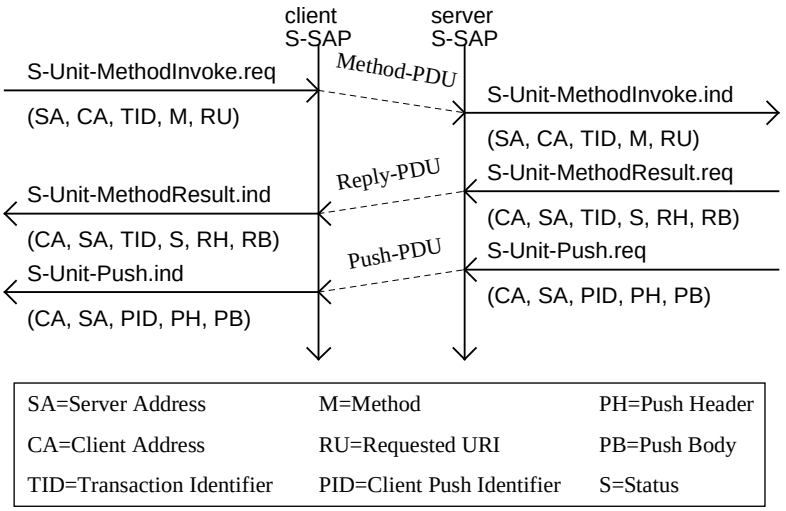

Abbildung 9: WSP/B über WDP

- WML – Wireless Markup Language: WML bietet die Möglichkeit, Texte und Grafiken darzustellen, und basiert auf XML. (Siehe auch im Abschnitt 2.6.1).
- WMLScript: WMLScript ist eine Scriptsprache, die ähnlich aufgebaut ist, wie JavaScript. der WMLScript-Interpreter interpretiert Bytecode und ist direkt auf Endgeräten ausführbar. WMLScript basiert auf ECMAScript. (Siehe auch Abschnitt 2.6.2).
- WTA – Wireless Telephony Application: WTA dient dem Zugriff auf erweiterte Telefondienste. Hierüber wird ein gesicherter Zugriff auf die Telephony API des Endgeräts bereitgestellt, der beispielsweise eine differenzierte Kostenkontrolle ermöglicht.

Ähnlich wie im Client-Server-Modell des WWW gibt es auch in der WAP-Technologie einen auf der Client-Seite ablaufenden Benutzeragenten (User Agent). WAE beinhaltet Benutzeragenten für die zwei primären Standardinhalte – kodierte Wireless Markup Language (WML) und kompiliertes Wireless Markup Language Script (WMLScript) – sowie einen WTA-Benutzeragenten, auf den allerdings im Folgenden nicht weiter eingegangen wird.

2.6.1 WML – Wireless Markup Language

WML ist eine tag-basierte Sprache. Sie als ein XML Dokumenttyp spezifiziert und an HTML und HDML (Handheld Device Markup Language) angelehnt. Während der Entwicklung von WML mussten verschiedene Schwierigkeiten drahtloser tragbarer Geräte berücksichtigt werden. Zunächst einmal hat eine drahtlose Verbindung im Vergleich mit einer festen Verbindung nur eine sehr geringe Kapazität. Des Weiteren haben derzeitige tragbare Geräte kleine Displays, begrenzte Speicherkapazität und nur geringe Prozessorleistung.

Eine WML-Seite nennt man Deck. Es besteht aus verschieden Cards. Ein Deck ähnelt einer HTML-Seite im WWW und wird auch durch eine URL identifiziert. Es ist die Einheit bei der Übertragung von Inhalten. Der Benutzer navigiert mit Hilfe seines Browsers durch eine Anzahl von Cards. Dabei kann er Informationen anschauen, erfragte Daten eingeben und in einem Auswahlmenü eine Auswahl treffen. Erwähnenswert ist, dass WML nicht spezifiziert, wie die Implementation eines Browsers mit dem Benutzer interagieren soll, sondern lediglich die Absicht der Interaktion in einer abstrakten Art und Weise beschreibt. Der Benutzeragent des tragbaren Gerätes muss entscheiden, wie alle Elemente einer Card bestmöglich dargestellt werden. Die Darstellung ist sehr von den Kapazitäten des Endgerätes abhängig.

WML beinhaltet verschiedene Grundelemente:

- Texte und Grafiken: WML gibt Hinweise, wie Texte und Grafiken dem Benutzer dargestellt werden können. Lediglich eine Menge von Markup-Elementen, wie Fettdruck, Italic, etc., werden bereitgestellt.
- Benutzer-Interaktion: WML unterstützt verschiedene Elemente für Eingaben durch den Benutzer, wie z.B. Eingabekontrollen für Texte oder Passwörter oder Auswahlmöglichkeiten. Dies könnte durch richtige Tasten, Softkeys oder durch Spracheingabe geschehen.
- Navigation: Wie auch HTML bietet WML einen History-Mechanismus, der es erlaubt, durch die bisher besuchten Seiten zu navigieren. Außerdem bietet WML Navigation durch Hyperlinks und andere innerhalb einer Card definierte Navigationselemente.
- Inhalts-Management: WML erlaubt es, Zustandsinformationen zwischen dem Auftreten verschiedener Decks ohne Serverinteraktion zu speichern. So kann z.B. der Zustand einer Variablen länger benötigt werden als ein einzelnes Deck, und somit kann Zustandsinformation durch verschiedene Decks hindurch bewahrt werden.

Abschließend soll noch ein kleines, aber sehr verständliches Beispiel eines WML-Dokuments gegeben werden.

```
<WML>
    <CARD>
        <DO TYPE=„ACCEPT“>
            <GO URL=„#card_two“/>
        </DO>
        This is a simple first card!
        On the next you can choose ...
    </CARD>
    <CARD NAME=„card_two“>
        ... your favorite pizza:
        <SELECT KEY=„PIZZA“>
            <OPTION VALUE=„Mar“>Margherita</OPTION>
            <OPTION VALUE=„Fun“>Funghi</OPTION>
            <OPTION VALUE=„Vul“>Vulcano</OPTION>
        </SELECT>
    </CARD>
</WML>
```

Zuerst wird die erste Card definiert. Sie „zeigt“ einen Text „an“, nachdem sie geladen wird (anzeigen kann z.B. auch Sprachausgabe sein). Sobald der Benutzer das DO-Element aktiviert (z.B. per Knopfdruck oder per Spracheingabe) wird die zweite Card „angezeigt“. Auf dieser kann der Benutzer zwischen drei Pizzen wählen. Je nach Auswahl des Benutzers hat PIZZA dann den Wert Mar, Fun oder Vul, was der Anbieter weiter verarbeiten kann.

WML kann durch eine kompakte Binärrepräsentation kodiert werden, um Bandbreite auf der drahtlosen Verbindung zu sparen. Diese kompakte Repräsentation basiert auf dem binären XML Content Format, wie es im WAP Forum spezifiziert wurde. Die binäre Kodierung in WML ist nur eine spezielle Version dieses Formats (die kompakte Repräsentation ist für XML-Inhalte allgemein gültig). Dieses kompakte Format erlaubt Übertragungen ohne Funktionsverluste oder den Verlust semantischer Informationen. Zum Beispiel wird das URL-Präfix „http://“, das allen URLs gemein ist, durch hexadezimal 4B kodiert. Das Schlüsselwort „CARD“ wird durch E9 kodiert, „SELECT“ wird durch 37 und „OPTION“ durch 35 kodiert. Dieser einfache Bytecode ist viel effizienter als der Klartext, der in HTML benutzt wird.

2.6.2 WMLScript

WMLScript dient als Ergänzung zu WML und bietet allgemeine Scripting-Fähigkeiten innerhalb der WAP-Architektur. Während WML-Inhalte statisch sind, bietet WMLScript die Möglichkeit, dynamisch Inhalte zu produzieren. WMLScript basiert auf ECMAScript und ist damit ähnlich aufgebaut wie JavaScript. WMLScript interpretiert Bytecode und ist direkt auf den Endgeräten ausführbar. Es wurde an die drahtlose Umgebung angepasst, was einen geringen Speicherbedarf und eine effiziente Übertragung durch das Medium Luft über einen platzsparenden Bytecode beinhaltet. Zum Generieren des Bytecodes wird ein WMLScript-Compiler eingesetzt. Diese Vorgehensweise spart Zeit und Speicherressourcen. Hier einige Dienste, die durch WMLScript ermöglicht werden:

- Validierung der Benutzereingaben: Bevor die eingegebenen Daten des Benutzers zum Server übertragen werden, kann sie WMLScript auf Gültigkeit prüfen und damit im Fehlerfall Bandbreite und Verzögerung sparen. Andernfalls müsste der Server alles überprüfen, was im Fehlerfall zu mindestens einer weiteren Umlaufzeit führt.
- Zugriff zu Geräteeinheiten: WMLScript bietet Funktionen, um Hardware-Komponenten und Software-Funktionen eines Gerätes anzusprechen. Auf einem Telefon kann ein Benutzer beispielsweise einen Anruf tätigen, auf das Adressbuch zugreifen oder eine Nachricht mit Hilfe des Nachrichtendienstes des Handys verschicken.
- Lokale Benutzerinteraktion: Ohne zu Verzögerungen zu führen, kann WMLScript direkt und lokal mit dem Benutzer interagieren, ihm Nachrichten anzeigen oder ihn zu Eingaben auffordern. So kann z.B. nur das Ergebnis einiger Interaktionen zum Server übertragen werden.
- Erweiterungen der Geräte-Software: Mit Hilfe von WMLScript kann ein Gerät neu konfiguriert werden und es kann ihm neue Funktionalität zugegeben werden. Der Benutzer kann neue Software vom Händler herunterladen und somit sein Gerät leicht aufwerten.

Des Weiteren ist zu sagen, dass WMLScript ereignisgesteuert ist, d.h. ein Script kann bei bestimmten Benutzerereignissen oder Ereignissen in der Umgebung zur Antwort einbezogen werden. Daneben hat WMLScript vollen Zugriff auf das Zustandsmodell von WML: Es kann Variablen setzen und lesen. Zuletzt sei noch erwähnt, dass WMLScript viele Fähigkeiten von Standardprogrammiersprachen unterstützt, wie z.B. Funktionsaufrufe, Ausdrücke, if, while, for, return, etc. Hier nun ein kleines Beispiel eines WMLScript-Programms:

```
function pizza_test(pizza_type) {
        var taste = „unknown“;
        if (pizza_type = „Mar“) {
                taste=„well...“;
        }
        else {
                if (pizza_type = „Vul“) {
                        taste = „quite hot“;
                };
        };
        return taste;
};
```

Der WMLScript-Compiler kann eine oder mehrere solcher Scripten in eine WMLScript Übersetzungseinheit kompilieren. Ein tragbares drahtloses Endgerät kann nun eine solche Übersetzungseinheit durch den Einsatz von Standardprotokollen mit HTTP-Semantik, wie z.B. WSP,

vom Server holen. Innerhalb einer Übersetzungseinheit kann der Benutzer eine bestimmte Funktion aufrufen, indem er Standard-URLs mit Fragment-Anker benutzt. Ein Fragment-Anker ist durch eine URL, „#“ und ein Fragment-Bezeichner spezifiziert. Wenn die URL der Übersetzungseinheit des Beispiels http://www.xyz.int/myscr wäre, könnte der Benutzer das Script mit dem Parameter „Vul“ folgendermaßen aufrufen: http://www.xyz.int/myscr# pizzatest('Vul').

Das WAP-Forum hat verschiedene Standardbibliotheken für WMLScript spezifiziert. Darunter sind Bibliotheken über Gleitkommaarithmetik, Zeichenkettenverarbeitung, URL und WML-Browser.

3 Datenübertragung

In diesem Abschnitt wird etwas näher auf die Datenübertragung vom Server zum mobilen Client eingegangen. Dabei wird in Abschnitt 3.1 auf den Kommunikationsweg eingegangen, ehe dann in den Abschnitten 3.2 und 3.3 die Diensten-Arten Pull und Push folgen.

3.1 Gateways

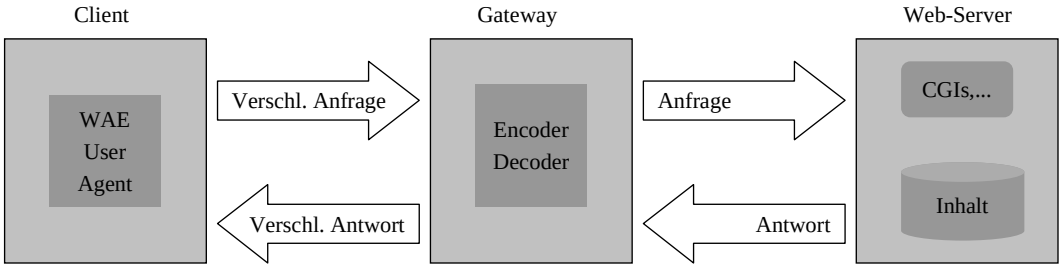

Abbildung 10: Gateway als Mittler zwischen Client und Server

Abbildung 10 zeigt den normalen Kommunikationsweg eines drahtlosen Endgeräts mit einem WWW-Server (siehe auch [Hitz99] und [WAP99]). Auf Basis des Wireless Session Protocols (WSP) wird eine Sitzung zwischen dem Endgerät und dem WAP-Gateway aufgebaut. Dieses Gateway übersetzt die endgerätespezifische Anfrage in einen Standard-HTTP-Request und schickt diesen an den WWW-Server weiter. Außerdem kann das Gateway binäres WML des Endgerätes in Klartext-WML und WML in HTML übersetzen.

Auf der Server-Seite kann der Anbieter der Webseite zum einen traditionelle HTML-Seiten für den Standardbesucher bereitstellen, die vom Gateway wieder in WML und binäres WML übersetzt werden. Zum anderen kann er aber auch spezielle Seiten bereitstellen, die in der Wireless Markup Language (WML) für mobile Endgeräte erstellt wurden. Diese Seiten werden dann vom Gateway in binäres WML übersetzt und zum mobilen Client gesendet. Dabei ist ein Gateway keinesfalls von Nöten; dieselben Funktionen kann auch ein separater Teil des Servers übernehmen.

Wie bereits in Abschnitt 2.6.1 erwähnt, basiert WML auf XML und ist auf kleine Bildschirme von Handys oder PDAs zugeschnitten. Hinzu kommt noch, dass die im Vergleich zu HTML geringere Anzahl an Tags auf die Telekommunikation zugeschnitten ist und dass der Zugriff auf HTML-Seiten über HTTP 1.1 erfolgt, was eine Nutzung der bestehenden Internet-Struktur für mobile Endgeräte gewährleistet. Alles in allem ist es somit möglich, normale WWW-Seiten bzw. speziell angefertigte WML-Seiten auf WAP-fähigen Endgeräten zu betrachten und somit einen mobilen Zugang zum WWW zu erlangen.

Man kann bei der Art und Weise, wie die Daten übertragen werden, zwischen zwei verschieden Modi unterscheiden. Da ist zum einen der in dem Abschnitt 3.2 noch genauer erläuterte Pull-Dienst, bei dem der Client traditionell die Daten beim Server anfragt und dann zugesendet bekommt. Zum anderen gibt es aber auch den Fall, dass der Server ohne eine Anfrage Daten zu einem oder mehreren Clients sendet. Man spricht dann von dem so genannten Push-Dienst (vgl. Abschnitt 3.3).

3.2 Pull-Dienst

Wie schon gesagt ist der Pull-Dienst der klassische Fall des Datentransfers, wie er auch im großen Bruder des WAP – dem WWW – vorkommt. Bei diesem Dienst baut der Client mittels WSP eine Verbindung zum Server auf. Dabei benutzt er eventuell ein Gateway (vergleiche Abschnitt 3.1). Die von ihm bzw. vom Gateway gesendete Anfrage wird nun auf der Server-Seite bearbeitet. Wird eine Seite angefragt, sendet der Server eine HTML- bzw. eine WML-Seite. Im Falle einer HTML-Seite übersetzt sie das Gateway wieder in WML und sendet dem Client die Daten. Der Server kann auch unter Zuhilfenahme von JavaScript bzw. WMLScript Anfragen an den Client übertragen. Auch diese werden im Gateway durch einen WMLScript Compiler kompiliert und als Bytecode, der für geringe Bandbreiten und mobile Clients mit eingeschränkter Kapazität entwickelt wurde, dem Client zugesandt. Auf dem Client kann nun der Bytecode ausgeführt werden.

3.3 Push-Dienst

Der Push-Dienst ist eine Entwicklung jüngerer Zeit. Dabei wird der Server in die Lage versetzt, dem Client ohne Anfrage Daten zu senden. Mögliche Anwendungsgebiete wären Wetterdaten, Börsendaten oder anderes. Vorraussetzung ist natürlich ein Einverständnis des Clients.

Ein Beispiel für einen solchen „Verbindungsaufbau“ habe ich bereits in Abschnitt 2.5.1, Abbildung 8 gegeben. Hier wurde einmal ein unzuverlässiger Datenaustausch mit WTP Klasse 0 aufgezeigt. Dabei wurde eine Push-PDU mittels dem Dienstprimitiv S-Push.req (Push Header, Push Body) vom Server zum Client übertragen, dessen SAP dann ein S-Push.ind (Push Header, Push Body) ausführte. Zum anderen wurde ein zuverlässiger Dienst mit WTP Klasse 1 aufgezeigt. Hierbei wurde nach dem Aufruf von S-ConfirmedPush.req (Server Push Identifier, Push Header, Push Body) durch den Dienstzugangspunkt des Servers eine ConfirmedPush-PDU zum Client übertragen, der nach dem Ausführen von S-ConfirmedPush.ind (Client Push Identifier, Push Header, Push Body) noch das Dienstprimitiv S-ConfirmedPush.res (Client Push Identifier) ausführte. Der Dienstzugangspunkt des Servers quittierte dies mit einem S-ConfirmedPush.cnf (Server Push Identifier).

Dieser Dienst bringt vor allem Vorteile bei Daten, die zu einer bestimmten Zeit sehr oft abgefragt werden. So wäre zum Beispiel ein Ergebnis-Ticker vorstellbar, der bei Sportereignissen, wie die Fußball-WM oder Fußball-EM, abends die Ergebnisse der Spiele zu interessierten WAP-Benutzern überträgt.

4 WAP Version 1.2

In diesem letzten Abschnitt wird noch auf die Version 1.2 und deren Neuerungen eingegangen und ein Blick in die Zukunft von WAP geworfen (siehe auch [Röwe00]).

Die bisher neueste Version, die das WAP-Forum standardisiert hat, ist die Version 1.2, die im Oktober 1999 verabschiedet wurde. Wie bereits in Abschnitt 1 erwähnt, wurde diesmal

– anders als bei dem Übergang von Version 1.0 auf 1.1 – auf Abwärtskompatibilität Wert gelegt. Die Version 1.2 stellt lediglich eine Erweiterung der Version 1.1 dar.

Dabei wurde neben etlichen kleinen Erweiterungen und Änderungen, die sich durch nahezu alle Bereiche von WAP ziehen, die WAP-Push-Architecture in den Vordergrund gestellt (vergleiche hierzu auch Abschnitte 2.5.1 und 3.3). Sie ermöglicht „Real Time Alerts“ von einem Server aus auf den mobilen Client zu initiieren. So können Änderungen am Flugplan an interessierte Clients gesendet, oder Börsenticker via WAP realisiert werden. Weitere wesentliche Neuerungen finden sich in der Erweiterung der Wireless Telephony Application (WTA) und der Unterstützung zusätzlicher Träger.

Das Ziel der WAP-Push-Architecture ist es, Inhalte verschiedenster Art vom Server zum mobilen Client zu „schieben“. Schematisch dargestellt erfolgt eine Push-Operation im WAP-Umfeld genau dann, wenn ein so genannter Push-Initiator Inhalte an einen Client via Push Over-The-Air Protocol (PushOTA) und/oder Push-Access Protocol sendet. Da sich der Push-Initiator aber in der Regel im Internet befindet und somit kein gemeinsames Protokoll mit dem WAP-Client teilt, wird ein zusätzliches Gateway zur Übersetzung benötigt. An dieser Stelle kommt das Push Proxy Gateway ins Spiel (Abbildung 11). Möchte ein Push-Initiator Inhalte an einen WAP-Client übermitteln, so muss er zunächst das Push Proxy Gateway (PPG) mit Hilfe eines Internet-Protokolls ansprechen. Das Push Proxy Gateway wiederum übernimmt anschließend alle notwendigen Schritte, um die Inhalte an den gewünschten Client in der WAP-Domäne zu übertragen. Das internetseitige Zugriffsprotokoll des PPG wird als Push Access Protocol bezeichnet. Das WAP-seitige Protokoll dagegen nennt sich Push Over-The-Air Protocol. Während das Push Access Protocol XML-Nachrichten nutzt, basiert das PushOTA Protocol auf den Diensten des WSP. Natürlich berücksichtigt die Spezifikation des Push Framework auch verschiedene Mechanismen zur Authentifizierung des Push-Initiators.

Neben den Neuerungen durch die WAP-Push-Architecture wurden auch die Zeichensätze genauer unter die Lupe genommen. So drückt sich die Spezifikationen 1.2 zum Thema „International Support“ etwas genauer aus als ihr Vorgänger.

Eine weitere Neuerung im Bereich WML/WMLScript stellt die Spezifikation der WMLScript Crypto Library dar, die Kryptographie auf Seiten des WAP-Clients ermöglicht. Darunter fällt auch ein so genanntes „Signed Content Format“, das beim Transport von verschlüsselten Informationen von bzw. zu WAP-fähigen Geräten verwendet werden kann. Derzeit beschränken sich die Funktionen der WMLScript Crypto Library lediglich auf den Bereich digitaler Signaturen. Es ist aber für zukünftige Versionen angedacht, weitere Quasistandards der Verschlüsselung zu integrieren.

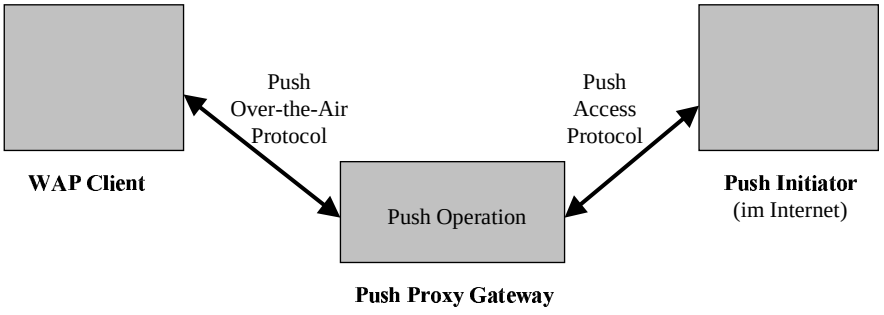

Abbildung 11: Push Proxy Gateway

5 Zusammenfassung und Ausblick

Zusammenfassend kann man sagen, dass es durch die verschiedenen, oben beschriebenen Techniken möglich ist, mit einem mobilen Endgerät ins Internet zu gelangen. Zwar gibt es noch kein großes Angebot an WAP-Seiten, doch wird sich dies in Zukunft sicherlich ändern. Auch wird die Hardware noch weiter verbessert werden: bessere Displays, Sprachausgabe, Spracheingabemöglichkeiten und schnellere Übertragungsleistungen sind nur einige Beispiele hierfür. Betrachtet man die Neuerungen innerhalb der WAP-Spezifikation 1.2 im Vergleich zu ihrer Vorgängerversion, zeigen sich vor allem zwei Dinge. Erstens ist noch lange kein endgültiger Stand erreicht, und zweitens scheint die enge Zusammenarbeit des WAP-Forums mit den führenden Unternehmen der drahtlosen Telekommunikation tatsächlich dazu zu führen, dass sich Neuerungen nicht an der Theorie, sondern an der Praxis orientieren. Besonders hoch anzurechnen ist dabei die Tatsache, dass das WAP-Forum den Versionsstand 1.1 mittlerweile als eine Art fixe Basis ansieht und bei jeglichen Neuerungen streng auf eine gegebene Abwärtskompatibilität achtet. Ein Sprung wie von Version 1.0 auf 1.1 soll in Zukunft auf jeden Fall vermieden werden, was die Hersteller WAP-fähiger Geräte danken werden.

Ebenfalls interessant dürften die Ergebnisse der erst vor kurzem beschlossenen Zusammenarbeit zwischen WAP-Forum und W3C sein. In Hinblick auf das Durchsetzen von Standards war dies sicher nicht die schlechteste Entscheidung des WAP-Forums. So wird im Augenblick darüber diskutiert, ob WML in XHTML integriert werden soll, so dass dieses erweiterbare HTML auch mobile Geräte bedienen könnte.

Man darf auf jeden Fall gespannt sein, was sich in der nächsten Zeit noch so an Neuerungen im WAP-Umfeld ergibt. Geplant sind zum Beispiel SimCard- und Billing-Interfaces. Darüber hinaus gibt es bereits heute, kurz nach Erscheinen der WAP-Spezifikation 1.2, eine neue Spezifikationen – WAP 1.3 ante portas.

Literatur

[Hitz99] Andreas Hitzig. Drahtlos surfen mit Volldampf. *iX*, Oktober 1999, S. 128–132.

[Röwe00] Lars Röwekamp. Drahtlos die Dritte – Wireless Application Protocol 1.2 verabschiedet. *iX*, Mai 2000, S. 104–107.

[Schi00] J. Schiller. *Mobile Communications*, Kapitel 11: Support for Mobility, S. 309–363. Addison-Wesley. 2000.

[WAP99] WAP Forum. *Wireless Application Environment Overview*, November 1999. WAP-152.

Automatische Netzwerkkonfiguration

Anne Fröhling

Kurzfassung

Es gibt verschiedene Protokolle, die zur automatischen Netzwerkkonfiguration verwendet werden können. Das Reverse Address Resolution Protocol (RARP) kann einem Endsystem automatisch eine vorher festgelegte IP-Adresse zuordnen. Um noch weitere Konfigurationsparameter zuzuweisen, müssen Protokolle wie BOOTP oder DHCP verwendet werden. Der große Vorteil von DHCP ist, dass es dynamische Konfiguration erlaubt, das heißt, dass IP-Adressen und die zugehörigen Parameter für eine bestimmte Zeit an einen Rechner verliehen werden können. Autokonfiguration ohne Vorhandensein eines Autokonfigurations-Servers wird derzeit nur in bestimmten Fällen unterstützt, z.B. in Ad-hoc-Netzwerken. Die neue Generation des Internet-Protokolls IPv6 bietet solche Mechanismen für alle Netzwerktypen an. Für Netzwerke ohne Administrator kann allerdings selbst Autokonfiguration oft nicht genutzt werden, hier wäre der Gebrauch von Zeroconf-Protokollen angebracht.

1 Einleitung

Der Einsatz der TCP/IP-Protokollfamilie für die Kommunikation in Computernetzwerken wird als große technische Errungenschaft angesehen. Allerdings sind vor ihrer Nutzung eine ganze Menge verschiedener Konfigurationseinstellungen in einem Netzwerk vorzunehmen, die ein bestimmtes Fachwissen voraussetzen. Da jedoch auch immer mehr "Computerlaien" zu Nutzern des Internets werden, sollte der manuelle Konfigurationsaufwand relativ klein gehalten werden, idealerweise sogar Null sein.

In dieser Seminararbeit wird auf Protokolle für automatische Netzwerkkonfiguration eingegangen. Nachdem in Abschnitt 2 kurz die zu konfigurierenden Parameter erläutert werden, beschreibt Abschnitt 3 die wichtigsten Vertreter dieser Autokonfigurationsprotokolle. Begonnen wird dabei mit dem sehr einfachen Reverse Address Resolution Protocol (RARP), das lediglich eine IP-Adresse zuweisen kann. Danach werden das so genannte Bootstrap-Protokoll (BOOTP) und das Dynamic Host Configuration Protocol (DHCP) beschrieben, wobei DHCP in der Praxis immer mehr an Bedeutung gewinnt, da es dynamische Konfiguration unterstützt. Am Ende des Abschnittes 3 werden die Autokonfigurationsmechanismen des neuen Internet-Protokolls IPv6 näher erklärt. Im anschließenden Abschnitt 4 wird auf die Autokonfigurationsfähigkeit von Ad-hoc-Netzwerken eingegangen.

Zerokonfigurationsprotokolle sind zwar noch eine Zukunftsvision, die IETF beschäftigt sich jedoch bereits mit den Anforderungen an solche Protokolle. Ein kurze Übersicht hierzu wird in Abschnitt 5 gegeben. Abschließend werden im letzten Abschnitt die Umsetzungsmöglichkeiten von Zeroconf-Protokollen diskutiert.

2 Konfiguration eines Internetrechners

Damit ein Rechner im Internet mit anderen Rechnern kommunizieren kann, muss er vorher konfiguriert werden.

Bei TCP/IP besitzt jeder Rechner eine eigene IP- und Hardware-Adresse, die ihn eindeutig identifizieren. Die IP-Adresse ist 32 Bit lang und wird normalerweise in "Punktschreibweise" angegeben ist. Dabei wird jedes Oktett als Dezimalzahl geschrieben und mit einem Punkt vom nächsten getrennt, z.B. 124.13.42.148. IP-Adressen werden von einer zentralen Verwaltungsstelle, der Internet Assigned Number Authority (IANA), vergeben und sind in verschiedene Klassen unterteilt. Bei einer Adresse der Klasse A zum Beispiel ist das erste Bit auf 0 gesetzt, die folgenden 7 Bit geben das jeweilige Netzwerk an und die restlichen 24 das Endsystem. Diese Netzwerke lassen sich wiederum in Subnetze unterteilen, um eine übersichtlichere Struktur zu erhalten, hierfür werden einige der 24 Rechnerbits benutzt, um das Subnetz zu identifizieren.

Will ein Rechner eine Nachricht verschicken, so muss auf der IP-Schicht entschieden werden, ob der Empfänger im eigenen Subnetz liegt oder ob die Nachricht über einen Router in ein anderes Subnetz geschickt werden muss. Um diese Entscheidung treffen zu können, muss für jeden Internetrechner eine Subnetzmaske konfiguriert werden, die angibt, bis zu welchem Bit in der IP-Adresse das Subnetz kodiert ist. Sieht die Subnetzmaske beispielsweise folgendermaßen aus: 255.255.255.0, so sind die ersten 24 Bit für die Identifizierung des Netzes und Subnetzes belegt, die restlichen 8 identifizieren das Endsystem.

Zum Vergleichen zweier Subnetze wird die Subnetzmaske sowohl mit der eigenen als auch mit der Empfängeradresse durch ein UND verknüpft. Sind die Ergebnisse gleich, liegen also beide Rechner im gleichen Subnetz, wird vom sendenden Rechner eine ARP-Anfrage (Address Resolution Protocol) verschickt. ARP wird verwendet, um die zu einer IP-Adresse gehörige Hardware-Adresse zu erfragen. Hierzu wird eine Nachricht an alle Rechner des Subnetzes gesendet (Broadcast). Als Empfängeradresse wird im ARP-Paketrahmen (vergleiche Abb. 2) eine subnetzspezifische IP-Adresse angegeben, die Broadcast-Adresse. Bei dem Rechner mit der Internetadresse 124.13.42.148 und der Subnetzmaske 255.255.255.0 lautet diese zum Beispiel 124.13.42.255. Die Broadcast-Adresse muss konfiguriert sein.

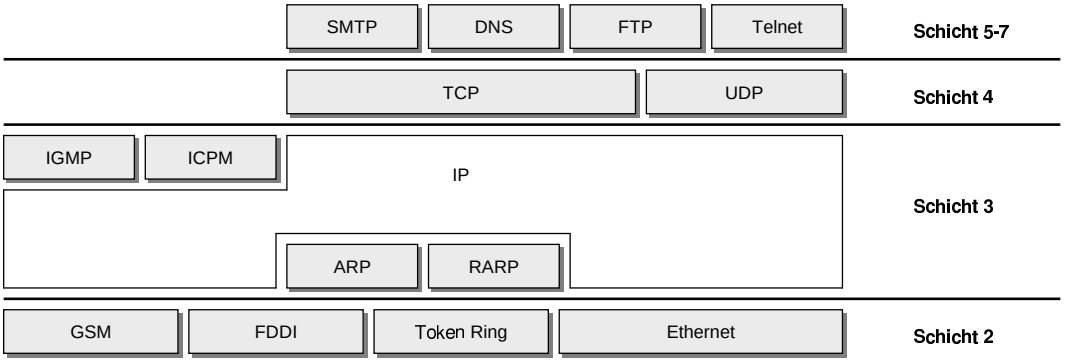

Abbildung 1: Die TCP/IP Protokollfamilie.

Hardware-Adressen sind vom Netzwerktyp abhängig und werden vom jeweiligen Hersteller fest in die Netzwerkkarten eingebrannt. Bei lokalen Netzen (LANs) werden sie auch als MAC-Adressen bezeichnet, da sie der Medium-Access-Control-Schicht (Schicht 2a) angehören. Diese Schicht benutzt die MAC-Adresse zum Versenden der Pakete.

Befinden sich Sender und Empfänger nicht im gleichen Subnetz und sind keine expliziten Routen für den entsprechenden Zielrechner bzw. das Zielnetz konfiguriert worden, so werden

die Pakete zunächst an den Router verschickt, dessen Adresse als Default Gateway angegeben ist. Dieser leitet alle eintreffenden Nachrichten dann an die entsprechenden Netzwerke weiter.

Unter TCP/IP gibt es außerdem eine so genannte logische Adresse, dies ist eine Zeichenkette beliebiger Länge, die einen Rechner weltweit eindeutig beschreibt. Sie wird auch als Rechnername oder Hostname bezeichnet. Ein Hostname ist in zwei Teile gegliedert, die durch einen Punkt voneinander getrennt werden. Der erste Teil ist der eigentliche Rechnername und der zweite Teil der Domänenname, der das Netzwerk angibt.

Durch die Verwendung von logischen Adressen wird die Bedienung von Anwendungsprogrammen wesentlich erleichtert, jedoch muss auf der Schicht 4 (Transportschicht) eine Umwandlung in die IP-Adresse vorgenommen werden. Dies kann auf zwei Arten geschehen, einmal durch Verwendung einer lokalen Tabelle, in der jeder logischen Adresse die entsprechende IP-Adresse zugeordnet wird, oder aber durch das Domain Name System (DNS). DNS ist ein System, das TCP/UDP benutzt, um einen DNS-Server mit der Namensauflösung zu beauftragen. Die IP-Adresse dieses DNS-Servers muss konfiguriert sein.

Abschließend noch einmal eine Auflistung der zu konfigurierenden Parameter:

- IP-Adresse
- Subnetzmaske
- Default Gateway (optional noch weitere Routen)
- Broadcast-Adresse
- IP-Adresse des DNS-Servers und/oder Pfad für die lokale Zuordnungstabelle
- logische Adresse (nicht zwingend)

3 Ansätze zur Autokonfiguration

In diesem Abschnitt werden zur Autokonfiguration verwendbare Protokolle beschrieben. Das Reverse Address Resolution Protocol (RARP) ist dabei das einfachste dieser Protokolle, weist allerdings auch nur die IP-Adresse zu. Das komplexere Bootstrap-Protokoll (BOOTP) und das Dynamic Host Configuration Protocol (DHCP) sind in der Lage, alle benötigten Parameter zu übertragen. DHCP ist eine Erweiterung von BOOTP und bietet zusätzlich die Möglichkeit einer dynamischen Konfiguration. Dies kann auch unter IPv6 angewendet werden, außerdem gibt es bei dem neuen Internet-Protokoll die Möglichkeit der Autokonfiguration ohne vorhandene Autokonfigurations-Server.

3.1 Reverse Address Resolution Protocol (RARP)

Die IP-Adresse eines Rechners ist unabhängig von seiner Hardware-Adresse (zumindest bei IPv4). Router und Endsysteme benutzen auf Schicht 2 die MAC-Adresse für die Übertragung von Datenpaketen, daraus resultiert die Notwendigkeit für das ARP-Protokoll (vergleiche Abschnitt 2). Einige Rechner, z.B. plattenlose Workstations, haben keinen permanenten Speicher für die IP-Adresse. Sie beziehen ein Speicherabbild ihres Betriebssystems von einem entfernten Datei-Server. Damit nicht jede Workstation ein eigenes Speicherabbild braucht, ist die IP-Adresse nicht darin eingebettet.

Will man TCP/IP nutzen, ist eine IP-Adresse jedoch von Nöten. Um diese zu ermitteln, wird das Reverse Address Resolution Protocol (RARP) der TCP/IP-Familie verwendet. Der

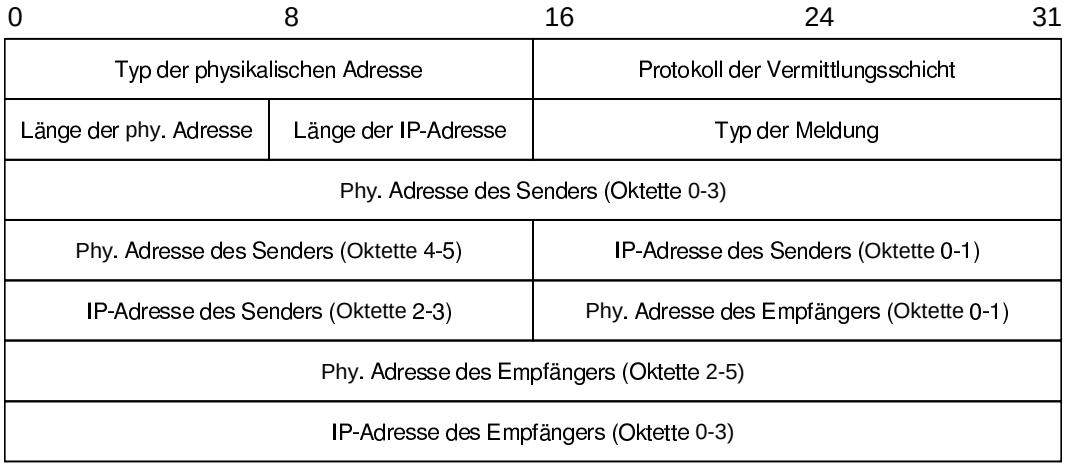

Abbildung 2: Der ARP/RARP-Paketrahmen bei der IP-zu-Ethernet-Adressauflösung.

bootende Rechner verschickt eine Schicht-2-Broadcast-Nachricht, die im Datenteil den RARP-Paketrahmen enthält. Im Kopf der Nachricht wird RARP als Nachrichtentyp durch eine festgelegte Bitfolge identifiziert. Im RARP-Paketrahmen (vergleiche Abb. 2) ist die MAC-Adresse des bootenden Rechners als Sender- und Empfängeradresse angegeben. Die Nachricht wird nur von Rechnern bearbeitet, die authorisiert sind, RARP-Dienste zu leisten. Diese Rechner bezeichnet man auch als RARP-Server. Als Zieladresse verwendet der Rechner die Broadcast-Adresse der Schicht 2. Da Broadcast-Nachrichten nicht von Routern weitergeleitet werden, muss in jedem Subnetz also mindestens ein RARP-Server vorhanden sein, um das Protokoll wirklich anwenden zu können. ARP und RARP verwenden den gleichen Rahmen , daher muss noch spezifiziert werden, dass es sich um eine RARP-Anfrage handelt. Hierzu wird das Feld *Typ der Meldung* auf 3, für eine RARP-Antwort auf 4 und bei ARP entsprechend auf 1 oder 2 gesetzt. Die Server suchen die IP-Adresse in ihren Abbildungstabellen bzw. Konfigurationsdateien und beantworten die Anfrage mit einer RARP-Antwort. Der Sender erhält Antworten von allen im Netz befindlichen RARP-Servern, obwohl nur die zuerst eintreffende verwendet wird. Der Rechner speichert die IP-Adresse ab und muss erst beim erneuten Booten wieder eine RARP-Anfrage starten.

Die Nachteile von RARP liegen auf der Hand: Die RARP-Nachricht kann nicht von Routern weitergeleitet werden und es wird nur die IP-Adresse übertragen. Daher wurden weitere Protokolle entwickelt, z.B. BOOTP.

3.2 Bootstrap Protocol (BOOTP)

Das Bootstrap-Protokoll (BOOTP) überträgt neben der IP-Adresse auch die Adresse des zuständigen Routers und des Nameservers, sowie die Subnetzmaske. Es benutzt dabei UDP/IP, und wie bei RARP ist auch hier nur ein Paketaustausch nötig.

Um IP zu benutzen braucht ein Rechner jedoch normalerweise eine IP-Adresse, daher wird in diesem Fall die 255.255.255.255 als Empfängeradresse angegeben (eingeschränktes Broadcast). Die Transportschicht des BOOTP-Servers kann das Broadcast empfangen und sendet meist als Antwort wiederum ein Broadcast, obwohl sie die IP-Adresse des Senders kennt. Nur falls die Software des Servers in der Lage ist, den ARP-Cache zu manipulieren, kann die IP-Adresse für das Versenden der Antwort verwendet werden, da dann im ARP-Cache die zugehörige MAC-Adresse zu finden ist.

Da BOOTP auch IP benutzt, können Nachrichten verloren gehen, sich verspäten, in der falschen Reihenfolge ankommen oder dupliziert werden. Außerdem besitzt das IP-Datagramm

keine Prüfsumme, dadurch können Bits unbemerkt verloren gehen. Um dies zu verhindern, verlangt BOOTP, dass UDP eine Prüfsumme benutzt. Weiterhin dürfen BOOTP-Nachrichten nicht fragmentiert werden, da einige Rechner nicht genügend Speicher für Fragmentierung und Reassemblierung haben. BOOTP akzeptiert auch mehrere Antworten auf eine Anfrage, es wird dann die erste weiterverarbeitet.

Um Kollisionen, zum Beispiel nach einem Netzwerkausfall, zu vermeiden, benutzt BOOTP eine Zufallsverzögerung bevor es nach einem bestimmten Zeitintervall (Timeout) mit einer Übertragungswiederholung beginnt. Dabei wird das Timeout nach jedem Übertragungsversuch verdoppelt bis es 60 Sekunden erreicht hat. Damit die BOOTP-Implementierung nicht unnötig viel Speicherkapazität braucht, haben alle Felder des Rahmens eine feste Länge, außerdem haben Anfrage- und Antwort-Nachricht das gleiche Format.

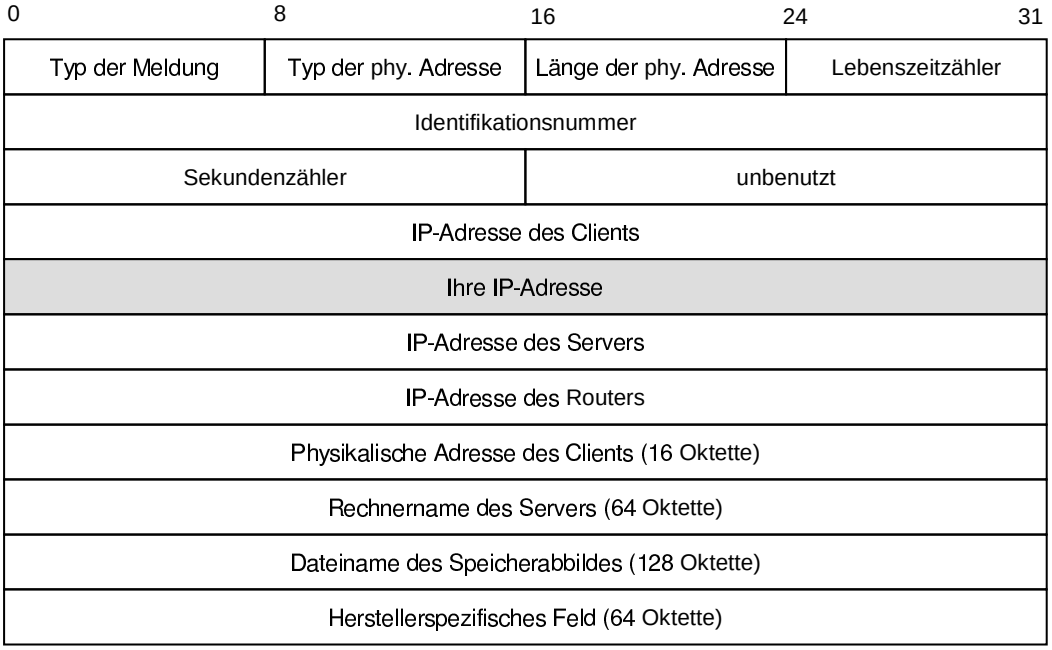

Abbildung 3: Der BOOTP-Paketrahmen wird in leicht abgewandelter Form auch von DHCP benutzt.

BOOTP bietet auch die Möglichkeit, den Dateinamen der Datei zu übersenden, die das Speicherabbild zum jeweiligen Betriebssystem beinhaltet. In der Anfrage kann dann der Name des Betriebssystems in das Feld *Dateiname des Speicherabbildes* eingetragen werden, dies ist jedoch nicht zwingend notwendig. Im *Herstellerspezifischen Feld* werden Informationen optional vom Server an den Dienstnehmer (Client) weitergegeben, dabei werden die ersten vier Oktette als Magic Cookie bezeichnet, sie definieren das Format der restlichen Parameter, z.B. der Subnetzmaske. Diese kann zwar auch per ICMP erfragt werden, aber um unnötige Netzbelastung zu vermeiden, schreiben die heutigen Standards vor, sie per BOOTP zu übermitteln. Weitere mögliche zu übertragende Parameter sind IP-Adressen von Routern, Zeit-Servern, DNS-Servern, der Rechnername, usw.

BOOTP setzt eine relativ statische Netzwerkumgebung voraus, in der jedes Endsystem einen permanenten Anschluss besitzt. Der Netzwerkmanager erstellt eine BOOTP-Konfigurationsdatei, die jedem Rechner eine IP-Adresse zuweist. Was geschieht nun in Netzwerken, in denen nicht genügend IP-Adressen für alle Rechner vorhanden sind, oder mit mobilen Endsystemen, die an jedes beliebige Netz angeschlossen werden können? Bei BOOTP müssten die Konfigurationsdateien ständig angepasst werden, der Aufwand hierfür wäre sehr groß. Für diese Fälle muss also eine dynamische Konfiguration verwendet werden.

3.3 Dynamic Host Configuration Protocol (DHCP)

Um die Autokonfiguration weiter voranzutreiben, hat die IETF (Internet Engineering Task Force) ein neues Protokoll entwickelt. Das Dynamic Host Configuration Protocol (DHCP) erweitert BOOTP in zwei Punkten: Erstens werden alle zur Konfiguration benötigten Informationen in einer Nachricht übermittelt, und zweitens kann jedem Rechner eine IP-Adresse dynamisch und schnell zugewiesen werden.

Dem DHCP-Server steht eine bestimmte Menge an IP-Adressen zur Verfügung, die dann, wenn sich neue Rechner im Netz anmelden, vergeben werden. Es gibt drei Arten der Adressvergabe, die für jedes Endsystem und für jedes Netzwerk möglich sind. Die Adresse kann durch den Netzwerkmanager manuell an einen bestimmten Rechner vergeben werden (*manuelle Konfiguration*), der Server kann eine permanente Adresse automatisch beim ersten Anmelden eines Rechners vergeben (*automatische Konfiguration*) oder aber die Adresse wird dynamisch und nur für eine begrenzte Zeit "verliehen" (*dynamische Konfiguration*). Während dieser Zeit wird die IP-Adresse vom Server nicht noch einmal vergeben.

Ein Endsystem kann also, ohne Einwirken des Administrators, mit Hilfe eines DHCP-Servers dynamisch mit allen benötigten Konfigurationsparametern (vergleiche Abschnitt 2) versorgt werden. Der Administrator legt lediglich die zu verwaltenden IP-Adressen und das Vorgehen bei der Adressvergabe fest. Sobald der Client die Informationen erhalten hat, sendet er eine Bestätigung und kann dann zu kommunizieren beginnen. Bei der dynamischen Konfiguration muss das Endsystem jedoch nach Ablauf einer bestimmten Zeitspanne ("Leihfrist") eine Verlängerung beantragen oder die Nutzung der Adresse stoppen. Eine maximale "Leihfrist" ist im DHCP-Standard nicht festgeschrieben, da in verschiedenen Netzen auch verschiedene Ansprüche an die Nutzdauer gestellt werden. Während in einem Studenten-Rechner-Pool vielleicht eine Stunde ausreicht, um etwas im Internet zu surfen, wären in einem Forschungsinstitut eventuell mehrere Tage oder Wochen als "Leihfrist" angebrachter (oder sogar eine permanente Zuweisung).

Probleme bei der Autokonfiguration können bei Rechnern mit mehreren Netzwerkkarten auftreten, wenn ein Rechner im Subnetz, ein so genannter Relay Agent, die DHCP- bzw. BOOTP-Anfrage an einen Server außerhalb des Subnetzes weiterleitet. Dabei könnte es passieren, dass dieser Server dann mehrere Anfragen vom gleichen Rechner erhält, deshalb muss der Client-Bezeichner im Nachrichtenkopf in einem solchen Fall unbedingt netzwerkkartenabhängig sein.

DHCP benutzt das BOOTP-Datagramm, allerdings mit zwei Änderungen. *Unbenutzt* wurde in *Flags* umbenannt und dient dazu, den Servern mitzuteilen, dass sie die DHCPOFFER-Antworten als Broadcast senden sollen. Das *Herstellerspezifische Feld* in BOOTP, heißt in DHCP *Optionen* und hat eine variable Länge. In diesem Feld können jedoch immer noch die gleichen Parameter übertragen werden. Zusätzlich wird hier der Nachrichtentyp (z.B. DHCPDISCOVER) codiert.

Der genaue Ablauf der dynamischen Adressvergabe kann mittels eines Zustandsdiagramms (Abb. 4) formal dargestellt werden. Wenn der Client bootet, tritt er in den INITIALIZE-Zustand ein. Um nun eine IP-Adresse zu bekommen, sendet er das DHCPDISCOVER-Broadcast an alle DHCP-Server in seinem Netz unter Verwendung der eingeschränkten Broadcast-Adresse (vergleiche BOOTP) und geht in den SELECT-Zustand über. Das Broadcast wird per UDP unter Angabe des BOOTP-Ports als Zielport verschickt. Der Client gibt nur seine MAC-Adresse an, die IP-Adresse ist ihm ja noch nicht bekannt. Alle DHCP-Server im Netz erhalten die Nachricht, aber nur diejenigen antworten dem Client, die darauf programmiert wurden. Der Client erhält nun von diesen Servern DHCPOFFER-Antworten, die die Konfigurationsinformationen für jeweils eine bestimmte IP-Adresse enthalten. Als Empfänger-Adresse würde in den Antworten normalerweise die MAC-Adresse des Clients verwendet, aber es könnte passieren, dass die IP-Schicht des Clients die Annahme dieses Paketes verweigert.

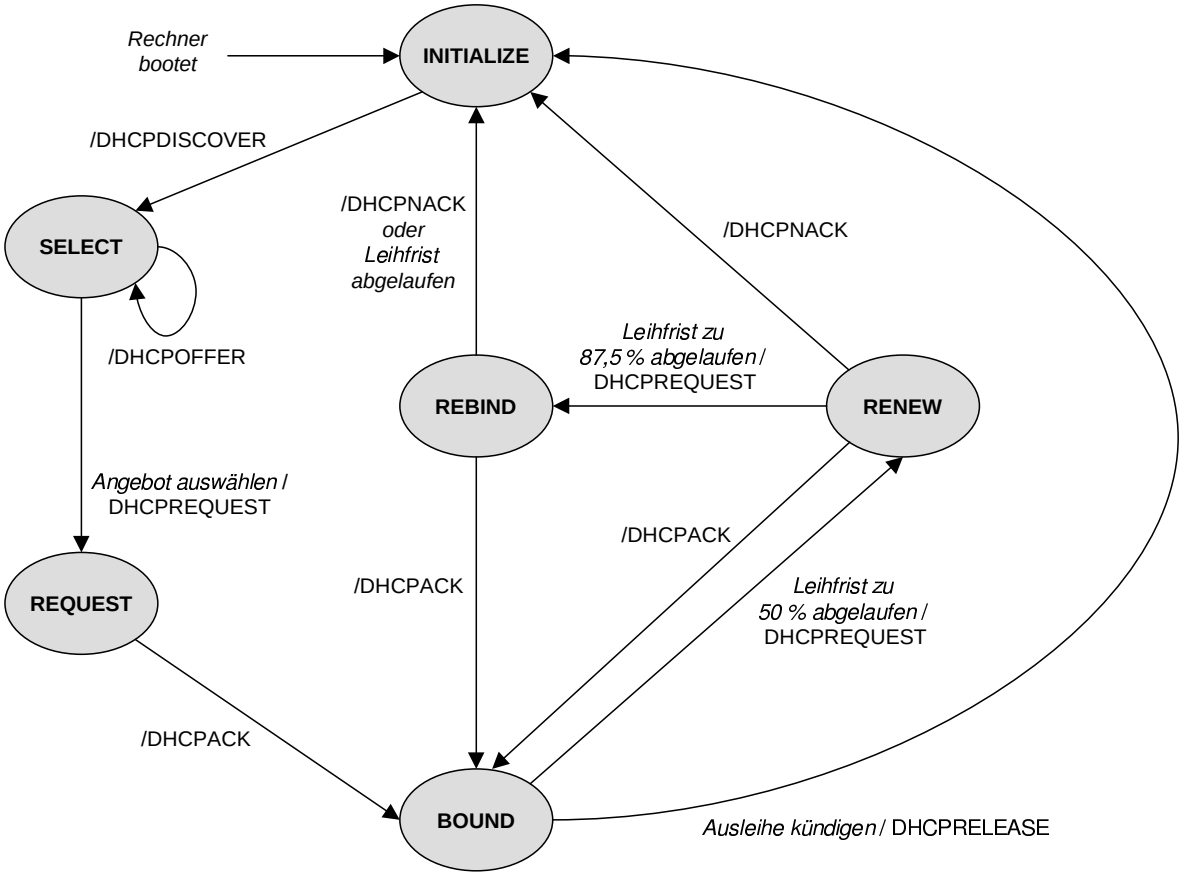

Abbildung 4: Die sechs möglichen Zustände eines DHCP-Clients. Die Übergangspfeile sind jeweils mit dem eingetretenen Ereignis beschriftet, gefolgt von einem Schrägstrich und der zu sendenden Nachricht.

Daher kann der Client in seiner Anfrage fordern, dass die Server die MAC-Broadcast-Adresse benutzen. Zur Identifikation der Nachricht wird die *Identifikationsnummer* verwendet, die bei Anfrage und Antwort übereinstimmen muss.

Der Client wählt ein Konfigurationsangebot aus (z.B. das erste) und beginnt mit dem entsprechenden Server über die Ausleihe zu verhandeln. Dazu sendet er ein DHCPREQUEST an den Server und geht in den REQUEST-Zustand über. Sobald der Server mit einem DHCPACK bestätigt, tritt der Client in den BOUND-Zustand ein und kann beginnen, die IP-Adresse zu nutzen.

DHCP macht es dem Client möglich, vor Ablauf der "Leihfrist" die IP-Adresse zurückzugeben, falls er sie nicht mehr benötigt. Dies ist vor allem in Netzwerken mit einem Engpass an IP-Adressen wichtig, denn oftmals stellt der Server zuviel Zeit zur Verfügung (DHCP verlangt ein Minimum von einer Stunde). Um die Ausleihe zu kündigen, sendet der Client die DHCPRELEASE-Nachricht an den Server und verläßt den BOUND-Zustand. Bevor der Client erneut IP nutzen kann, muss er wieder beim INITIALIZE-Zustand starten.

Jedesmal, wenn ein Client in den BOUND-Zustand übergeht, setzt er drei Zeitschalter (Timer), welche die Leihfristerneuerung und den Fristablauf kontrollieren. Der Server kann die Zeiten explizit festlegen, ansonsten benutzt der Client Standardwerte. Die Standardlänge des ersten Timers beträgt die Hälfte der ausgehandelten "Leihfrist". Wenn diese Zeit abgelaufen ist, muss der Client mittels einer DHCPREQUEST-Nachricht eine Erneuerung der Ausleihe beantragen. Er geht in den RENEW-Zustand über, während er auf eine Antwort wartet. Der Server kann nun ein DHCPACK senden und somit die weitere Verwendung der IP-Adresse durch den Client bestätigen, dieser würde dann wieder in den BOUND-Zustand eintreten.

Falls der Server nicht mit der Weiternutzung der IP-Adresse durch den Client einverstanden ist, sendet er ein DHCPNACK (negative Bestätigung) und der Client muss unverzüglich die Nutzung der IP-Adresse stoppen. Er geht in den INITIALIZE-Zustand über.

Falls ein im RENEW-Zustand befindlicher Client nach Ablauf des zweiten Timers (Standard: 87,5 Prozent der Leihfrist) keine Antwort vom Server erhält, wendet er sich mit einem DHCPREQUEST-Broadcast an alle Server im lokalen Netz. Erhält der Client eine positive Antwort von einem dieser Server, so tritt er wieder in den BOUND-Zustand ein und setzt die beiden Timer neu. Ansonsten muss der Client die Nutzung der IP-Adresse stoppen und somit in den INITIALIZE-Zustand übergehen.

Obwohl DHCP ein sehr großer Fortschritt in der Autokonfiguration ist, gibt es noch einige Verbesserungsmöglichkeiten. So kann ein DHCP-Server noch nicht auf einen DNS-Server zugreifen, um die Zuordnungsdateien dynamisch zu aktualisieren. Es müsste also bei jeder Neuvergabe einer IP-Adresse der Administrator die Dateien von Hand aktualisieren, falls ein Endsystem eine permanenten logischen Namen besitzt. Daher ist es im Moment noch üblich, die Rechnernamen an eine IP-Adresse zu binden. Dies ist natürlich sehr unpraktisch, da sich der Name bei jedem Booten ändert.

3.4 Autokonfiguration unter IPv6

Version 4 des Internet-Protokolls (IPv4) hat sich seit ihrer Einführung 1974 kaum verändert. Damals schien eine 32-bit-Adresse lang genug für die wenigen Nutzer des Internets. In den letzten Jahren ist die Anzahl der Internetrechner jedoch auf fast 60 Millionen angestiegen und es ist abzusehen, dass spätestens 2020 der IP-Adressvorrat ausgeschöpft sein wird, da durch die Adressstrukturierung nicht alle Adressen verwendet werden können. Außerdem sind die Anforderungen an das Protokoll durch die aufkommende Nutzung des Internets für Echtzeitübertragungen und E-Commerce gestiegen. Auch Computer- und Netzwerk-Hardware hat sich in den letzten Jahren stetig weiterentwickelt. Es wurde also Zeit, IP endlich den gegebenen Umständen anzupassen. Die neue IP-Generation (IPv6 oder IPng) basiert zwar auf dem gleichen Konzept wie IPv4, es unterstützt z.B. immer noch den verbindungslosen Datagrammdienst, aber viele Erweiterungen wurden vorgenommen [Huit98], hier die wichtigsten:

- *Erweiterte Adressierung:* Die Adressen sind 4-mal so lang wie bei IPv4, also 128 bit. Damit ist der Adressvorrat, zumindest aus heutiger Sicht, nahezu unerschöpflich. Außerdem wurden Anycast-Adressen und neue Hierarchieebenen eingeführt.
- *Flexibles Paketformat:* Nach dem Standardpaketkopf können noch beliebige Paketkopferweiterungen in das Datagramm eingefügt werden, die die Aufgaben des variablen Optionsfeldes in IPv4 übernehmen, das in IPv6 nicht mehr existiert.
- *Ressourcenreservierung:* IPv6 stellt Paketrahmenfelder für die Ressourcenreservierung zur Verfügung.
- *Autokonfiguration und Neunumerierung von Netzwerken: Das Internet Control Message Protocol (ICMP) der TCP/IP-Prtokollfamilie wurde erweitert und wird verstärkt für Autokonfigurationszwecke benutzt.*

Die letzten 64 Bit der IPv6-Adresse werden als Schnittstellenidentifikationsnummer bezeichnet, sie können aus der Hardware-Adresse hergeleitet werden. Während des Bootens kann nun ein Rechner eine lokal gültige Adresse (Lokaladresse) für sich generieren, indem er an das feste, binäre Präfix 1111 1110 10 zuerst 54 Nullen und dann seine Schnittstellenidentifikationsnummer anhängt. Für die neuen IP-Adressen wird die Hexadezimalschreibweise (mit

Doppelpunkt) verwendet, eine Lokaladresse würde also folgendermaßen geschrieben werden: FE80:0:0:0:XXXX:XXXX:XXXX:XXXX.

Diese lokale Adresse benutzt der Rechner nun zum Versenden einer ICMP-Solicitation-Nachricht an alle Router, dafür wird die spezielle Zieladresse FF02::2 verwendet. Erhält ein Router nun eine Solicitation-Anfrage, so muss er darauf mit einer ICMP-Router-Advertisement-Nachricht antworten, die unter anderem das M-Bit und ein Präfix-Feld beinhaltet. Ist M gesetzt, so sollte der Rechner einen DHCP-Server zur Konfiguration kontaktieren, ansonsten kann er seine globale IP-Adresse ohne Serverhilfe selbst generieren. Hierzu benutzt er das im Präfix-Feld angegebene 64-Bit-Präfix (eventuell sind auch mehrere Präfixoptionen angegeben) und einen zufälligen 64-Bit-Bezeichner.

Obwohl die Wahrscheinlichkeit minimal ist, könnte die gewählte Adresse im gleichen Subnetz bereits existieren, dieses muss der Rechner testen, bevor er die IP-Adresse nutzen kann. Dazu sendet er mehrere ICMP-Neighbor-Discovery-Anfragen an die Adresse. Erhält er hierauf keine Antwort, so kann er beginnen, die IP-Adresse zu nutzen. Trotzdem muss das Senden in regelmäßigen Abständen wiederholt werden, da ein Netzwerkfehler aufgetreten sein könnte.

Diese Art der Konfiguration ist zwar sehr einfach, hat allerdings zwei Mängel: Der Adressraum wird nicht effizient genutzt, und die Netzzugangskontrolle ist ungenügend. Außerdem beinhaltet Netzwerkkonfiguration mehr als nur das Generieren einer IP-Adresse. Obwohl die Subnetzmaske und die Adresse des lokalen Routers mittels der so genannten Neighbor Discovery ermittelt werden können, braucht der Rechner noch die Adressen erreichbarer Server (z.B. DNS-Server). Daher wird er, zumindest für diese Informationen, versuchen einen DHCP-Server zu kontaktieren, allerdings nur dann, wenn in der ICMP-Router-Advertisement-Nachricht ein entsprechendes Bit (O-Bit) gesetzt ist.

Wurde dem Rechner mitgeteilt, dass seine gesamte Konfiguration über einen DHCP-Server laufen soll, so sendet er eine DHCP-Solicitation-Nachricht (benutzt UDP) an alle DHCP-Server und Relay Agents (mit der spezifischen Zieladresse FF02::1:2). Als Senderadresse wird wiederum die Lokaladresse verwendet. Die erreichbaren Server antworten mit einer DHCP-Advertisement-Nachricht. Der Rechner wählt einen Server (und ggf. Relay Agent) aus und sendet eine DHCP-Anfrage an ihn. Um eine IP-Adresse anzufordern, wird an diese Nachricht eine DHCP-Adresserweiterung angehängt. Diese wird vom Server ausgefüllt und dann zurückgesendet. Der Rechner kann auch noch beliebige andere Erweiterungen mitsenden, z.B. DNS-Server-Erweiterung, die dann auch vom Server beantwortet werden. Ansonsten stimmt das Vorgehen mit dem im Abschnitt 3.3 beschriebenen überein.

Nutzt ein Rechner einen zugewiesenen Parameter nicht mehr, so schickt er eine DHCP-Freigabe-Nachricht an den Server. Falls sich Parameter ändern sollten, sendet der Server eine DHCP-Rekonfigurations-Nachricht an das entsprechende Endsystem.

Die IP-Adressen der neuen Generation werden eine limitierte Lebenszeit haben, um sie an Topologieänderungen anpassen zu können. DHCP-Server werden diese Lebenszeit mit der jeweiligen IP-Adresse zusammen versenden. Dabei unterscheidet man die tatsächliche Lebenszeit und die verwendbare Lebenszeit. Läuft letztere ab, so ist die Adresse zwar noch eine Weile gültig, der Rechner darf sie jedoch nicht mehr nutzen, um neue Verbindungen aufzubauen. Nach Ablauf der tatsächlichen Lebenszeit ist die Adresse ungültig. Wird die IP-Adresse ohne DHCP-Server vom Endsystem selbst konfiguriert, so ist die Lebenszeit des Präfixes ausschlaggebend für die Gültigkeit der IP-Adresse. Diese Lebenszeit wird vom Router beim Senden des Präfixes bekannt gegeben.

IPv6 erlaubt zwar dynamische Konfiguration, wie in Abschnitt 3.3 bereits erwähnt, existiert für DHCP jedoch noch keine Methode, um den DNS-Server automatisch bei jeder Adressvergabe und -rückgabe zu aktualisieren. Aber die IETF arbeitet derzeit an diesem Problem.

4 Autokonfiguration in Ad-hoc-Netzwerken

Für bestimmte Ad-hoc-Netzwerke gibt es (unter IPv4) bereits Lösungen für das Zeroconf-Problem [Trol00]. Zwei Betriebssysteme (MS Windows98, Apple MacOS 8.5) bieten Methoden an, mit denen ein Endsystem sich automatisch eine lokale IP-Adresse zuweisen kann, falls kein DHCP-Server erreichbar ist. Auf diese Methoden wird später eingegangen.

Der dynamische Aufbau eines Ad-Hoc-Netzwerkes ist besonders da sinnvoll, wo nur sehr wenige Rechner zusammen ein Netzwerk bilden, z.B. in einer Arztpraxis. Hier wird es meist keinen Administrator geben, der das Netz konfiguriert oder einen Server für die Adressverteilung verwaltet, jeder Rechner muss also seine IP-Adresse selbst festlegen. Diese Adressen gelten jedoch nur im lokalen Netzsegment und werden nicht geroutet. Es muss noch erwähnt werden, dass der hier beschriebene Mechanismus nur auf DHCP-fähigen Rechnern funktioniert. Das Endsystem greift erst auf die automatische Adresswahl zurück, falls der Versuch, einen DHCP-Server zu erreichen, mehrere Male fehl schlägt. Das genaue Vorgehen wird im folgenden beschrieben.

Wie in Abschnitt 3.3 beschrieben, sendet ein Rechner zunächst DHCPDISCOVER-Nachrichten aus, um die Anwesenheit eines DHCP-Servers festzustellen. Dabei versendet ein Client unter Win98 4-mal diese Nachricht im Abstand von jeweils 6 Sekunden, unter MacOS 8.5 nur 3-mal im Abstand von 4, 8 und 16 Sekunden. Erhält der Rechner auf all seine DHCPDISCOVER-Nachrichten keine Antwort, so beginnt er selbst eine IP-Adresse zu konfigurieren. Der zulässige Adressbereich wurde von der IANA (Internet Assigned Number Authority) mit 169.254/16 (die ersten 16 Bit kodieren das Netzwerk) festgelegt, wobei die ersten und letzten 256 Nummern nicht vergeben werden dürfen. Sobald der Rechner per Zufallsalgorithmus eine Adresse gewählt hat, muss er testen, ob diese in seinem lokalen Netz bereits benutzt wird. Dieses geschieht durch das Senden eines ARP-Broadcasts, die IP-Adresse des Senders muss hierbei mit Nullen ausgefüllt werden. Ist die gewählte IP-Adresse bereits vergeben, so antwortet der betroffene Rechner an die angegebene MAC-Adresse, und das Endsystem muss eine neue IP-Adresse auswählen, die wiederum getestet werden muss. Während er auf die ARP-Antwort wartet, muss der Rechner auch auf Anfragen anderer Rechner achten, die vielleicht exakt die gleiche IP-Adresse konfigurieren wollen, wie die von ihm gewählte.

Um das Netz nicht mit unendlich vielen ARPs zu überschwemmen, muss ein Endsystem, nach einer bestimmten Anzahl an Fehlversuchen (d.h. die IP-Adressen sind bereits vergeben), ohne IP-Adresse booten. Bei beiden oben genannten Betriebssystemen wurde diese Anzahl auf 10 Versuche festgelegt. Da während der DHCPDISCOVER-Nachricht ein Netzwerk- oder Serverfehler aufgetreten sein kann, muss ein Rechner in regelmäßigen Abständen (bei Ethernet: 5 Minuten) neue Broadcasts dieser Art verschicken. Erhält das Endsystem irgendwann eine Antwort von einem DHCP-Server, so muss es sofort versuchen, eine IP-Adresse vom Server zu mieten (zum genauen Vorgehen sh. Abschnitt 3.3). Ist dieser Versuch erfolgreich und unterstützt der Rechner die Zuweisungen mehrerer IP-Adressen auf eine Schnittstelle nicht, so muss die autokonfigurierte Adresse sofort aufgegeben werden. Hierbei sollte jedoch beachtet werden, dass es eventuell noch laufende Verbindungen mit dieser Adresse gibt und deshalb Daten verloren gehen könnten.

Ein weiteres Problem ist, dass, falls während des Bootens das Netzwerk unterbrochen ist, mehrere Rechner die gleiche IP-Adresse konfiguriert haben können. Wird ein solcher Fehler entdeckt, so muss das gesamte Netzwerk neue IP-Adressen autokonfigurieren.

Aus diesen Ausführungen ist ersichtlich, dass der beschriebene Autokonfigurationsmechanismus nicht auf größere Netzwerke angewandt werden kann, die Bootzeiten und Kollisionswahrscheinlichkeiten wären zu hoch, und bei einem Netzwerkfehler würde ein totales Chaos eintreten.

5 Zero-Konfigurationsanforderungen

Für immer mehr Netzwerke lohnt sich die Administration durch Fachpersonal nicht, da sie zu klein oder nur von kurzer Lebensdauer sind. Und auch die Nutzer dieser Netze haben zumeist weder Zeit noch Interesse, sich mit der Netzwartung und Konfiguration auseinanderzusetzen. Für solche Netzwerke ist Autokonfiguration nicht genug, es müssen Protokolle entwickelt werden, die keinerlei Konfiguration durch die Nutzer erfordern und auch nicht von Informationen eines zentralen Servers abhängen: Zero-Konfigurationsprotokolle, oder einfach Zeroconf-Protokolle.

Welche Anforderungen an solche Protokolle gestellt werden, wurde von der Zeroconf Working Group der IEFT in einem Internet Draft zusammengestellt [Hatt00]. Zeroconf- und Non-Zeroconf-Protokolle (Protokolle, die Konfigurations-Server nutzen) müssen nebeneinander existieren können, und ein Endsystem sollte in der Lage sein, zu erkennen, wann es von der Nutzung des einen zur Nutzung des anderen übergehen muss. Dies kann der Fall sein, wenn der Rechner das Netzwerk wechselt oder ein Non-Zeroconf-Server on- bzw. offline geht. Aber auch wenn ein Rechner von einem Netz in ein anderes wechselt und beide Netze benutzen Zeroconf-Protokolle, muss das Protokoll neu initialisiert werden, um Adresskonflikte zu vermeiden. Außerdem sollten Zeroconf- und Non-Zeroconf-Protokolle auf den verschiedenen Ebenen der Konfiguration kompatibel sein, z.B. kann ein Rechner für die IP-Rechnerkonfiguration ein Zeroconf-Verfahren benutzen, die IP-Namensauflösung wird jedoch mit Hilfe eines Non-Zeroconf-Protokolles realisiert, falls es ein DNS-Server erreichbar ist.

Für die folgenden Betrachtungen wird der gesamte Konfigurationsvorgang in vier Teilgebiete getrennt: IP-Rechnerkonfiguration, IP-Namensauflösung, IP-Multicast-Adresszuweisung und Diensterkennung (Service Discovery).

5.1 IP-Rechnerkonfiguration

Die IP-Rechnerkonfiguration ist abgeschlossen, wenn ein Rechner auf der entsprechenden Schnittstelle seine IP-Adresse, Subnetzmaske und den Default Gateway (falls Router existent) nutzen kann.

Um eine reibungslose Kommunikation zu garantieren, muss das Zeroconf-Protokoll sicherstellen, dass alle Rechner eines Subnetzes die gleiche Subnetzmaske benutzen, und dass die Verknüpfung von IP-Adresse und Maske für alle Rechner das gleiche Ergebnis liefert. Es dürfen nicht zwei gleiche IP-Adressen in einem Netzsegment existieren. Falls dies jedoch der Fall sein sollte, so müssen die betroffenen Rechner in der Lage sein, die Adresskollision zu erkennen. Falls ein Router im Subnetz vorhanden ist, sollte es jedem Endsystem möglich sein, diesen als Default Gateway auszuwählen.

Außerdem müssen zwei Subnetze miteinander zu einem verschmelzbar sein. Falls durch eine Verschmelzung oder andere Umstände ein DHCP-Server erreichbar wird, muss dieser zur Konfiguration benutzt werden und nicht das Zeroconf-Protokoll.

5.2 IP-Namensauflösung

Die IP-Namensauflösung wird normalerweise mit Hilfe von DNS (Domain Name System) vorgenommen, dazu muss jedoch vorher die IP-Adresse des DNS-Servers konfiguriert werden.

Es muss jedem DNS-fähigen Rechner möglich sein, festzustellen, ob ein DNS-Server erreichbar ist oder nicht. Ist kein Server erreichbar, oder kann ein Rechner aus anderen Gründen DNS nicht nutzen, so muss er das Zeroconf-Protokoll verwenden. Dieses muss Rechnernamen in

IP-Adressen umwandeln können, und es muss jedem Rechner erlauben, einen eigenen, noch nicht vergebenen Namen zu wählen und diesen auch zu verteidigen. Außerdem muss jedes Endsystem in regelmäßigen Abständen prüfen können, ob sein Name von einem anderen Rechner in der Domäne genutzt wird (Namenskollision). Natürlich sollten alle Rechner einer Domäne den gleichen Domänennamen verwenden.

5.3 IP-Multicast-Adresszuweisung

Beim IP-Multicast wird eine Nachricht gleichzeitig an eine Gruppe von Rechnern verschickt. Für die Adressierung einer solchen Gruppe werden IP-Adressen der Klasse D verwendet. Die Gruppenzugehörigkeit muss vor dem Senden festgelegt werden.

Nur falls keine Non-Zeroconf-Protokolle (z.B. MADCAP) für die Multicast-Adresszuweisung verwendet werden können, sollte das Zeroconf-Protokoll benutzt werden. Das Protokoll sollte es einem Endsystem erlauben, eigenständig eine noch nicht verwendete Multicast-Adresse für einen bestimmten Rechnerbereich und mit einer bestimmten Lebenszeit festzulegen. Dabei muss jeder Rechner Kollisionen mit der gewählten Adresse bemerken und gegebenenfalls eine neue Adresse generieren können. Der gewählte Rechnerbereich muss sich nicht ausschließlich auf das eigene Subnetz beschränken, daher müssen Multicast-Nachrichten auch von Routern weiterleitbar sein. Der letzte Rechner, der ein bestimmte Multicast-Adresse nutzt, muss fähig sein, diese Adresse zu vernichten, egal ob er oder ein anderer Rechner sie initialisiert hat.

5.4 Diensterkennung (Service Discovery)

In den meisten Netzwerken werden zentrale Dienste angeboten, zum Beispiel Druckerdienst oder sogar Druckermanager. Über die Existenz dieser Dienste müssen die Hosts informiert werden.

Das Zeroconf-Protokoll zur Diensterkennung muss unabhängig von einem bestimmten Dienst sein, damit es für alle möglichen Dienste genutzt werden kann. Damit sich jeder Rechner über die angebotenen Dienstleistungen (Art der Dienste, Identifikation, Eigenschaften) informieren kann, muss das Protokoll außerdem Solicitation- und Advertisement-Nachrichten unterstützen.

6 Fazit

Die beschriebenen Autokonfigurationsprotokolle BOOTP und DHCP stellen ein Alternative zum sehr simplen RARP dar, wenn ein Rechner eine IP-Adresse zugewiesen haben möchte. Da beide Protokolle UDP benutzen, können Adressanfragen auch von Routern weitergeleitet werden, dieses ist bei RARP nicht möglich. Neben der IP-Adresse können BOOTP und DHCP auch andere benötigte Konfigurationsparameter zuweisen, z.B. Router- oder Server-Adressen. Für Netzwerke mit einer begrenzten Menge an IP-Adressen oder drahtlose Netzwerke bietet DHCP einen großen Vorteil. Es kann IP-Adressen dynamisch, also ohne den anmeldenden Rechner zu kennen, für eine bestimmte Laufzeit vergeben. DHCP wird auch zur Autokofiguration von Ad-hoc und IPv6-Netzwerken verwendet.

Die im vorangegangenen Abschnitt erwähnten Anforderungen an Zerokonfigurationsprotokolle wurden zum Teil schon in den früher beschriebenen Autokonfigurationsmechanismen von Ipv4 und Ipv6 realisiert. Rechner in Ad-hoc- und Ipv6-Netzwerken können sich selbst IP-Adressen zuweisen, falls kein Konfigurations-Server erreichbar ist. Mit diesen Adressen ist es zumindest möglich, im lokalen Netzsegment zu kommunizieren.

Im Konflikt zu den Zerokonfigurationsanforderungen stehen jedoch häufig Sicherheitsaspekte, die wesentlich wichtiger zu bewerten sind als eine aufwandlose Rechnerkonfiguration. In einem Zeroconf-Netzwerk ohne Sicherheitsbestimmungen könnte ein Rechner beispielsweise alle zur Verfügung stehenden Ressourcen, wie IP-Adressen oder Rechnernamen, horten, indem er auf alle eintreffenden Anfragen mit "Schon belegt" antwortet. Außerdem wäre es möglich, dass ein Server illegal Dienste anbietet und dann einen Rechner absichtlich falsch konfiguriert.

Um diesen Mißbrauch zu verhindern, müssen bereits durch Softwarehersteller und -entwickler Sicherheitsvorkehrungen getroffen werden. Dieses wird die baldige Einführung von Zeroconf-Protokollen sicher bremsen.

Literatur

[Hatt00] M. Hattig. *Zeroconf Requirements*. IETF, März 2000. draft-ietf-zeroconf-reqts-03.txt.

[Huit98] Ch. Huitema. *IPv6 - The New Internet Protocol*, Kapitel 4: Plug and Play, S. 91–143. Prentice Hall. 1998.

[Trol00] R. Troll. *Automatically Choosing an IP Address in an Ad-Hoc IPv4 Network*. IETF, März 2000. draft-ietf-dhc-ipv4-autoconfig-05.txt.

Zero-Knowledge-Protokolle

Michael Böhl

Kurzfassung

Zero-Knowledge-Protokolle stellen eine Möglichkeit dar, eine andere Person davon zu überzeugen, dass man ein Geheimnis besitzt, ohne dieses preiszugeben. Genau genommen wird absolut keine Information über das Geheimnis übermittelt – eine Eigenschaft, die z.B. gängige Public-Key-Authentisierungsverfahren nicht erfüllen. Dies prädestiniert Zero-Knowledge-Verfahren für den Einsatz in Anwendungen, die über unsichere Kommunikationssysteme wie das Internet Identitäten von Personen oder Systemkomponenten überprüfen müssen. Anwendungsgebiete für Zero-Knowledge-Protokolle sind zur Zeit in erster Linie Smartcards („intelligente“ Chipkarten), die beispielsweise als elektronischer Pass verwendet werden können. Daneben gibt es z.B. auch Protokollvarianten, die für den Einsatz im Bereich elektronischer Bargeldsysteme entworfen wurden.

1 Einleitung

Daten, die im Klartext über das Internet übertragen werden, sind relativ einfach für Unbeteiligte lesbar und manipulierbar. Dies stellt ein Problem für verteilte Anwendungen dar, die hohe Anforderungen an die Sicherheit (Authentizität, Integrität und Vertraulichkeit) der übermittelten Daten stellen. Der vorliegende Text stellt eine sichere Lösung für das Problem der Authentisierung vor.

Bei Kommunikationsvorgängen in einem Computernetzwerk gibt es für die Teilnehmer keine andere Möglichkeit, sich zu authentisieren oder ihr Gegenüber zu identifizieren als mit Hilfe der übermittelten Daten. Dem steht jedoch das erwähnte Problem gegenüber, dass die Daten unter Umständen durch Unbeteiligte mitgelesen und anschließend missbraucht werden – also müssen die Protokolle Mechanismen enthalten, durch die eine solche Manipulation zumindest erkannt werden kann. Die Erkennung reicht jedoch nicht aus, wenn der Angreifer beim Abhören in Besitz der geheimen Information kommt, über welche die Kommunikationspartner einander identifizieren. Sobald dies geschehen ist, kann man den Angreifer nämlich nicht mehr vom „echten“ gewünschten Kommunikationspartner unterscheiden, da beide die gleiche Menge an relevanter Information besitzen. Bei den meisten Protokollen muss jedoch die geheime Information irgendwie über das Kommunikationsmedium übertragen werden, wenn auch nur indirekt z.B. als Schlüssel einer chiffrierten Nachricht. Hierin unterscheiden sich Zero-Knowledge-Protokolle von anderen Verfahren zur Authentisierung – der Protokollablauf kann keine Informationslecks aufweisen, weil einfach überhaupt keine wichtige Information übertragen wird. Dennoch gelingt es mit ihnen, sein Gegenüber von der eigenen Identität zu überzeugen.

Kapitel 2 gibt einen Überblick über verschiedene Authentisierungsverfahren und ihre jeweiligen Schwachstellen mit den dadurch möglichen Angriffen. Die Konsequenz daraus, nämlich dass man überhaupt keine wichtige Information übertragen sollte, führt zu den Zero-Knowledge-Protokollen. Kapitel 3 stellt verschiedene Protokoll-Varianten vor und zeigt den allgemeinen Aufbau von Zero-Knowledge-Authentisierungsverfahren. In Kapitel 4 werden die

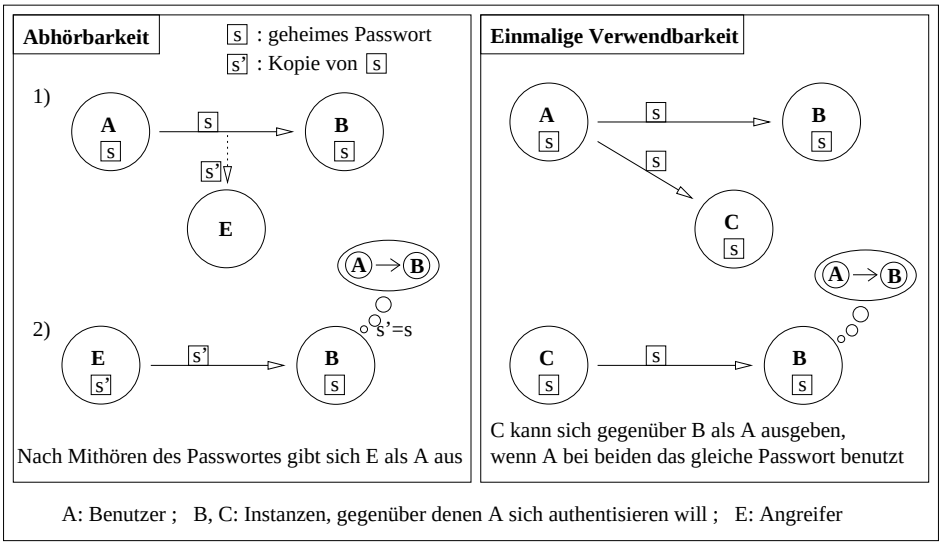

Abbildung 1: Nachteile bei der Verwendung von (unverschlüsselten) Passwörtern

bisher realisierten oder zumindest vorgeschlagenen Anwendungsgebiete der Protokolle besprochen, Smartcards und Electronic Cash. Zero-Knowledge-Protokolle eignen sich prinzipiell für alle interaktiven Authentisierungsvorgänge, sind aber sehr viel weniger bekannt als Public-Key-Verfahren – vielleicht der Grund für die geringe Verbreitung.

Im Folgenden werden zwei Personen, die miteinander kommunizieren, beispielhaft als Alice und Bob bezeichnet.

2 Authentisierung

Damit Alice sich eindeutig identifizieren kann, muss sie etwas besitzen, das kein anderer hat. Dies kann ein biologisches Merkmal sein, zum Beispiel ein Fingerabdruck, oder etwas vergleichbar Einmaliges, wie eine Unterschrift. Man kann seine Identität aber auch durch Wissen beweisen, das niemand anderem zur Verfügung steht – also durch ein Geheimnis. In diesem Fall bedarf es zur Authentisierung von Alice gegenüber Bob eines Verfahrens, das Bob davon überzeugt, dass Alice das Geheimnis kennt.

2.1 Passwörter

Ein einfacher Fall eines solchen Geheimnisses ist ein Passwort. Bei diesem Verfahren kennt Bob das Geheimnis und kann so überprüfen, ob es von einer als Alice zu identifizierenden Person korrekt übertragen wurde. Diese Methode eignet sich also nicht zur Authentisierung gegenüber verschiedenen unabhängigen Instanzen oder Personen, da jede Instanz das Passwort gegenüber anderen Instanzen verwenden könnte, um sich als Alice auszugeben (Bild 1).

2.2 Challenge-Response-Verfahren

Ein Challenge-Response-Verfahren besteht aus einer Anfrage an eine zu identifizierende Person und einer Antwort, die aus der Anfrage durch eine Funktion mit einem weiteren Parameter (Geheimnis) erzeugt wird. Wird als Geheimnis ein symmetrischer Schlüssel eingesetzt, den beide beteiligten Parteien kennen, unterscheidet sich das Verfahren von der Identifikation

mittels Passwort nur dadurch, dass das Geheimnis nicht direkt übertragen wird, was einen Angriff durch Abhören der Kommunikationsverbindung erschwert. Benutzt man statt der symmetrischen eine Public-Key-Verschlüsselungstechnik, so ist es auch möglich, sich gegenüber verschiedenen unabhängigen Instanzen zu identifizieren, da diese das Geheimnis (privater Schlüssel) nicht mehr kennen müssen, sondern den dazu passenden öffentlichen Schlüssel zur Verifizierung benutzen (Bild 2).

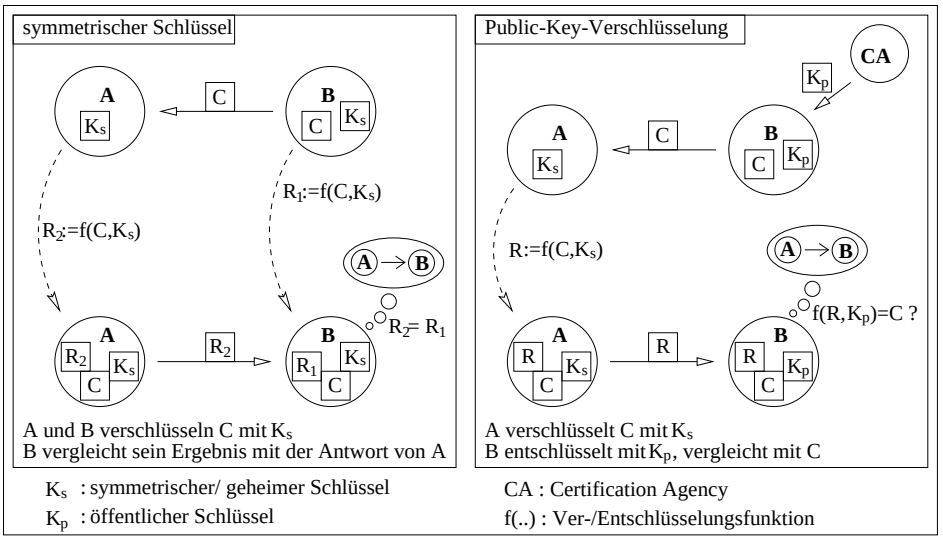

Abbildung 2: 2-Phasen-Protokolle (Challenge-Response)

Im Idealfall sollte Bob durch den Authentisierungsvorgang keine Information über das Geheimnis von Alice erlangen, ansonsten könnte er sich nach vollständiger Kenntnis des Geheimnisses als Alice ausgeben. Benutzt man „Challenge-Response“ als Authentifizierungs-Verfahren, ist diese Forderung nur bedingt erfüllt, da jede Durchführung etwas über die geheime Information verrät – es ist also möglich, durch mehrmalige (u.U. sehr häufige) Durchführung des Verfahrens mit unterschiedlichen günstig gestellten Anfragen (*Chosen-Plaintext-Attack*, Bild 3) in den Besitz der geheimen Information zu gelangen.

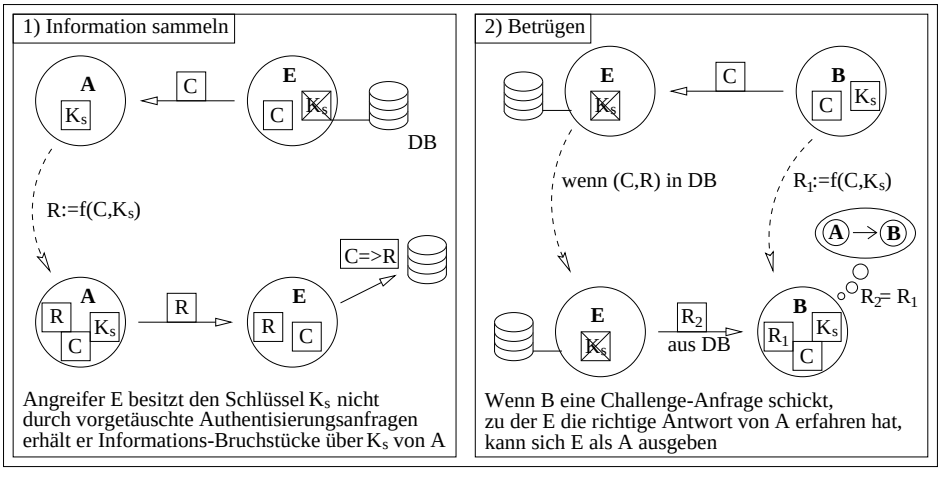

Abbildung 3: Schwachpunkt von Challenge-Response-Protokollen: E kann Information über den geheimen Schlüssel gewinnen (*Chosen-Plaintext-Attack*)

Verfahren, bei denen effektiv absolut keine Information über das Geheimnis an Bob übertragen wird und nach deren erfolgreicher Durchführung Bob trotzdem von der Identität seines

Kommunikationspartners überzeugt ist, nennt man Zero-Knowledge-Protokolle. Interessanterweise lässt sich für diese Protokolle sogar beweisen, dass keine „wichtige“ Information übertragen wird.

2.3 Zero-Knowledge-Verfahren

In der Literatur gibt es verschiedene Analogien, um die Funktionsweise von Zero-Knowledge-Verfahren anschaulich zu beschreiben. Sehr einfach zu verstehen ist z.B. die kurze Einführung in [QUGU90] in Form eines Märchens für Kinder.

Das folgende Szenario geht davon aus, dass die Information, die nicht preisgegeben werden soll, ein Tastatur-Code ist, der eine Tür öffnet. Die Tür befindet sich zwischen zwei Räumen, die einen gemeinsamen Vorraum haben (Bild 4). Das Protokoll, durch welches Alice Bob beweisen möchte, dass sie den Code kennt, ohne dass er ihn erfährt, beginnt, indem Alice den Vorraum betritt, während Bob draußen wartet. Sie wählt dann zufällig einen der beiden Räume aus, geht hinein und schließt die Tür. Danach kommt Bob in den Vorraum und wählt selbst einen Raum aus, aus dem er Alice herauszukommen bittet. Ist Alice bereits im richtigen Raum, so kommt sie einfach heraus – ansonsten muss sie zuerst den geheimen Code für die Tür mit dem Schloss benutzen, um den Raum zu wechseln. Bob kann nicht wissen, ob sie den Code benutzen musste – nur falls Alice aus dem falschen Raum kommt, ist der Beweis fehlgeschlagen (Bob nimmt dann an, Alice kenne den geheimen Code nicht). Kommt sie aus dem richtigen Raum, so besteht eine 50-prozentige Wahrscheinlichkeit, dass Alice den Code nicht benutzen musste und ihn vielleicht auch gar nicht kennt; diese Wahrscheinlichkeit lässt sich jedoch durch wiederholte Durchführung des Verfahrens beliebig verkleinern (zum Beispiel auf etwa 0,0001% (1 zu 1 Million) nach 20 Abläufen).

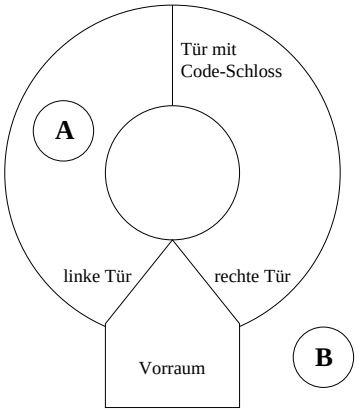

Abbildung 4: Grundriss des Gebäudes mit Codeschloss-Tür

2.4 Die Zero-Knowledge-Eigenschaft

Bob kann alle Informationen aufzeichnen, die er während der Durchführung des Verfahrens erhält, indem er den Vorgang aus seiner Sicht mit einer Videokamera filmt. Auf dem Film sieht man also Alice in den Vorraum verschwinden, dann betritt Bob den Vorraum, man hört ihn „rechts“ oder „links“ rufen, worauf Alice durch die entsprechende Tür kommt. Die Zero-Knowledge-Eigenschaft (also die Bedingung, dass kein Wissen über das Geheimnis übertragen wird) lässt sich dadurch belegen, dass es einem Simulator gelingen kann, mit einem Alice-Double ohne Kenntnis des Zahlencodes ein identisches Video zu erstellen. Dazu filmt er jede Szene genau so, wie es Bob tut – nur dass in 50 Prozent aller Versuche das Alice-Double durch

die falsche Tür kommen wird, da sie nicht in den anderen Raum wechseln kann. Diese Szenen kann der Simulator jedoch aus dem Video herausschneiden, so dass sich für einen Unbeteiligten das Video von Bob und das des Simulators nicht unterscheiden lassen, die Protokollabläufe sehen alle gültig aus. Folglich gelingt es Bob auch nicht, mit Hilfe des Videos einen Dritten davon zu überzeugen, dass Alice das Geheimnis kennt – dieser hält entweder das Video für manipuliert oder er unterstellt Bob, dass er sich mit Alice abgesprochen hätte.

Die Zero-Knowledge-Eigenschaft besteht also darin, dass ein Außenstehender einen präparierten Dialog nicht von einem echten unterscheiden kann, auch wenn der echte Dialog einen der Kommunikationspartner gleichzeitig davon überzeugt, dass der andere das Geheimnis kennt. Dies zeigt, dass keine Information übertragen wird, da man diese Information ja sonst irgendwie „festhalten“ könnte.

Schon bei einer leichten Modifikation des oben beschriebenen Verfahrens geht die Zero-Knowledge-Eigenschaft verloren: Die scheinbare Erleichterung, dass Alice und Bob den Vorraum gemeinsam betreten und Bob sich demonstrieren lässt, dass Alice in einem der Räume verschwinden und aus dem anderen wieder auftauchen kann, überzeugt zwar Bob schneller, dass Alice das Geheimnis kennt – dies ist aber kein Zero-Knowledge-Protokoll mehr, da Alice offensichtlich die Kenntnis des Geheimnisses demonstrieren muss, was auch einen Außenstehenden überzeugen würde.

3 Zero-Knowledge-Protokolle

Das bekannteste und in der Praxis wichtigste Zero-Knowledge-Verfahren ist der Fiat-Shamir-Algorithmus, der 1987 von Amos Fiat und Adi Shamir veröffentlicht wurde ([FiSh87]). Er beruht auf der Schwierigkeit der Berechnung von modularen Quadratwurzeln, welche ein NP-Problem darstellt, also eines der Probleme, bei denen lediglich die Verifizierung einer gegebenen Lösung mit polynomialem Zeitaufwand möglich ist, das Finden einer Lösung aber nicht, zumindest nicht mit den bekannten Algorithmen.

Prinzipiell kann man auf alle NP-Probleme ein Zero-Knowledge-Protokoll aufbauen, das haben Goldreich, Micali und Wigderson 1986 in [GuUQ92] gezeigt – dieses Protokoll ist dann aber nur so lange sicher, wie keine effiziente Lösung für ein NP-vollständiges Problem gefunden wird. Beispielhaft für die Umsetzung verschiedener mathematischer Probleme in ein Zero-Knowledge-Protokoll werden nachfolgend neben dem Fiat-Shamir-Algorithmus (Abschnitt 3.2) auch Varianten beschrieben, die den diskreten Logarithmus (Abschnitt 3.3) bzw. die Isomorphie von Graphen (Abschnitt 3.4) benutzen. Abschnitt 3.5 beschreibt anschließend die Gemeinsamkeiten der Protokoll-Varianten und versucht so, die allgemeine Struktur von Zero-Knowledge-Protokollen deutlich zu machen.

3.1 Mathematische Grundlagen (Modulare Arithmetik)

Beim Fiat-Shamir-Algorithmus werden alle Berechnungen in einer multiplikativen Restgruppe durchgeführt, der gleichen mathematischen Struktur, die auch zum Beispiel vom Public-Key-Algorithmus RSA benutzt wird. Der folgende Abschnitt behandelt die Eigenschaften dieser Gruppen und geht insbesondere auf die erhöhte Schwierigkeit der Berechnung von Quadratwurzeln in ihnen ein.

$(Z_n, \oplus)$ bildet bei geeigneter Definition der Addition und Subtraktion eine additive Restgruppe mit den n Elementen $0, 1, \ldots, n-1$ [BeSW98]. Zusammen mit der Multiplikation bildet die Menge Z_n im Allgemeinen jedoch keine Gruppe – es gibt Elemente, die kein Inverses haben, da Vielfache von ihnen auf das neutrale Element abgebildet werden (nichttriviale Nullteiler,

z.B. 5 und 11 in Z_{55}). Nimmt man aber nur die Elemente, die zu n teilerfremd sind und bildet daraus die Menge Z_n^*, so ist $(Z_n^*, \otimes)$ eine multiplikative Gruppe. Ein Element $a \in Z_n^*$ heißt *quadratischer Rest modulo* n, wenn es eine Zahl $b \in Z_n^*$ gibt mit $b^2 = b \otimes b \equiv a \pmod{n}$. Dann heißt b eine Quadratwurzel von a modulo n. Zum Beispiel hat die Zahl 34 in Z_{55}^* die Quadratwurzeln 12, 23, 32 und 43, da z.B. $23 \otimes 23 = 529 \bmod 55 = 34$.

Die Berechnung von Quadratwurzeln in multiplikativen Restgruppen ist dann besonders schwierig, wenn n keine Primzahl ist. Ist z.B. n das Produkt von zwei Primzahlen p und q, dann ist das Berechnen von Quadratwurzeln genauso schwierig wie das Faktorisieren von n (Beweis siehe [BeSW98], S.122) – und auf die ausreichende Schwierigkeit der Faktorisierung großer Zahlen verlässt sich z.B. der Public-Key-Algorithmus RSA. Auf dem Problem der Quadratwurzel-Berechnung beruht die Sicherheit des Fiat-Shamir-Algorithmus, der nun beschrieben wird.

3.2 Der Fiat-Shamir-Algorithmus

Das Verfahren nimmt die Existenz einer vertrauenswürdigen zentralen Instanz (trusted center) an, die im Folgenden als Zentrale bezeichnet wird. Die Zentrale wird benutzt, um die von Protokollteilnehmern veröffentlichten Daten zu speichern und zuverlässigen Zugriff auf sie zu ermöglichen. Alternativ können zwei Kommunikationspartner, die das Protokoll zur Authentisierung durchführen wollen, die benötigten Daten zu einem früheren Zeitpunkt ausgetauscht haben – der Austauschvorgang setzt jedoch einen sicheren Kommunikationskanal voraus, um Manipulationen vorzubeugen.

Die Zentrale berechnet das Produkt n zweier frei gewählter Primzahlen und veröffentlicht es – die Primzahlen bleiben geheim. Alle Berechnungen im Rahmen des Algorithmus werden in der dadurch bestimmten Gruppe Z_n^*, also „mod n“ durchgeführt.

In der Schlüsselerzeugungsphase wählt Alice eine Zahl $s \in Z_n^*$, die ihr Geheimnis darstellt und bildet daraus $v \in Z_n^*$ durch Quadrierung. Die Zahl v ist der öffentliche Schlüssel von Alice, er wird in der Zentrale hinterlegt und ist dort für jeden abrufbar. Bob kann n und v von der Zentrale erfahren und möchte sich davon überzeugen, dass die Person, mit der er kommuniziert, die Zahl s kennt und folglich Alice ist. Die Kommunikation zwischen beiden läuft in drei Schritten ab:

1. Alice wählt eine Zahl $r \in Z_n^*$ und schickt deren Quadrat $x := r^2 \bmod n$ an Bob.
2. Bob schickt ein Challenge-Bit $e \in 0, 1$ zurück
3. Abhängig von e berechnet Alice die zu sendende Antwort: $y := r \cdot s^e \bmod n$

Jetzt kann Bob überprüfen, ob Alice ihm das korrekte Resultat geschickt hat, indem er y quadriert, denn es muss gelten:

$$y^2 \equiv x \cdot v^e \pmod{n}$$

Eine Betrügerin, die versucht, sich als Alice auszugeben, aber das Geheimnis s nicht kennt, kann nur entweder $r \cdot s$ oder r kennen. Sie muss sich vor Beginn des Protokolls entscheiden, ob sie sich auf ein r festlegen will oder (wenn sie annimmt, dass Bob ihr das Bit $e = 1$ schicken wird) auf ein $r \cdot s$. Im zweiten Fall berechnet sie bei frei gewähltem y die erste zu schickende Zahl als $x := y^2 \cdot v^{-1}$, damit y (als Antwort auf das Challenge-Bit) anschließend obige Formel erfüllt. Da sie sich also im ersten Schritt ähnlich festlegen muss wie durch das Betreten eines Raumes im Szenario mit der Codeschloss-Tür, hat Bob durch seine Wahlmöglichkeit im zweiten Schritt eine 50-prozentige Chance, den Betrug zu entdecken.

3.3 Der diskrete Logarithmus

Ein Grundbaustein vieler kryptographischer Verfahren sind Funktionen, die einfach berechnet werden können, deren Umkehrung aber ungleich schwieriger zu berechnen ist („Einwegfunktionen“). Dazu gehören auch die diskreten Exponentialfuntionen:

> Seien p eine Primzahl und g eine natürliche Zahl mit $g \leq p-1$. Dann ist die *diskrete Exponentialfuntion* zur Basis g definiert durch
>
> $$k \mapsto g^k \bmod p \qquad (1 \leq k \leq p-1)$$
>
> Der diskrete Logarithmus ist die zugehörige Umkehrfunktion.

Beim entsprechenden Protokoll besitzt A das Geheimnis s und $v = g^s$ wird veröffentlicht – s ist also der diskrete Logarithmus von v. Zur Authentisierung gegenüber B gehen beide wie folgt vor:

1. A wählt zufällig eine Zahl r und sendet $x = g^r$ an B
2. B antwortet mit einem Bit b
3. Ist $b = 0$, so sendet A die Zahl $y = r$, andernfalls berechnet sie die Anwort als $y = r + s$.

B kann nun überprüfen, ob $g^y = x$ bzw. $g^y = xv$.

3.4 Isomorphie von Graphen

Das folgende Protokoll basiert darauf, dass es im Allgemeinen schwierig ist, zu entscheiden, ob zwei große Graphen isomorph sind, also ob es eine Permutation der Knoten gibt, die beide ineinander überführt – es handelt sich um ein NP-Problem. A identifiziert sich, indem sie beweist, dass sie den Isomorphismus zwischen zwei isomorphen Graphen kennt. Das Geheimnis von A ist also die Permutation σ, die einen Graphen in den anderen überführt.

In der Schlüsselerzeugungsphase erzeugt A ein Paar von isomorphen Graphen (G_0, G_1), das sie veröffentlicht, und eine Permutation σ. Die Isomorphie der Graphen lässt sich sicherstellen, indem A den Graphen G_0 und die Permutation σ frei wählt und den zweiten Graphen $G_1 = \sigma(G_0)$ setzt.

In einem Protokolldurchgang wählt A zunächst einen der Graphen G_0 und G_1 und wendet eine zufällige Permutation τ darauf an. Das Ergebnis, den Graphen H, sendet A an B. B kann jetzt wählen, ob er den Isomorphismus von H mit G_0 oder mit G_1 bewiesen haben möchte, er tut dies im Form eines Bits. A schickt ihm daraufhin die entsprechende Permutation, also entweder τ, wenn sie zu Beginn den gleichen Graphen wie B gewählt hatte, oder $\tau\sigma^{-1}$ bzw. $\tau\sigma$. B kann die Korrektheit der Antwort von A leicht nachprüfen, da die Anwendung der Permutation auf H genau den gewünschten Graphen erzeugen muss.

3.5 Allgemeine Struktur von Zero-Knowledge-Protokollen

Die Aufteilung des Ablaufs in drei Schritte ist eine Gemeinsamkeit vieler Zero-Knowledge-Protokolle, mit weniger als drei Datentransfers lassen sie sich nicht realisieren. Im ersten Schritt trifft Alice eine zufällige Entscheidung (*Commitment*) und legt sich darauf fest, indem sie Bob einen so genannten Zeugen (*Witness*) ihrer Entscheidung schickt. Mit Hilfe des

Zeugen kann Bob nach dem dritten Schritt des Protokolls feststellen, ob Alice tatsächlich bis zum Ende das zu Anfang gewählte *Commitment* benutzt hat, da der Zeuge aus diesem *Commitment* berechnet wird. Das *Commitment* bewirkt eine Randomisierung zwischen mehreren Protokollabläufen, es ist eine Art Kurzzeit-Ersatzgeheimnis, das am Ende eines Ablaufs von Alice aufgedeckt werden kann, ohne etwas über ihr eigentliches Geheimnis zu verraten. Mit dem Zeugen legt sich Alice auf eine Anzahl von Fragen fest, die sie beantworten zu können vorgibt – nur der Inhaber des Geheimnisses ist in der Lage, wirklich alle Fragen zu beantworten. Alice erlaubt Bob aber nur, genau eine Frage zu stellen (*Challenge*), denn würde sie Bob mehrere Fragen beantworten, könnte er daraus auf das Geheimnis schließen. Zur Berechnung der korrekten Antwort (*Response*) benötigt Alice mit einer gewissen Wahrscheinlichkeit das Geheimnis – und diese Wahrscheinlichkeit kann Bob durch mehrmalige Protokolldurchführung beliebig erhöhen.

Zero-Knowledge-Protokolle vereinen somit die Ideen von Teile-und-wähle-Protokollen (eine faire Möglichkeit, um z.B. ein Stück Kuchen zu teilen: einer halbiert das Stück, der andere wählt einen Teil für sich; im Fall von Zero-Knowledge bietet Alice zwei gleichwertige Informationen an und Bob wählt genau eine, die er mitgeteilt haben möchte) und Challenge-Response-Protokollen. Die Sicherheit des Geheimnisses ist nur gewährleistet, wenn Alice in einem Ablauf immer nur auf eine Frage von Bob antwortet und außerdem darauf achtet, kein *Commitment* (also auch keinen Zeugen) mehr als einmal zu verwenden. Diese auf den ersten Blick störend erscheinende Einschränkung macht die Anwendung des Protokolls in elektronischen Bargeldsystemen interessant, wo man vermeiden möchte, dass eine Person den gleichen Geldschein mehrmals zum Bezahlen einsetzt.

4 Anwendungen

Nach der Theorie sollen jetzt Anwendungen für Zero-Knowledge-Protokolle aufgezeigt werden. Die zwei in diesem Kapitel beschriebenen Anwendungsgebiete sind Smartcards (Abschnitt 4.1) und elektronisches Bargeld (Abschnitt 4.2).

4.1 Smartcards

Eines der derzeit wichtigsten Anwendungsgebiete für Zero-Knowledge-Protokolle sind Smartcards ([MeOV96]). Diese Karten können zum Beispiel als elektronischer Pass oder als ein sicherer Ersatz für die heute gebräuchlichen Kreditkarten benutzt werden. Ihr Vorteil gegenüber einfachen passiven Chipkarten ist, dass sie in der Lage sind, eigenständig Rechenoperationen durchzuführen. Damit lassen sich zum Beispiel kryptographische Operationen implementieren.

4.1.1 Eigenschaften von Smartcards

Smartcards enthalten einen Microchip, der eine Schnittstelle nach außen in Form von elektrischen Kontakten auf der Oberfläche der Chipkarte besitzt. Der Chip ist ein vollständiger Microcomputer, er enthält einen Prozessor, nichtflüchtigen Speicher, einen Bus zwischen beiden und eine I/O-Einheit. Die I/O-Einheit hat keinen direkten Zugriff auf den Speicher, d.h. man kann über die Kontakte auf der Smartcard den Speicherinhalt nicht auslesen – was wichtig ist für die Möglichkeit, kryptographische Funktionen auf dem Chip zu implementieren, da dafür geheime Daten sicher vor einem äußeren Zugriff abgespeichert sein müssen. Über die Schnittstelle lassen sich nur die dafür vorgesehenen Funktionen des Chips aufrufen, und nur über diese können Daten mit dem Chip ausgetauscht werden.

Die Daten im nichtflüchtigen Speicher können nur während des Herstellungsprozesses der Smartcard uneingeschränkt modifiziert werden. Der Speicher enthält das Programm, welches die Funktionsweise des Prozessors bestimmt; im Fall einer kryptographischen Anwendung sind z.B. auch private Schlüssel darin abgespeichert.

4.1.2 Adaption des Fiat-Shamir-Algorithmus für Smartcard-Systeme

Fiat und Shamir beschreiben den Einsatz ihres Verfahrens (siehe Abschnitt 3.2) für die Verwendung in Smartcards (vgl. [FiSh87]). Bei dieser Variante des Algorithmus ist während des Authentisierungsvorgangs keine Kommunikation mit der Zentrale nötig. Außerdem wird die Anzahl der Protokollwiederholungen dadurch reduziert, dass mehrere Geheimnisse parallel überprüft werden können.

Die Zentrale wählt wie oben beschrieben eine Zahl n, außerdem noch eine Funktion $f(x, y)$, die einen String x und eine Konstante y möglichst gleichverteilt und pseudo-zufällig auf Z_n^* abbildet. Die Zahl n und die Funktion f werden veröffentlicht.

Für die Ausstellung einer Smartcard wird von der Zentrale ein Identifikations-Datensatz I erstellt, der alle nötigen Informationen über den künftigen Besitzer der Smartcard enthält. Mit Hilfe der Funktion lassen sich mehrere Werte $v_k := f(I, k)$ für verschiedene k berechnen. Da die Zentrale die Faktorisierung von n kennt, ist es für sie kein Problem, die Quadratwurzeln von $v_k(\bmod n)$ zu berechnen – die Wurzeln (s_k) speichert sie auf der Smartcard, sie stellen das Geheimnis dar, über welches keine Information die Smartcard verlassen darf. Daneben werden auch die bei der Berechnung der s_k verwendeten Konstanten k und der Datensatz I auf der Smartcard abgelegt, diese Daten sind auslesbar – sie stehen also zur Identitäts-Prüfung zur Verfügung. Das Protokoll soll in diesem Fall den Prüfer (B) davon überzeugen, dass die Smartcard wirklich von der Zentrale ausgestellt wurde und folglich die Angaben im Datensatz I korrekt sind.

Zum Überprüfen der Echtheit der Smartcard (die hier die Rolle von A übernimmt) berechnet B zuerst die v_k aus den ihm verfügbaren Informationen n, f, I und k. Nachdem A dann ein r gewählt und $x := r^2$ an B geschickt hat, sendet B einen binären Vektor mit den Challenge-Bits. A muss jetzt für jedes v_k die entsprechende Antwort $y := r \cdot s_k^e \bmod n$ schicken und B kann alle Antworten parallel überprüfen.

4.2 Electronic Cash

Durch die massive Ausbreitung des Internet hat auch die Bedeutung der elektronischen Bezahlung stark zugenommen. Es gibt jedoch noch immer kein allgemein akzeptiertes, sicheres System, in dem Geld nur durch Daten auf einer Festplatte repräsentiert wird und welches bis auf die Möglichkeit des Transfers über Rechnernetze die gleichen Eigenschaften wie Bargeld besitzt.

Bei den bisher genutzten Systemen muss der Benutzer starke Einschränkungen in Kauf nehmen – dass dies nicht so sein müsste, zeigen zum Beispiel Tatsuaki Okamoto und Kazuo Ohta in ihrem Entwurf des ihrer Meinung nach ersten idealen elektronischen Bezahlungssystems (siehe [OkOh92]). Die notwendige Sicherheit wird dabei durch kryptographische Protokolle garantiert, wobei auch Zero-Knowledge-Verfahren zum Einsatz kommen.

4.2.1 Anforderungen und Eigenschaften

Okamoto und Ohta beschreiben zunächst sechs Eigenschaften eines idealen Systems für digitales Geld:

- Unabhängigkeit: Das digitale Geld ist nicht abhängig von einem physikalischen Aufenthaltsort; es kann über Rechnernetze transferiert werden
- Sicherheit: Es ist nicht möglich, digitales Geld zu kopieren und mehrfach zu verwenden
- Privatsphäre: Die Identität eines Benutzers bleibt verborgen; niemand kann seine Geschäfte nachvollziehen.
- Offline-Bezahlung: Während eines Bezahlungs-Vorgangs mit digitalem Geld ist keine Verbindung zu einem Zentralrechner notwendig.
- Transferierbarkeit: Mit dem Geld kann nicht nur bei einem Händler bezahlt werden, sondern es ist auch auf andere Benutzer übertragbar, die es dann weiterverwenden können.
- Teilbarkeit: Eine Einheit digitalen Geldes kann in mehrere Einheiten kleinerer Beträge aufgespalten werden. Der Gesamtwert bleibt dabei natürlich unverändert.

Für die Praxis bedeutsam ist auch die Kosteneffektivität – ein System für digitales Geld darf keine höheren Kosten verursachen als die Zahlung mit Bargeld.

4.2.2 UEC – Universal Electronic Cash

Das von Okamoto und Ohta entworfene elektronische Bezahlungssystem („Universal Electronic Cash“) benutzt verschiedene kryptographische Protokolle, um Authentizität des Geldes und Anonymität des Bezahlvorgangs sicherzustellen. Das gesamte System ist relativ komplex, so dass es an dieser Stelle nicht komplett beschrieben werden kann; umfassende Information darüber findet sich in [OkOh92].

Das System ist für Offline-Transaktionen ausgelegt, es kann also beim Bezahlen mit einem elektronischen Geldschein nicht festgestellt werden, ob mit dem Schein schon einmal bezahlt wurde – ob er also doppelt ausgegeben wurde. Dies wird erst beim Einlösen des Geldscheins bei der ausstellenden Bank erkannt, diese muss dann in der Lage sein, die Identität des Käufers zu ermitteln. UEC setzt dazu ein spezielles Zero-Knowledge-Protokoll ein – das Verfahren wird „Secret-Splitting“ genannt, es ist in stark vereinfachter Form in Bild 5 dargestellt. Das Geheimnis kennt nur der Geldschein-Besitzer, der sich den Geldschein von der Bank ausstellen ließ, es enthält unter anderem Identität. Während des Protokollablaufs wird geprüft, ob der Käufer (Geldschein-Benutzer) dieses Geheimnis kennt und folglich der rechtmäßige Besitzer des Geldes ist. In den Daten des Geldscheins sind k *Commitments* enthalten (wobei k eine Systemkonstante ist), die im ersten Protokollschritt übermittelt werden. Da der Geldschein von der Bank signiert wurde, kann der Besitzer des Scheins diese Daten nicht unmerklich manipulieren, die *Commitments* sind fest mit dem Geldschein verbunden. Bei einem Bezahlvorgang wird der Käufer nach einer Überprüfung des Geldscheins auf Gültigkeit (Signatur der Bank) durch den Händler im zweiten Protokollschritt von diesem (durch das Senden mehrerer *Challenge*-Bits) aufgefordert, zu beweisen, dass er die korrekte *Response* zu jedem *Commitment* und *Challenge*-Bit kennt. Er kann die richtigen Antworten nur berechnen, wenn er wirklich die Person ist, für die der Geldschein ausgestellt wurde, dies stellt das Zero-Knowledge-Protokoll sicher. Gleichzeitig sorgt eine nützlich eingesetzte Eigenschaft des Protokolls dafür, dass der Käufer mit hoher Wahrscheinlichkeit von der Bank identifiziert werden kann, wenn er den Geldschein zweimal ausgeben sollte. Wie in Abschnitt 3.5 erwähnt, sollte man nämlich nie zweimal das gleiche *Commitment* benutzen, da sonst der Protokollpartner das Geheimnis berechnen kann (zumindest, wenn seine *Challenge*-Bits bei beiden Abläufen unterschiedlich waren). Der Besitzer des Geldscheins kann die *Commitments*

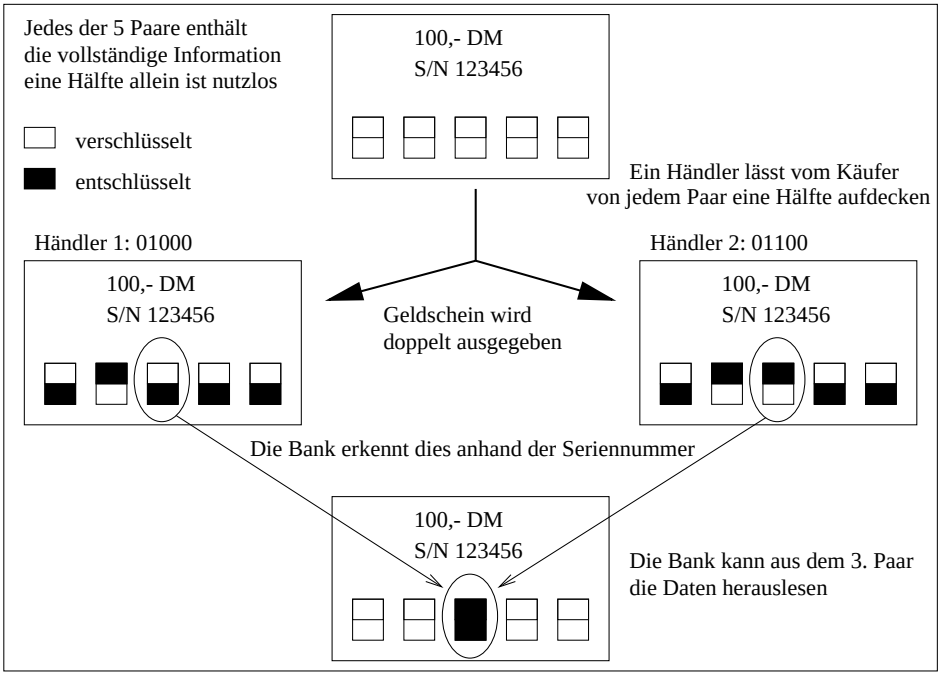

Abbildung 5: „Secret-Splitting“ offenbart Identität des Benutzers bei doppeltem Ausgeben eines Geldscheins

aber nicht ändern, da sonst der Geldschein ungültig würde. Er könnte mit seinem Betrug nur in dem unwahrscheinlichen Fall Erfolg haben, dass beide Händler, bei denen er bezahlt, die gleichen *Challenge*-Bits wählen – aber die Wahrscheinlichkeit dafür ist lediglich 2^{-k}, was bei einem groß genug gewählten k (etwa $k = 100$) um Größenordnungen unwahrscheinlicher wäre als ein Lotto-Hauptgewinn. In den meisten Fällen kann also die Bank, die ja bei Einlösung der beiden Geldscheine alle nötigen Informationen erhält, den Betrüger zur Rechenschaft ziehen.

UEC erfüllt alle Anforderungen an ein elektronisches Zahlungssystem, es ist jedoch noch nicht in die Praxis umgesetzt worden. Inzwischen wurden andere Verfahren für die Realisierung von elektronischen Bargeld entwickelt (z.B. E-Cash), die zwar nicht alle Kriterien des „idealen“ elektronischen Zahlungssystems erfüllen, dafür aber bereits eingesetzt werden. Informationen zur Praxistauglichkeit der Systeme finden sich in [Benn97].

5 Zusammenfassung

Wie gezeigt wurde, ermöglichen Zero-Knowledge-Protokolle die Authentisierung, ohne dass dabei geheim zu haltende Daten zwischen den Kommunikationspartnern ausgetauscht werden müssen – ein möglicher Angreifer hat also keine Möglichkeit, abgehörte Daten zum Identitätsbetrug zu verwenden. Allerdings sind die Protokolle auch nur so sicher wie die Komplexität der verwendeten Probleme – wenn sich das Geheimnis zu einfach aus bekannten Daten berechnen lässt, ist das Protokoll wirkungslos. Ebenso muss darauf geachtet werden, dass die zu authentisierende Person das Protokoll nie zweimal gleich beginnt (mit dem gleichen *Commitment*), da sie sonst ihr Geheimnis gefährdet.

Ein paar interessante Eigenschaften der für die Anwendungen angepassten Protokolle sollten noch erwähnt werden: Die Variante für Smartcards hat die praktische Eigenschaft, dass zur Authentisierung einer Person alle für diese Person spezifischen Daten (also die Identität und die v_k) von der Person selbst kommen – es muss also auch dann keine Verbindung zur Zentrale aufgebaut werden, wenn die Person noch nie authentisiert wurde und unbekannt ist; es

genügt, n und f von der Zentrale zu kennen, und diese sind personenunabhängig. Bei UEC erfüllt das Protokoll gleich zwei Funktionen: es verhindert zum einen, dass ein Geldschein gestohlen werden kann (außer das Geheimnis des Benutzers wird vom Dieb in Erfahrung gebracht), und es sorgt für die Erkennung eines Betrügers, der doppelt mit einem Geldschein bezahlt hat – obwohl der Käufer (wie bei echtem Bargeld) bei einer rechtmäßigen Transaktion gegenüber allen Teilnehmern außer dem direkten Handelspartner (also auch gegenüber der Bank) anonym bleibt.

Literatur

[Benn97] Andreas Benne. Zahlungen im Internet. Seminararbeit, Universität Saarbrücken, 1997. http://www.stud.uni-saarland.de/∼anbe/seminar/inhalt.htm.

[BeSW98] Albrecht Beutelspacher, Jörg Schwenk und Klaus-Dieter Wolfenstetter. *Moderne Verfahren der Kryptographie. Von RSA zu Zero-Knowledge*. Vieweg. 1998.

[FiSh87] Amos Fiat und Adi Shamir. How to proove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO '86 Proceedings*. Springer, 1987, S. 186–194.

[GuUQ92] Louis Claude Guillou, Michel Ugon und Jean-Jaques Quisquater. *Contemporary Cryptology. The Science of Information Integry*, Kapitel 12: The Smart Card, S. 561–614. IEEE Press. 1992.

[MeOV96] A. Menezes, P. van Oorschot und S. Vanstone. *Handbook of Applied Cryptography*, Kapitel 10: Identification and Entity Authentication, S. 385–424. CRC Press. http://cacr.math.uwaterloo.ca/hac, 1996.

[OkOh92] T. Okamoto und K. Ohta. Universal Electronic Cash. In *Advances in Cryptology – CRYPTO '91 Proceedings*. Springer, 1992, S. 324–337.

[QUGU90] Jean-Jaques, Myriam, Muriel & Michaël QUISQUATER und Louis, Marie Annick, Gaïd, Anna, Gwenolé & Soazig GUILLOU. How to Explain Zero-Knowledge Protocols to Your Children. In *Advances in Cryptology – CRYPTO '89 Proceedings*. Springer, 1990, S. 628–631.

Neue Vorschläge für Multicast-Routing

Lars Kühn

Kurzfassung

Der Bedarf an Multicast-Anwendungen wird immer größer. Die heute im Einsatz befindlichen Multicast-Protokolle haben dabei noch einige Einschränkungen, besonders wenn es darum geht kostengünstig kleinere Gruppen mit weit verteilten Mitgliedern zu bilden. Dazu werden im folgenden zwei neue Ansätze vorgestellt, welche eben dies ermöglichen. Dabei geht es zum einen darum ein Protokoll namens REUNITE zu implementieren, welches es erlaubt, Multicast über Unicast-Bäume zu betreiben. Dabei wird gänzlich auf den Einsatz von Klasse-D-IP-Adressen verzichtet. Dieses Protokoll wird in für das Multicasting relevanten Routern implementiert. Im Gegensatz dazu wird im zweiten Ansatz die Möglichkeit der Verlagerung der Multicast-Funktionalität in die Endsysteme betrachtet. Beim Endsystem-Multicast wird also ausschließlich ein Netz aus Unicast-Routern benötigt, über welches die Endsysteme ein von ihnen verwaltetes Multicast-Netz spannen. Der große Vorteil dieser Protokolle ist die Möglichkeit auf bereits bestehende Unicast-Routing-Netze aufzubauen, und diese für die Kommunikation zwischen den zu implementierenden Protokollen zu nutze. Dadurch wird eine einfache und kostengünstige Möglichkeit geschaffen, Multicasting schrittweise für einzelne Gruppen zu implementieren.

1 Einleitung

Multicasting ist die Fähigkeit eines Kommunikationsnetzwerkes, die Nachricht eines Senders in Kopie an mehrere Empfänger auszuliefern. Um diese Multicastfähigkeit in Netzwerken zu realisieren, gibt es verschiedene Ansätze des Multicast-Routing mit der Zielsetzung, die zur Verfügung stehenden Netzressourcen minimal zu belasten. Stellen wir uns beispielsweise vor, ein Videoserver würde einen Film an 1000 Empfänger übertragen, und diese 1000 Verbindungen würden als Ende zu Ende Verbindungen vom Videoserver zu jedem einzelnen Empfänger gehen, so müsste das Video 1000 mal über die kompletten Teilstrecken des Netzes übertragen werden. Um die verfügbare Bandbreite besser auszunutzen, ist der Grundgedanke des Multicast-Routing, das Video vom Server nur einmal zu bestimmten Multicast-Routern zu übertragen, welche an Knotenpunkten des Netzes das Video auf jene Teilstrecken weiterleiten, an denen ein Empfänger des Videos angeschlossen ist. Dadurch werden die Daten auf jeder Teilstrecke nur einmal übertragen.

Es existieren heute sehr viele Multicast-Anwendungen, wie Videokonferenz, Gruppennachrichten, interaktive Spiele, usw. , jedoch ist die Implementierung nicht immer effizient, da die heutigen WANs überwiegend darauf ausgelegt sind, Ende-zu-Ende-Verbindungen (Unicast) zu unterstützen. Da zukünftig jedoch immer mehr bandbreitenintensive Multicast-Anwendungen zum Einsatz kommen werden, ist eine effiziente Unterstützung von Multicasting in WANs dringend notwendig. Ein WAN besteht aus vielen Knoten (z.B. Switches oder Routern), die über Verbindungen miteinander kommunizieren. Eine Verbindung zwischen zwei Kommunikationseinheiten wird durch die Knoten dieses Kommunikationsnetzes geroutet. Die Verbindungen zwischen den einzelnen Knoten können jetzt verschiedene Eigenschaften besitzen, was beispielsweise ihre Bandbreite oder die Laufzeit betrifft. Bei Unicast-Verbindungen wird das

Routing oft als Shortest-path-Problem in Graphen behandelt, d.h. der Weg mit der geringsten Laufzeit wird ausgewählt. Beim Multicasting sendet jedoch ein Sender an mehrere Empfänger, und es ist deshalb der Minimum Weight Tree für die Auswahl der besten Verbindungen von Interesse, das heißt, die Summe aller Gewichte (z.B. Laufzeit) soll minimal sein.

Bei kleineren Gruppen, deren Mitglieder verteilt über ein großes Netzwerk (z.B. das Internet) eine Multicast-Anwendung nutzen, entsteht oft ein ähnlich geformter Baum wie in Abbildung 1 gezeigt.

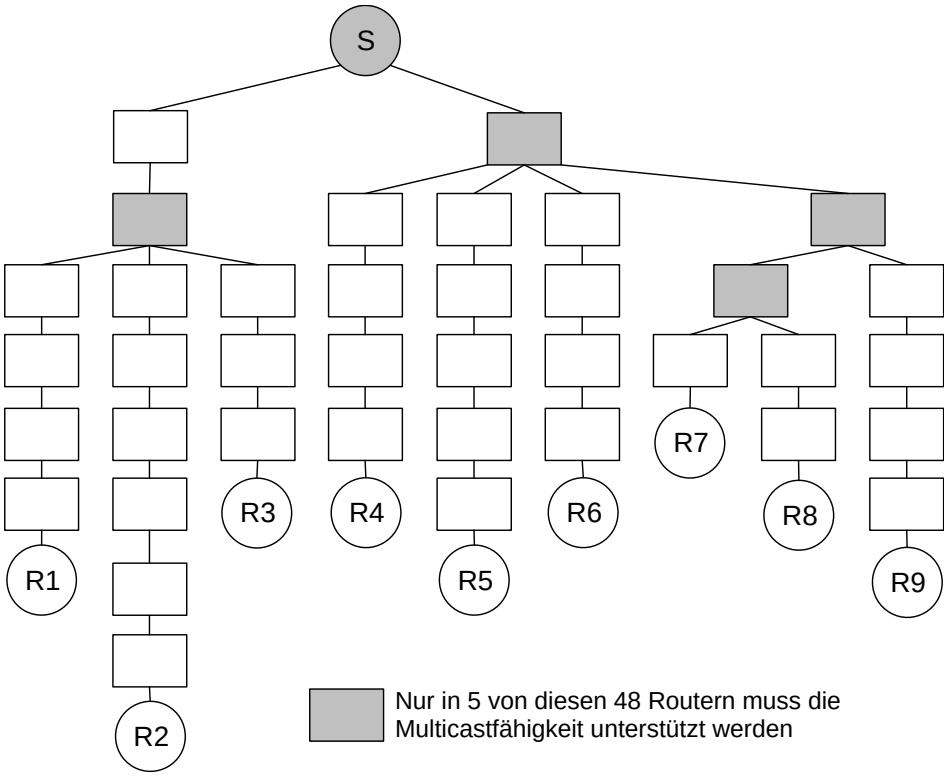

Abbildung 1: Beispiel eines Multicast-Baumes für eine verteilte, kleine Gruppe

Hierbei fällt besonders auf, dass nur wenige (in diesem Beispiel 5 von 48) Router tatsächlich das Multicasting unterstützen müssen. Durch Platzierung von Multicast-fähigen Routern mit einem entsprechenden Protokoll an den „wichtigen“ Knotenpunkten, könnte dadurch Multicasting im ganzen Netz ermöglicht werden (wenn sicherlich die Frage der Wichtigkeit von Knoten nicht einfach zu klären ist). Diese und weitere Möglichkeiten versuchen neue Multicast-Routing-Protokolle zu schaffen. Zunächst werden jedoch die wichtigsten der bestehenden Protokolle vorgestellt.

2 Bestehende Multicast-Routing-Protokolle

Die Implementierung von Multicasting im Internet wird durch den Einsatz von drei verschiedenen Protokolltypen erreicht. Der erste Protokolltyp wird bei den Endsystemen dafür eingesetzt, einer Gruppe beizutreten und sie zu verlassen. Ein Beispiel hierfür ist das Internet Group Management Protocol (IGMP)[S.E.94]. Der zweite Protokolltyp ist das Multicast Interior Gateway Protocol (MIGP), welches bei den Multicast-Routern eingesetzt wird, um die Multicast-Kommunikation innerhalb eines autonomen Systems (einem Routernetzwerk, dessen Verwaltung das entsprechende Multicast-Routing-Protokoll auf allen Routern dieser Domäne implementiert) zu ermöglichen. Beispiele hierfür sind DVMRP[WaPD88], Core-Base

Tree (CBT)[Ball97] und Protocol-Independent Multicast (PIM)[SEFJ94]. Der dritte Typ von Protokollen wird für die Kommunikation zwischen den autonomen Systemen eingesetzt. Ein Beispiel ist das Border Gateway Multicast Protocol (BGMP)[ThEM98]. Das Zusammenspiel der drei Protokolltypen wird in Abbildung 2 dargestellt.

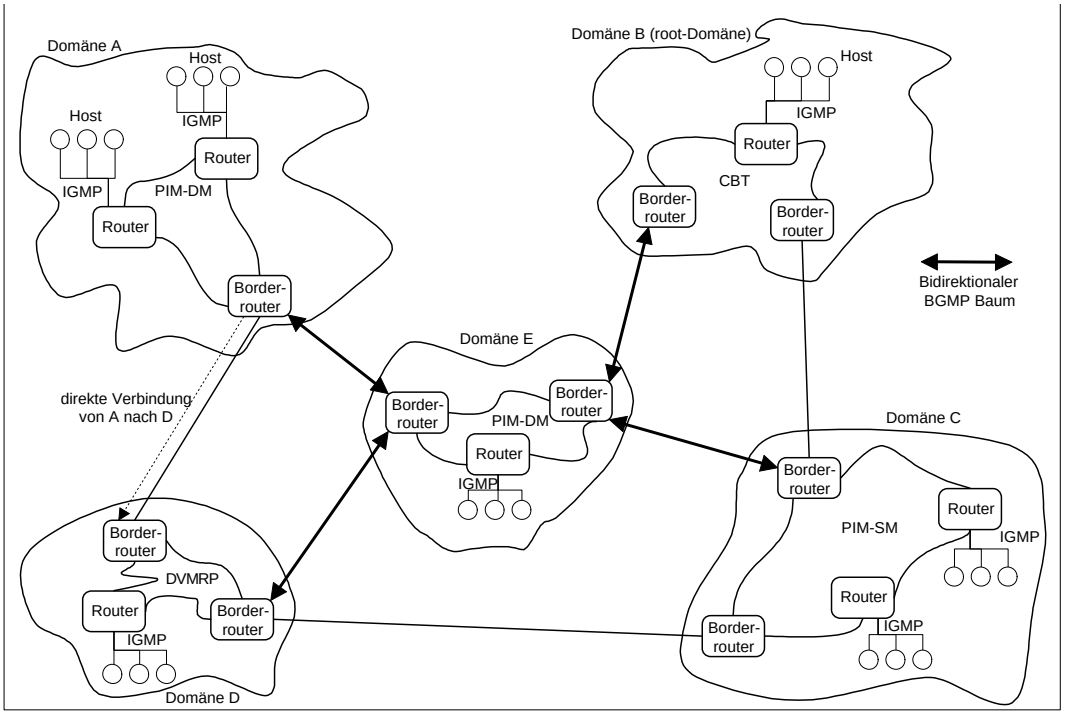

Abbildung 2: Zusammenspiel der Multicast-Protokolle im Internet

2.1 Internet Group Management Protocol (IGMP)

IGMP setzt über IP auf und ermöglicht die Unterstützung von Multicasting. IGMP wird zwischen einem Endsystem und seinem unmittelbar benachbarten Multicast-Router eingesetzt. Bei der Kommunikation unterscheidet man zwei Datenformate. Zum einen die Query-Pakete, die vom Multicast-Router an eine permanente Gruppe (All Hosts Group) versandt werden, um die Gruppenzugehörigkeit der Endsysteme abzufragen, und zum anderen die Report-Pakete, über welche die Endsysteme auf die Anfrage antworten. Ein Endsystem kann jedoch auch ohne eine Anfrage des Routers abzuwarten, ein „join-group“ Paket zum Beitreten zu der Gruppe, oder ein „leave-group“ Paket zum Verlassen der Gruppe an den Multicast-Router senden.

2.2 Multicast Interior Gateway Protocol (MIGP)

2.2.1 Reverse-Path Multicast (RPM)

Das RPM ermöglicht das Multicast-Routing über ein Netzwerk von Routern. Bei RPM wird eine Pruning-Technik verwendet, um anzuzeigen, ob Pakete über einen Link weitergeleitet oder nicht weitergeleitet werden müssen. Die Pakete werden dabei zunächst nach dem Truncated-Reverse-Path-Broadcast-Verfahren (TRPB) gesendet. Hierbei bestehen zwischen Router und Link eine Vater-Sohn-Beziehung, wobei derjenige Router der Vater des Links ist, welcher die kürzeste Entfernung zum Sender besitzt (bei Gleichheit der mit der kleineren

Adresse). Dadurch wird erreicht, dass keine Informationen doppelt über einen Link übertragen werden. Dieses Verfahren erfordert jedoch, dass die Router die entsprechenden Informationen wie die Entfernung zum Sender über einen Link mit den benachbarten Routern austauschen. Erhält nun also ein Router ein Packet, an dessen Links keine Gruppenmitglieder angeschlossen sind, so sendet er an den nächsten Router Richtung Quelle eine spezielle Nachricht (Non Membership Report) für diese spezielle Gruppe und diese spezielle Quelle. Um die Wiederaufnahme eines Gruppenmitgliedes zu erreichen, werden ebenfalls Kontrollnachrichten gesendet, welche das Blockieren von Links wieder aufheben.

2.2.2 Distance-Vector Multicast Routing Protocol (DVMRP)

DVMRP ist ein Multicast-Routing-Protokoll, welches RPM einsetzt, um Multicast-Pakete zu versenden. Bei DVMRP werden jeder Verbindung ein Kostenfaktor *metric* und ein *treshold* zugeordnet. Die *metric* gibt die Routing-Kosten für diese Verbindung an, und wird zur Erstellung des Reverse Shortest Path Tree (RSPT) verwendet. Das *treshold* ist die Minimum Time-To-Live (TTL), die ein Multicast-Paket benötigt, um auf dem gegebenen Link weitergeleitet zu werden. Ein Paket wird also nur dann weitergeleitet, wenn der *treshold* der Verbindung den im Paket enthaltenen Time-To-Live (TTL) nicht übersteigt. Der TTL-Wert der weitergeleiteten Pakete wird dann um den Kostenfaktor der Verbindung erniedrigt. Dadurch kann der *treshold* dazu benutzt werden, die geografische Ausbreitung einer Multicast-Übertragung zu begrenzen. Die Multicast-Router bei DVMRP tauschen in regelmäßigen Abständen Informationen über „Ziel“ und „Kosten zu diesem Ziel“ mit ihren direkten Nachbarn aus. Basierend auf den Informationen seiner Nachbarn, erstellt ein Router seine Multicast-Routing-Tabelle. Abbildung 3 zeigt ein Beispiel einer solchen Routing-Tabelle.

Source	Subnet mask	From gateway	Metric	Status	TTL
128.1.0.0	255.255.0.0	128.9.5.2	4	Up	200
128.5.0.0	255.255.0.0	128.9.5.2	3	Up	150
128.7.0.0	255.255.0.0	128.6.3.1	5	Up	200

Abbildung 3: Beispiel einer DVMRP-Routing-Tabelle

2.2.3 Core-Based Tree (CBT)

Für sehr große Netzwerke, mit vielen gleichzeitig aktiven Multicast-Gruppen, ist DVMRP nicht besonders geeignet. Wenn in einem Netz mit Tausenden von Knoten nur wenige zur Multicast-Gruppe gehören, werden unnötig viele Broadcast-Initialisierungspakete gesendet, und zudem muss jeder Router die Information (Quelle, Gruppe) speichern, was schnell sehr viel werden kann. Bei CBT wird nur ein Auslieferungsbaum pro Gruppe erstellt, über den alle Sender und Empfänger dieser Gruppe kommunizieren. Dadurch wird die Größe der Multicast-Routing-Tabellen der einzelnen Router deutlich reduziert, da nur noch ein Eintrag pro Gruppe vorgehalten werden muss. Den Wurzelknoten dieses Multicast-Baumes nennt man „Core“. Möchte nun ein Router der Gruppe beitreten sendet er einfach eine JOIN/REQUEST Nachricht an den Core-Router, welcher dann ein Update der Routing-Tabelle an alle Gruppenmitglieder sendet.

2.2.4 Protocol-Independent Multicast (PIM)

Da, wie schon erwähnt, ein CBT Ansatz eher für viele Sender mit geringer Datenrate geeignet ist, da hier eine Konzentration des Verkehrs auf den einen Baum stattfindet, und ein Shortest-Path-Routing für wenige Sender mit höher Datenrate sinnvoll erscheint, bietet es sich an einen flexiblen Ansatz zu wählen, welcher sich für das entsprechende Einsatzszenario anpassen lässt. PIM kann entweder Bäume nach dem SPT-Verfahren oder aber gemeinsam genutzte Bäume erstellen. In PIM sind zwei verschiedene Arbeitsmodi vorgesehen, zum einen PIM-Dense Mode (PIM-DM), ein DVMRP-ähnliches Verfahren, und zum anderen PIM-Sparse Mode (PIM-SM), welches unidirektionale gemeinsam genutzte Bäume erstellt. Das Kreieren von PIM-DM-Multicast-Bäumen wird durch den Datenfluss gesteuert, d.h. dass es erst ab einer gewissen Datenrate eingesetzt wird. In PIM-SM wird für jede Gruppe ein sogenannter Rendezvous-Punkt (RP) bestimmt. Hier melden sich die Sender an, und die Empfänger erfahren die Existenz der verschiedenen Gruppen. Ein Host oder ein Router der einer Gruppe beitreten will, sendet ein JOIN-Paket an den Rendezvous-Punkt, wodurch er in den Multicast-Baum aufgenommen wird.

2.2.5 Simple Multicast

Bei Simple Multicast wird zur global eindeutigen Identifizierung einer Gruppe ein Tupel aus (Adresse des Core-Routers, Multicast-Adresse) verwendet. Das Simple-Multicast-Protokoll erstellt dabei einen bidirektionalen gemeinsam genutzten Baum (was auch bei CBT und BGMP angewendet wird), welcher auf einen zentralen Knoten geroutet wird. Somit entfällt das Verwalten global eindeutiger Multicast-(Klasse D)-Adressen.

2.3 Border Gateway Multicast Protocol (BGMP)

Um nun verschiedene Domänen untereinander zu verbinden, wird das BGMP in den Grenz-Routern implementiert (zusätzlich zu dem entsprechenden MIGP). BGMP erstellt nun für diese Grenz-Router einen bidirektionalen gemeinsam genutzten Baum, dessen Wurzel die so genannte „Root-Domäne“ ist. Auch hier können die Router durch „JOIN-“ und „PRUNE-Nachrichten“der Gruppe beitreten oder sie wieder verlassen. Die Grenz-Router nutzen für den Austausch ihrer Nachrichten untereinander eine TCP-Verbindung. Da die Verbindung über diesen gemeinsamen Baum jedoch nicht immer die kürzeste Verbindung zu einem Ziel sein muss, haben die Grenz-Router, welche zueinander eine kürzere „direkte Verbindung“ haben die Möglichkeit, Pakete über diese „direkte Verbindung“ zu senden (siehe Abbildung 2).

2.4 Probleme von IP-Multicasting

- Die effektive Nutzung von IP-Multicasting im Internet erfordert eine flächendeckende Implementierung der Protokolle.
- Router die kein IP-Multicast unterstützen, müssen es ermöglichen, Verbindung zwischen Multicast-Routern herzustellen (z.B. durch Tunnel).
- Die eindeutige Identifizierung der Gruppe muss gewährleistet sein. Dies ist in großen Netzen (z.B. Internet), bei alleiniger Nutzung von Klasse D-IP-Adressen mit einem hohen Verwaltungsaufwand verbunden. Hierbei ist auch die begrenzte Anzahl der Klasse D-IP-Adressen zu beachten.
- Die heute im Einsatz befindlichen Technologien (z.B. DVMRP) im MBone, skalieren nur sehr schlecht hinsichtlich großer Teilnehmerzahlen.

3 REUNITE: A Recursive Unicast Approach to Multicast

3.1 Die Grundidee von REUNITE (REcursive UNIcast TreE)

REUNITE[StNZ00] ist ein Multicast-Protokoll welches im März diesen Jahres von I. Stoica, T. Eugene Ng, und H. Zhang veröffentlicht wurde. Die Grundidee von REUNITE ist es, Multicast-Dienste mit Hilfe rekursiver Unicast-Bäume zu verwirklichen. REUNITE benutzt dabei keine Klasse D-IP- Adressen. Stattdessen wird sowohl die Gruppenidentifikation als auch das Weiterleiten der Daten durch Unicast-IP-Adressen realisiert. Dabei müssen nur Router die als „Knoten“ in dem Multicast-Baum für eine Gruppe agieren, wirklich ein Multicasting der Datenpakete vornehmen, alle anderen Router, welche auf der Verbindungsstrecke zwischen diesen Knoten liegen, müssen die Datenpakete einfach nur durch das übliche Unicast-Routing weiterleiten. Dadurch ist es möglich REUNITE auch nur in einem Teilnetz zu implementieren.

3.2 Die Arbeitsweise von REUNITE

Wie schon erwähnt, nutzt REUNITE im Gegensatz zu allen heute im Einsatz befindlichen IP-Multicast-Protokollen, keine Klasse D-IP-Adressen. Eine Gruppe wird durch ein Tupel aus <root IP adress, root port number> gekennzeichnet, wobei der root-Knoten entweder der Sender oder ein spezieller Knoten sein kann. Dadurch wird die Festlegung global eindeutiger Gruppen-IDs sehr einfach, da nur der Wurzel-Knoten für eine lokal eindeutige Portnummer zu sorgen hat. Für jede Gruppe erstellt REUNITE einen Auslieferungsbaum, ausgehend von einer Wurzel dem root-Knoten. Jeder verzweigende Knoten des Baumes enthält eine Liste mit Empfängeradressen, die Multicast-Forwarding-Tabelle (MFT). Man sagt also, dass ein Empfänger R in Knoten N an den Multicast-Baum anknüpft, wenn die Adresse von R in der Liste von Knoten N eingetragen ist. In REUNITE wird eine Empfängeradresse in genau einem Knoten eines Auslieferungsbaumes einer Gruppe eingetragen. Um ein Multicast-Paket zu versenden, schickt der root-Knoten eine Kopie des Paketes an jeden Empfänger auf seiner Liste. Diese Prozedur wird in jedem verzweigenden Knoten wiederholt, bis das Paket jeden Empfänger dieser Gruppe erreicht hat.

Nachfolgend wird in Abbildung 4 eine Multicast-Gruppe mit vier Empfängern gezeigt. Knoten S ist der root Knoten. R1 knüpft an S, R3 an N3 und R2 an N4 an. Festzustellen ist hier, dass nur die Knoten N3 und N4 verzweigende Knoten sind. Die Liste der Empfänger ist als letzter Eintrag in der zugehörigen Tabelle zu sehen. Wenn also S ein Multicast-Paket versendet, sendet er einfach das Paket an alle Empfänger in seiner Liste, hier also an R1. Wenn N3 das Paket weiterleitet, sendet er zusätzlich eine Kopie an alle Empfänger auf seiner Liste, hier also R3, und N4 sendet eine Kopie an R2. Die Knoten N1 und N2 können die Multicast-Pakete in diesem Fall also einfach wie Unicast-Pakete behandeln, und anhand ihrer dafür vorhandenen Weiterleitungstabellen weiterleiten.

Der MFT enthält für jede Multicast-Gruppe, welche an dem dazugehörigen Knoten verzweigt, einen Eintrag mit folgendem Format:

<root addr, root port>; <dst,stale>; <(rcv_1,$alive_1$), ..., (rcv_n,$alive_n$)>.

Hierbei kennzeichnet <root addr, root port> die Gruppe; dst ist die IP Adresse desjenigen Empfängers, der zuerst der Gruppe beigetreten ist; rev_i, i=1,..,n ist die Empfängerliste, bestehend aus IP-Adressen, an die der Router eine Kopie des Multicast-Paketes für die entsprechende Gruppe sendet; stale und alive sind boolesche Variablen, welche den Zustand des Zielknotens beschreiben. Im MFT des root-Knotens wird kein dst eingetragen.

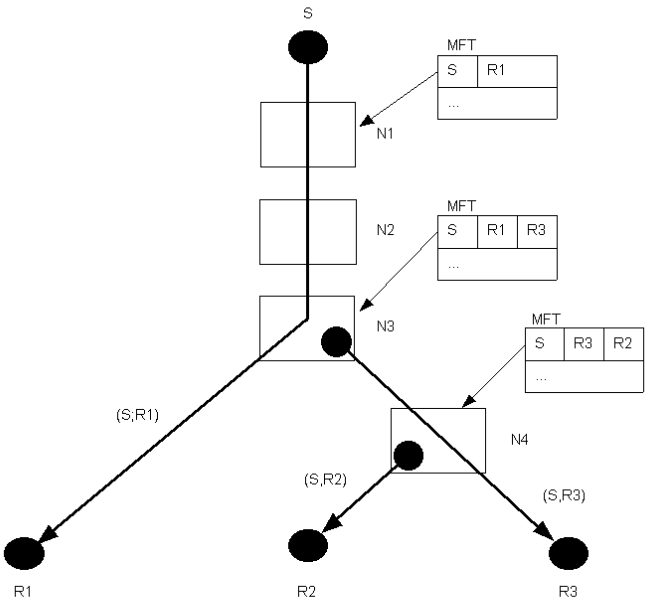

Abbildung 4: Beispiel eines REUNITE-Forwarding-Baumes

Neben dem bereits besprochenen MFT, gibt es noch eine zweite Tabelle in REUNITE, den MCT (Multicast Control Table), in dem für jede Multicast-Route, die durch diesen Knoten geht, jedoch nicht verzweigt, ein Eintrag der Form <root addr, root port>, <dst> vorgenommen wird. Dies bedeutet also, dass sobald ein REUNITE-Router zu einem Auslieferungsbaum einer Multicast-Gruppe gehört, dieser einen Eintrag entweder in den MFT oder den MCT macht. Hierbei ist zu beachten, dass für die Weiterleitung eines Paketes nur die Einträge in der Tabelle MFT einzusehen sind. MCT wird nur benötigt, wenn Kontrollnachrichten verarbeitet werden.

REUNITE verwendet zwei Typen von Kontrollnachrichten: JOIN-Nachrichten, welche periodisch über Unicast vom Empfänger zum root gesendet werden, und TREE-Nachrichten, welche ebenfalls periodisch vom root über Multicast an die in der MFT des root eingetragenen Empfänger gesendet werden. In beiden Fällen sollte die Periodendauer unter TO1 Sekunden liegen. Dabei werden die JOIN-Nachrichten zum Erstellen und Auffrischen der Empfängereinträge in der MFT, und die TREE-Nachrichten zum Erstellen der MCT, und zum Auffrischen der MCT- und MFT-Einträge verwendet.

Möchte ein Empfänger R1 also einer Gruppe beitreten, sendet er ein JOIN welches, wenn dieser Router das erste Gruppenmitglied ist, bis zum Sender S läuft (Abbildung 5(a)). Wenn S den JOIN erhält, macht er einen Eintrag in seine MFT, was bedeutet, dass R1 an den Multicast-Baum in S anknüpft. Nun beginnt S Datenpakete an R1 zu senden. Zusätzlich sendet S aber auch in periodischen Abständen TREE-Nachrichten entlang des Baumes. Alle auf der Route zwischen S und R1 befindlichen Router aktualisieren nun ihre MCT (Abbildung 5(b)). Möchte nun ein weiterer Empfänger R2 der Gruppe beitreten, so sendet auch er ein JOIN. Trifft nun dieses JOIN auf dem Weg zu S auf einen Router N3 der bereits Teil des Auslieferungsbaumes dieser Gruppe ist, so wird der JOIN an diesem Knoten durchgeführt, d.h. der Router löscht den MCT-Eintrag für diese Gruppe, und fügt einen Eintrag für R2 in

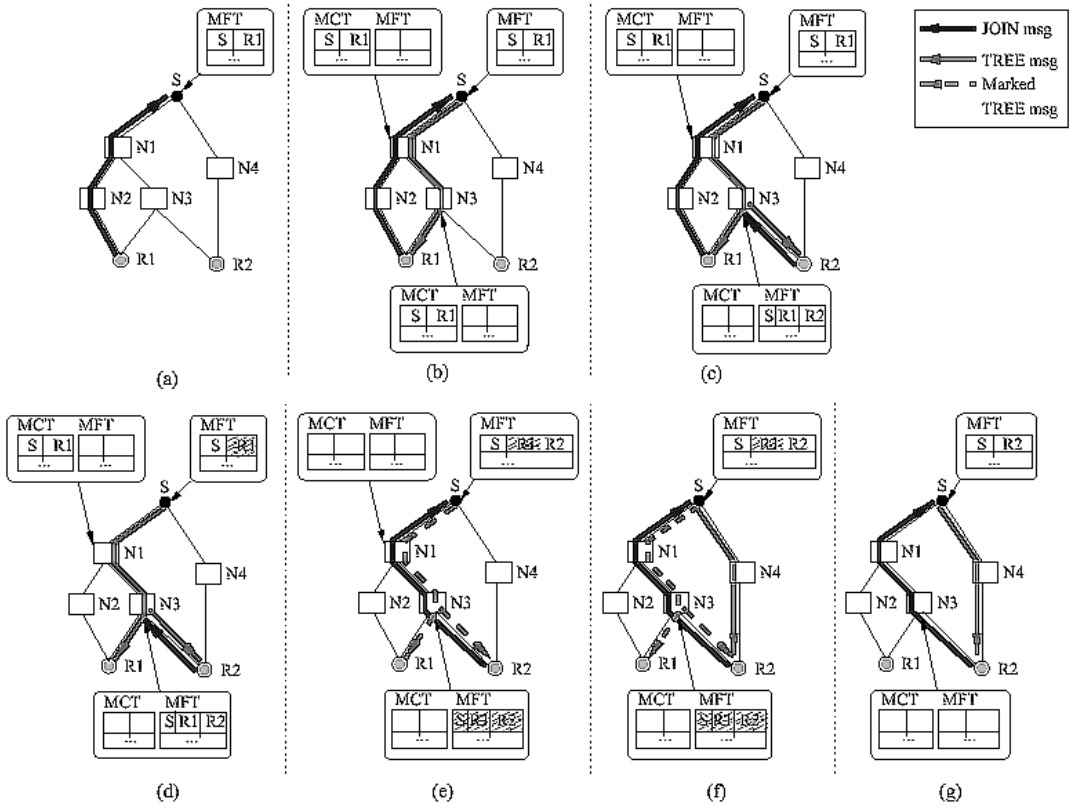

Abbildung 5: Erstellen eines REUNITE-Multicast-Baumes

seine MFT ein (Abbildung 5(c)). Von nun an sendet N3 jeweils eine Kopie der Datenpakete, wie auch der TREE-Pakete an R2. Jeder der Empfänger sendet weiterhin in periodischen Abständen JOIN-Pakete in Richtung Sender, welche jedoch von dem Knoten, in dessen MFT der Empfänger eingetragen ist, nicht weitergeleitet werden. Möchte jetzt ein Empfänger die Gruppe verlassen, so hört er einfach auf seine JOIN-Pakete zu senden. Gehen wir nun davon aus, dass R1 die Gruppe wieder verlassen möchte (Abbildung 5(d)). Erhält also S keine JOIN-Pakete mehr von R1, so setzt er nach einem Time-Out von einer Sekunde (TO1) den MFT-Eintrag für R1 auf not alive, sendet jedoch für weitere Sekunden (TO2) TREE-Pakete an R1, da ja möglicherweise noch andere Gruppenteilnehmer an diesen Ast angeschlossen sind. Diese TREE-Nachrichten werden als stale gekennzeichnet (stale-Bit ist gesetzt). Nicht verzweigende Knoten die diese TREE-Nachricht erhalten, löschen den entsprechenden Eintrag in ihrer MCT. Verzweigende Knoten hingegen setzten bei dem dazugehörigen MFT-Eintrag das stale-Bit. Dadurch wird die nächste JOIN-Nachricht von einem angeschlossenen Empfänger nicht mehr verworfen, sondern Richtung Sender weitergeleitet, wodurch der Empfänger wie bereits besprochen wieder in die Gruppe aufgenommen wird (Abbildung 5(e)). Nachdem R2 über eine neue Route an den Sender angebunden ist, empfängt er bis zu dem Zeitpunkt TO2 einige Datenpakete doppelt (Abbildung 5(f)), bis S das Senden an R1 einstellt (Abbildung 5(g)).

Weitere mögliche Entscheidungen der Knoten beim Erhalten einer JOIN- oder TREE-Nachricht sind in den folgenden Flussdiagrammen in Abbildung 6 dargestellt.

Zur Bewertung der Performance von REUNITE, wurden die „durchschnittliche Redundanz (AR)“, d.h. das Verhältnis von gesamter Anzahl der gesendeten Multicast-Pakete zu gesamter Anzahl unterschiedlicher Pakete und die „maximale Redundanz (MR)“, d.h. die maximale

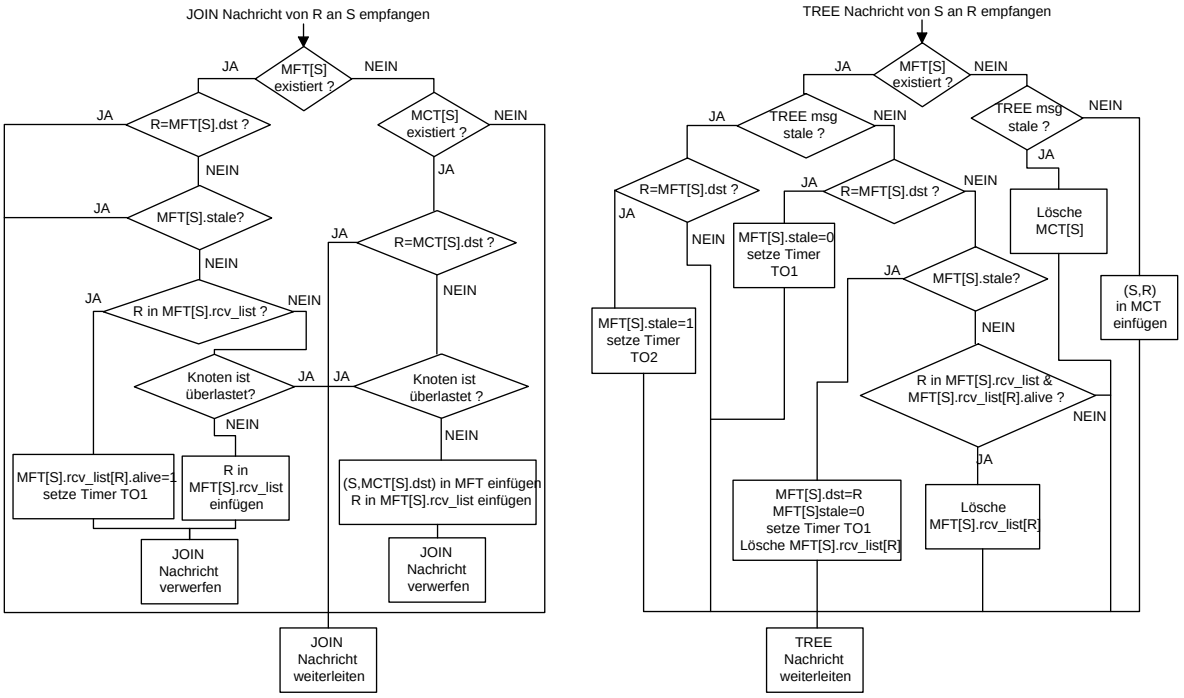

Abbildung 6: JOIN- und TREE-Nachrichten-Verarbeitung

Anzahl von Kopien eines Paketes, inkl. dem Original-Paket. Simulationen eines Multicast-Router-Netzes mit REUNITE, welches aus 8 Sendern und 64 Empfängern bestand, führte zu Folgenden Ergebnissen:

% REUNITE Router	0	20	40	60	80	100
AR	2,063	1,697	1,418	1,257	1,132	1
MR	12	8	5	4	3	1

Abbildung 7: AR und MR in Prozent, abhängig von der Anzahl der REUNITE-Router

Die Tabelle in Abbildung 7 zeigt AR und MR auf allen Links des Netzwerkes, in Abhängigkeit der prozentuellen Anzahl der Router die REUNITE unterstützen. Im Falle von 0 Prozent würden alle Multicast-Pakete über Unicast-Verbindungen übertragen. Im Idealfall einer 100-prozentigen Implementierung würde auf keinem Link mehr ein Paket doppelt übertragen werden.

4 Endsystem-Multicast

4.1 Idee der Implementierung von Multicast Services in die Endsysteme

Ein Verlagerung der Multicast-Funktionalität in die Endsysteme wird von Y. Chu, S. Rao und H. Zhang in einem Artikel über Endsystem-Multicast[ChRZ00] diskutiert. Eine klassische Netzwerkarchitektur besteht aus Endsystemen und dem diese Endsysteme verbindenden Netzwerk. Bei der Frage der Implementierung neuer Funktionen in die Netzwerkarchitektur stellt sich deshalb immer die Frage der Aufgabenverteilung. Funktionen wie z.B. QoS

können nicht alleine in den Endsystemen implementiert werden, ganz im Gegensatz zum Multicasting. Die sich daraus ergebende Frage ist nun, wie sich eine solche Verlagerung der Multicast-Funktionalität in die Endsysteme realisieren lässt.

4.2 Wie funktioniert das Endsystem-Multicast

Das Endsystem-Multicast arbeitet unabhängig von den Routern. Es abstrahiert die physikalische Topologie durch einen kompletten Virtuellen Graphen (CVG) wie nachfolgendem gezeigt wird.

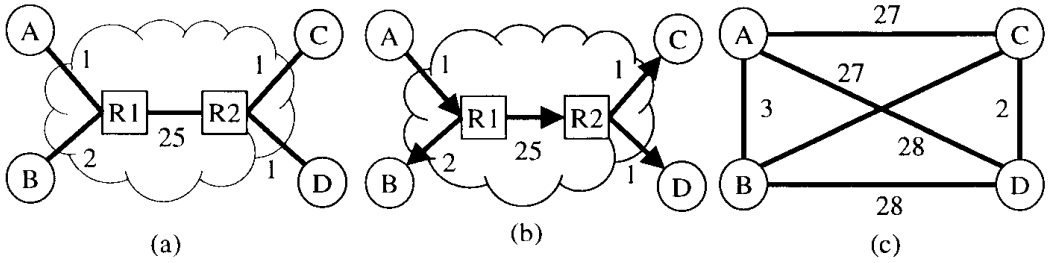

Abbildung 8: Beispiel eines CVG

In Abbildung 8(a) sind die Endsysteme A, B, C und D über die Router R1 und R2 verbunden, wobei die Verbindungen mit ihren Kosten bewertet sind. Gehen wir davon aus, das A eine Multicast-Verbindung zu allen anderen Endsystemen aufbauen möchte. Abbildung 8(b) zeigt einen IP-Multicast-Baum wie ihn DVMRP in diesem Fall erstellen würde. In Abbildung 8(c) ist der CVG für die Verbindungen zwischen den Endsystemen abgebildet. Das Endsystem-Multicast muss nun für ein CVG einen überspannenden Baum kreieren, welcher eine optimale Ausnutzung der Ressourcen gewährleistet. Zur Lösung dieser Aufgabe wurde NARADA entwickelt.

4.3 NARADA, ein Protokoll für Endsystem-Multicast

Durch NARADA wird über das bestehende Netz von Unicast-Routern ein virtuelles Netz gespannt (Endsystem-Overlay), welches von NARADA in den Endsystemen verwaltet wird, und das darunter liegende Router-Netzwerk nutzt. Beim Design von NARADA wurden folgende Ziele verfolgt: Das Erstellen des Endsystem-Overlays soll in verteilter Form stattfinden und unempfindlich gegen dynamischen Wechsel der Gruppenmitglieder sein. Die konstruierten Bäume sollen möglichst wenig Konzentration von Daten auf einzelnen Links aufweisen, geringen Relative Delay Penalty (RPD, Verhältnis von Multicast zu Unicast Delay) und geringen Ressourcenverbrauch aufweisen. Des Weiteren müssen Mechanismen eingebaut werden, welche Informationen über das Netz sammeln und eine Optimierung erlauben.

Wie schon angesprochen geht NARADA zur Konstruktion der Bäume in zwei Schritten vor. Zuerst wird ein gerichteter Graph konstruiert, das so genannte NETZ, aus welchem dann im zweiten Schritt ein (Reverse) Shortest Path Tree für das NETZ erstellt wird (Abbildung 9).

Bei NARADA hält jedes Gruppenmitglied eine Liste mit Einträgen für alle Gruppenmitglieder vor. Da NARADA für kleine Gruppen entwickelt wurde, stellt dies kein größeres Problem dar. Diese Gruppenmitgliedslisten müssen alle aktualisiert werden, wenn ein Mitglied neu zur Gruppe hinzukommt oder diese verlässt. Um dies zu gewährleisten, sendet jedes Gruppenmitglied periodisch eine Aktualisierungsnachricht an seine Nachbarn im Netz. Empfängt nun ein Mitglied von seinem Nachbarn eine solche Aktualisierungsnachricht, so vergleicht es

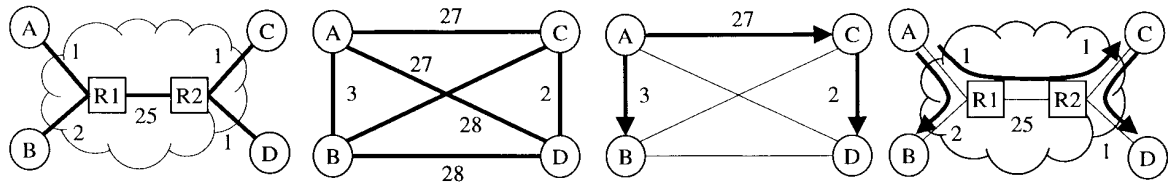

Abbildung 9: Erstellen eines Shortest Path Tree aus dem CVG

die Einträge aller Gruppenmitglieder, wobei es sowohl fehlende, als auch aktuellere Einträge aktualisiert. Möchte nun ein neues Gruppenmitglied der Gruppe beitreten, so fordert es über einen „bootstrap"-Mechanismus (welcher in NARADA nicht näher spezifiziert wird) eine Gruppenliste für die gewünschte Gruppe an. An diese Gruppenmitglieder sendet es dann einen REQUEST, als Nachbar aufgenommen zu werden. Diesen Vorgang wiederholt es solange, bis es einen RESPONSE bekommt. Danach beginnt es REFRESH-Nachrichten auszutauschen. Will ein Mitglied die Gruppe verlassen, so teilt es dies seinen Nachbarn mit, welche die Information im NETZ verteilen. Wenn ein Mitglied ausfällt, und keine REFRESH-Nachrichten mehr sendet, so schicken ihm seine Nachbarn eine Kontrollnachricht. Bekommen diese darauf keine Antwort, so wird das Mitglied als DEAD im NETZ bekannt gemacht. Die DEAD-Member-Informationen in den Mitgliedstabellen können nach einem bestimmten Zeitraum wieder gelöscht werden. Das so entstehende NETZ kann aus verschiedenen Gründen nicht optimal sein:

- Ein Gruppenmitglied welches neu zur Gruppe hinzu kommt, wird zufällig und unter Berücksichtigung seiner meist unvollständigen Informationen über das NETZ vorgenommen.
- Um ein geteiltes NETZ wieder zu verbinden, werden schnell Verbindungen hergestellt, welche auf Dauer optimiert werden können.
- Die Zugehörigkeit von Gruppenmitgliedern ändert sich im Laufe der Zeit.
- Das darunter liegende Unicast-Router-Netz kann sich ändern.

Deshalb gibt es eine Utility-Funktion, anhand derer entschieden wird, ob ein Link hinzugefügt oder entfernt wird. Diese Funktion bewertet die Verzögerung im Vergleich zur Unicast-Verzögerung zwischen den Nachbarn.

Das Endsystem-Multicast zeigt, das es für kleine Gruppen möglich ist eine effiziente Multicast-Funktionalität in die Endsysteme zu verlagern, welche auch die Gruppenverwaltung übernimmt. Dies wird durch ein sich selbst organisierendes Overlay-Netzwerk ermöglicht, welches über das dynamische IP-Unicast-Netz gelegt wird. Diese Form von Overlay-Netz ist unabhängig vom Multicasting und kann auch in anderen Zusammenhängen eingesetzt werden.

5 Vergleich der vorgestellten Architekturen und ihrer Einsatzgebiete

Bei den meisten sich heute im Einsatz befindenden Multicast-Routing-Protokollen ist es notwendig, dass jeder Router des Multicast-Baumes das entsprechende Protokoll unterstützt, um die Datenpakete weiterzuleiten. Das liegt daran, dass die Weiterleitung der Datenpakete auf

Klasse-D-IP-Adressen basiert. Dies ist ein wesentlicher Unterschied zu den beiden hier vorgestellten Protokollen, bei denen *Unicast-IP-Adressen für die Weiterleitung der Daten* genutzt werden. Dadurch bieten beide Protokolle die Möglichkeit Multicast-Funktionalität auch nur für kleine verteilte Gruppen in Teilbereichen eines Netzwerkes zu implementieren. Während das Endsystem-Multicast speziell nur für kleinere Gruppen ausgelegt ist – hier wird ja in jedem Host ein Eintrag für jedes Gruppenmitglied vorgehalten – ist REUNITE auch für große Gruppen sehr gut geeignet. Ein weiterer Vorteil von REUNITE ist es, Lasten dynamisch verteilen zu können. Während es bei heutigen Multicast-Protokollen durch Überlastung eines Routers zur Aufspaltung der Gruppe kommen kann, wird bei REUNITE durch die Möglichkeit des Ignorierens eines JOIN-Paketes, der JOIN einfach an den nächsten Router in Richtung Quelle weitergeleitet. Da Gruppen in REUNITE durch Unicast-Adressen und lokal eindeutige Portnummern und beim Endsystem-Multicast durch den Eintrag von Unicast-Adressen in spezielle Gruppentabellen festgelegt werden, *entfällt* hier auch *die Verwaltung global eindeutiger Klasse D-IP-Adressen*, die ja bekanntlich sehr begrenzt sind.

Die Möglichkeit dieser Protokolle auf bereits bestehende Unicast-Routing-Netze aufzubauen, und diese für die Kommunikation zwischen den zur Unterstützung der Multicast-Fähigkeit zu implementierenden Protokollen zu nutzen, macht sie für viele Einsatzgebiete sehr interessant. Die Haupteinsatzgebiete für diese neuen Protokolle sehe ich vor allem bei dem oben schon angesprochenen Aufbau kleiner Multicast-Gruppen, beispielsweise im Bereich von Firmen, welche über das Internet ihre Multicast-Anwendungen implementieren möchten. Ein großer Vorteil hierbei liegt in den geringeren Investitionskosten, die eben nur für die Hosts, oder einige wenige Router anfallen. Den Vorzug von REUNITE gegenüber dem Endsystem-Multicast sehe ich in der größeren Flexibilität, was die Erweiterung der Gruppen angeht, da hier auch ohne Probleme sehr große Gruppen gebildet werden können. Der Ausbau des Multicast-Netzes kann mit REUNITE sehr einfach durch die Implementierung von weiteren REUNITE-Routern an *wichtigen Knoten* erreicht werden. Weiterhin profitieren alle Nutzer dieses Protokolls auch von der Implementierung „fremder Firmen“ welche REUNITE auf Ihren Routern einsetzen. Durch das Implementieren von REUNITE auf den Unicast-Routern, muss auch beim Weiterleiten von Paketen die „nicht“ für eine Gruppe bestimmt sind, dies zuerst durch einsehen der MFT geprüft werden. Darum sollte in weiteren Schritten durch ein Zusammenfassen von Adressen, die Größe der MFT weiter reduziert werden. Ich gehe jedoch davon aus, dass aufbauend auf diese Protokolle die Anforderungen zukünftiger Multicast-Anwendungen besser gewährleistet werden können.

Literatur

[Ball97] A. Ballardie. Core Based Trees (CBT) Multicast Routing Architecture. RFC 2201, September 1997.

[ChRZ00] Y. Chu, S. Rao und H. Zhang. A Case for End System Multicast. Proceedings of ACM SIGMETRICS, Juni 2000.

[S.E.94] S.E.Deering. Host Extension for IP Multicasting. RFC1112, August 1994.

[SEFJ94] S.E.Deering, D. Estrin, D. Farinacci und V. Jacobson. An Architecture for Wide-Area Multicast Routing. ACM SIGCOMM, 1994.

[StNZ00] I. Stoica, T. Eugene Ng und H. Zhang. REUNITE: A Recursive Unicast Approach to Multicast. INFOCOM '00, März 2000.

[ThEM98] D. Thaler, D. Estrin und D. Meyer. Border Gateway Multicast Protocol (BGMP). INTERNET DRAFT, August 1998.

[WaPD88] D. Waitzman, C. Patridge und S. Deering. Distance Vector Multicast Routing Protocol. RFC 1075, November 1988.

Multicast-Transportprotokolle

Tobias Kraft

Kurzfassung

Aufgrund neuer Anwendungen und Dienste im Internet verändern sich auch die Anforderungen an die Übertragungsprotokolle. Multicasting ist ein Schlagwort das man in diesem Zusammenhang immer wieder hört. Es ermöglicht die Übertragung von Daten an mehrere Empfänger gleichzeitig. Dies ist mit der bisher eingesetzten Unicast-Kommunikation ebenfalls eingeschränkt möglich, allerdings führt dies zu einer hohen Inanspruchnahme der Netzwerkressourcen. Multicasting ist nicht ohne Veränderungen der bisherigen Protokolle möglich. Im Bereich der Transportschicht trifft dies inbesondere auf TCP zu, das für den Multicast-Verkehr nicht geeignet ist. Deshalb wurden in letzter Zeit einige neue Multicast-Transportprotokolle entwickelt. Im folgenden Dokument werden die Anforderungen an Multicast-Transportprotokolle aufgezeigt und 6 ausgewählte Protokolle kurz vorgestellt.

1 Einleitung

Das Internet verwendet als eine seiner Basistechnologien zur Übertragung von Daten die TCP/IP Protokollfamilie, die eine Punkt- zu Punkt-Kommunikation ermöglicht. Dies bedeutet z.B. für das World Wide Web, dass eine Person eine bestimmte Webseite aufruft und danach von einem Server die angeforderte Seite zurückerhält. Diese Verbindungsart wird auch als Unicasting gekennzeichnet. Aufgrund der technischen Weiterentwicklung sind auch andere Arten der Kommunikation nötig. Hier ist speziell das Multicasting gemeint, das eine Punkt-zu Mehrpunktverbindung ermöglicht oder auch die Multipeer-Kommunikation (Mehrpunkt-zu Mehrpunktverbindung). Hierdurch können unter anderem Videokonferenzen mit mehreren Teilnehmern abgehalten werden, das Zustellen von Software-Updates für viele Kunden oder auch die Ausstrahlung von Liveübertragungen vorgenommen werden. Momentan läuft der Vorgang im Internet so ab, dass der Sender die Daten einfach X-mal versendet, wenn es X Empfänger gibt. Dies kostet natürlich unnötig Ressourcen, da es effektiver wäre die Daten nur einmal zu versenden und bei den Routern, die in verschiedene Richtungen weiterleiten müssen, zu duplizieren. Die Daten fließen so nur einmal über die Verbindungsstrecke von Server zu Router und von Router zu Router. Es entsteht hierdurch eine Baumstruktur mit dem Sender als Wurzel (siehe Abbildung 1). Dieser Baum wird auch Multicastbaum genannt.

Für dieses Verfahren sind auf Ebene der Vermittlungsschicht sogenannte Multicast-Gruppen-Adressen (Class-D-Adressen) nötig. Eine Multicast-Adresse ist für alle Empfänger, die an der gleichen Anwendung teilnehmen, identisch. Die Empfänger und auch Sender mit einer gleichen Adresse werden Gruppe genannt. Eine Gruppe hat laut [ZiWi99] verschiedene Eigenschaften. Diese sind *Offenheit*, *Dynamik*, *Lebensdauer*, *Sicherheit*, *Bekanntheit* und *Heterogenität*. In *offenen* Gruppen können Sender auch außerhalb der Gruppe existieren. *Geschlossene* Gruppen erlauben dagegen nur interne Kommunikation. Die *Dynamik* bezieht sich auf die Dauerhaftigkeit der Gruppe. Bei statischen Gruppen ist die Zusammensetzung vorgegeben. Bei dynamischen steht eine Art Registrierung zur Verfügung, die einem Host jederzeit die Möglichkeit gibt

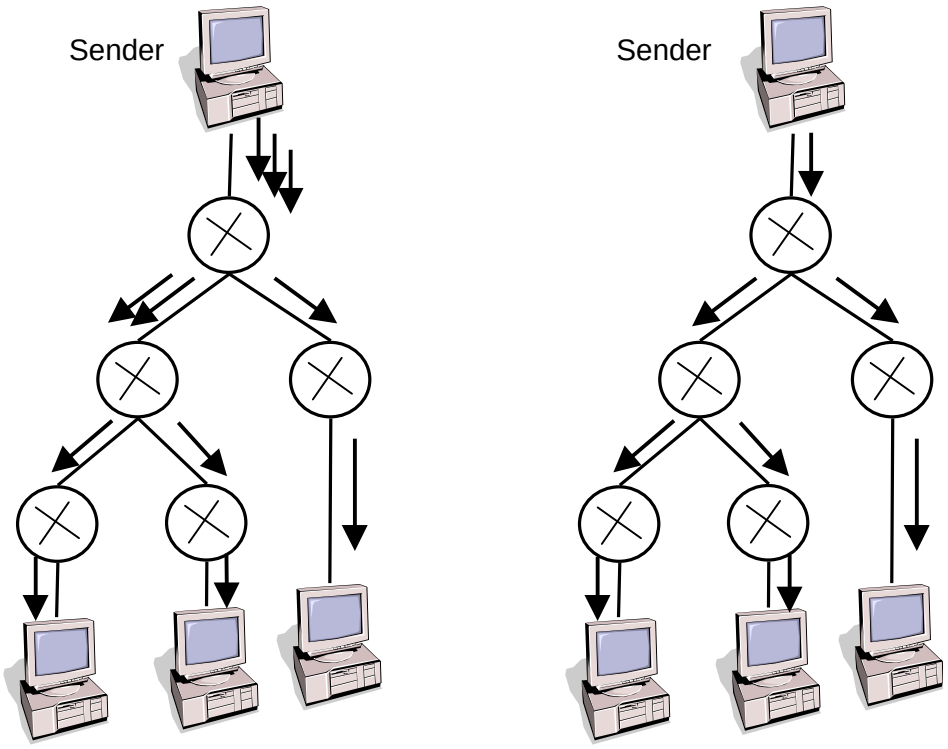

Abbildung 1: Punkt-zu-Mehrpunkt-Kommunikation im Unicast-Modell und im Multicast-Modell

der Multicast-Gruppe beizutreten bzw. diese zu verlassen. Die Sicherheitsanforderungen unterscheiden sich von Teilnehmerkreis zu Teilnehmerkreis (z.B. Vorstandsvideokonferenz versus Ausstrahlung von Nachrichten). Die Bekanntheit einer Gruppe hat wesentliche Eigenschaften auf die realisierbaren Gruppendienste. Bei *anonymen* Gruppen ist die Identität der einzelnen Mitglieder nicht bekannt und somit ist es auch schwierig eine zuverlässige Auslieferung der Daten zu garantieren. Bezüglich der Lebensdauer werden *permanente* und *transiente* Gruppen unterschieden. Erstere existieren auch dann weiter, wenn es momentan keine aktiven Mitglieder gibt. Zuletzt lassen sich noch *homogene* und *heterogene* Gruppen unterscheiden. Bei heterogenen Gruppen haben die Mitglieder z.B. unterschiedliche Netzanbindung oder verschiedene Qualitätsanforderungen.

Damit die heutigen Router und auch Endsysteme diese Art von Internetverkehr unterstützen können, müssen sie um Funktionalitäten erweitert werden. Multicast-fähige sind Systeme, die IP-Multicast-Adressen verstehen, das Gruppenmanagement-Protokoll IGMP und Multicast-Routing durchführen können. Mit Hilfe der genannten Mechanismen existiert die Möglichkeit Multicastbäume aufzubauen. Eine erste Implementierung eines multicast-fähigen Systems ist auch schon mit dem MBone geschehen, das aus multicastfähigen Teilnetzen besteht, die über Verbindungen (Tunnel und Domänen) miteinander verbunden sind.

Bis jetzt haben wir nur die notwendigen Veränderungen bis zur Schicht 3 kurz betrachtet. Es sind aber auch auf der Transportschicht Änderungen nötig, die im nächsten Kapitel erläutert werden.

2 Multicast-Transportprotokolle und an sie gerichtete Anforderungen

Bisher verwendet die Transportschicht als Protokoll hauptsächlich das Transmission Control Protocol (TCP) und in einigen Fällen auch UDP (User Datagram Protocol). TCP basiert auf einer reinen Punkt-zu-Punkt-Kommunikation und hat folgende typische Eigenschaften (siehe auch [KrRe00]):

- Verbindungsorientierung: Aufbau einer virtuellen Verbindung und nach dem Datenaustausch wieder Abbau dieser Verbindung.
- Zuverlässigkeit: Der Empfänger muss jedes erhaltene Paket bestätigen. Bleibt eine Bestätigung aus, wiederholt der Sender die Übertragung. Übertragungsfehler werden durch eine Prüfsumme im Header erkannt. Des weiteren sorgt TCP dafür, dass Pakete in der richtigen Reihenfolge und ohne Duplikate beim Empfänger ankommen.
- Flusskontrolle: Um Daten kurzfristig zu speichern besitzt jede TCP-Instanz einen Puffer. Um einen Überlauf dieses Puffers zu vermeiden, teilt die empfangende Instanz dem Sender mit, wieviel Bytes sie akzeptiert.
- Vollduplex: TCP-Instanzen können gleichzeitig senden und empfangen.

TCP genügt den Eigenschaften, die für den Unicastverkehr benötigt werden, eignet sich allerdings nicht für den Multicastverkehr. Die Aufgaben der Protokolle in der Transportschicht (Schicht 4 des ISO/OSI-Referenzmodells) sind bei der Multicastkommunikation zwar ähnlich denjenigen der Unicastkommunikation, es werden aber doch in einigen Fällen andere Anforderungen gestellt.

Würde man TCP beim Multicasting einsetzen, wären Probleme bei der Fehlerkontrolle zu erwarten. Jeder Empfänger muss an den Sender eine Quittung über die erhaltenen Daten schicken. Bei vielen Empfängern bzw. großen Gruppen führt dies zu einer *Quittungs-Implosion* beim Sender. Der Sender erweist sich als Flaschenhals, da er nicht alle Quittungen verarbeiten kann. Die Fehlerkontrolle kann deshalb aber nicht vernachlässigt werden, da im Bereich Multicasting die *Zuverlässigkeit* eine große Rolle spielt (*Reliable Multicast*). Definitionsgemäß liegt ein zuverlässiger Dienst vor, wenn alle Daten fehlerfrei, in der richtigen Reihenfolge und ohne Duplikate beim Empfänger ankommen. Um dies zu gewährleisten müssen Kontrolldaten zwischen den Kommunikationspartnern ausgetauscht werden, beispielsweise um den ordnungsgemäßen Erhalt von Dateneinheiten zu garantieren. Für zuverlässiges Multicasting gibt es außer dem Engpass beim Sender auch Probleme hinsichtlich der Bekanntheit von Gruppenmitgliedern. Die Bekanntheit aller Mitglieder ist nicht ohne weiteres zu gewährleisten, da eine Gruppe oftmals einer hohen Dynamik unterliegt.

Im Hinblick auf die *Skalierbarkeit* muss man bei Multicasting an die Gruppentopologie und die Heterogenität denken. Aus gruppentopologischer Sicht ist es möglich, dass ein Empfänger in nächster Nähe zum Sender sitzt, während ein weiterer Empfänger sich auf einem anderen Kontinent aufhält. Aufgrund verschiedener Netze können einige Empfänger direkt an Hochleistungsnetze gekoppelt sein und andere hingegen an langsamere und fehleranfälligere drahtlose Netze. Beide genannten Fälle führen zu unterschiedlichen Verzögerungsraten. Sind die Verzögerungsraten zu lang, können beispielsweise bei Videoanwendungen Daten nicht wiederübertragen werden, da sie schon „verfallen“ sind.

Zusammenfassend kann man sagen, dass die Protokolle der Transportschicht in Unicast und Multicast viele gemeinsame Funktionen haben. In beiden Fällen sind sie für die Fehlererkennung und -behebung, die Verbindungsverwaltung und die Fluß- und Staukontrolle zuständig.

Im Bereich Multicasting existieren an das Protokoll der Transportebene, aufgrund verschiedener Anwendungen, ganz unterschiedliche Anforderungen.

Im nachfolgenden Abschnitt wird eine Auswahl an Multicast-Transportprotokollen näher betrachtet und miteinander verglichen.

3 Multicast-Transportprotokolle im Vergleich

3.1 UDP (User Datagram Protocol)

Das User Datagram Protocol ist laut [ZiWi99] ein einfaches Transportschichtprotokoll, das zur Internet-Protokollfamilie gehört. Es wird hier in diesem Rahmen nur kurz vorgestellt, da es momentan in der Praxis unter anderem für Multicasting verwendet wird und auch verschiedene andere Multicast-Protokolle auf UDP aufsetzen. Allerdings ist es aufgrund mehrerer Einschränkungen nicht das geeignetste Protokoll für diesen Zweck. UDP ist ein verbindungsloses Protokoll, mit dem Daten unzuverlässig verschickt werden, d.h. der Sender kann nicht feststellen, ob die Daten beim Empfänger korrekt angekommen sind. Der fehlende Verbindungskontext verhindert die Flusskontrolle und Sicherung gegen Datenverlust auf dieser Schichtenebene. Eventuell ist eine Verlagerung dieser Funktionen auf höhere Schichten nötig. Ein weiteres Problem ist, dass Gruppenmanagement nicht vorgesehen ist.

In UDP, wie auch in TCP, wird die Adressierung der Transportschicht durch eine Portnummer und eine Identifikation des Protokolls (hier also UDP) vorgenommen. Eine Portnummer ist innerhalb einer UDP-Instanz eindeutig und ist eine Zahl zwischen 0 und 65535 (16 Bit). Portnummer und IP-Adresse zusammen ermöglichen eine eindeutige Identifikation des Empfängers und werden auch Socket genannt. Wird nun bei einer Datenübertragung eine IP-Unicast-Adresse angegeben erfolgt Punkt-zu-Punkt-Kommunikation. Bei Angabe einer IP-Multicast-Adresse wird alles per Multicast verschickt. Die Nutzung von UDP als Multicast-Protokoll ist möglich, da im Gegensatz zu TCP keine Modifikationen der Protokollfunktionen notwendig sind. Die Quittungsimplosion und Probleme mit der Übertragungswiederholung bleiben aufgrund der Verbindungslosigkeit aus.

Typische Anwendungsgebiete von UDP per Multicast sind die Übertragung von Audio- und Videodaten.

3.2 RMTP (Reliable Multicast Transport Protocol)

Bei RMTP handelt es sich um ein Protokoll, bei dessen Entwicklung speziell die Skalierbarkeit und die Zuverlässigkeit berücksichtigt wurden. Außerdem wurde auch der Einsatz für globale Netze (also z.B. Internet) berücksichtigt, da viele andere Transportprotokolle nur in lokalen Netzen den Anforderungen genügen. Dieses Protokoll wurde erstmals 1996 in [PaSa96] vorgestellt und ist hauptsächlich für Verteildienste, wie z.B. das Ausliefern von Aktienkursen an mehrere Kunden per Multicast, entwickelt worden.

Von den darunter liegenden Schichten des Netzwerks wird lediglich erwartet, dass ein Multicast-Baum vom Sender zu den Empfängern bereitgestellt wird. Dieser kann von einem Multicast-Routing-Protokoll, wie z.B. DVMRP (Distance Vector Multicast Routing Protocol) oder PIM (Protocol-Independent Multicast) erzeugt werden.

Die Architektur von RMTP sieht außer dem Sender und den Empfängern noch Designated Receiver (DR) vor, die selbst auch Empfänger mit Zusatzfunktionen sind. Die erweiterten Funktionen eines DR sind das Sammeln und Verarbeiten von Quittungen, Weiterleitungen an den Sender und Wiederübertragung von Paketen. Ein Beispiel für einen Multicast-Baum

mit angebundenen Stationen wird in Abbildung 2 gezeigt. Die Datenübertragung vom Sender erfolgt mit Hilfe der IP-Multicastadresse zu allen Empfängern einschließlich der DR.

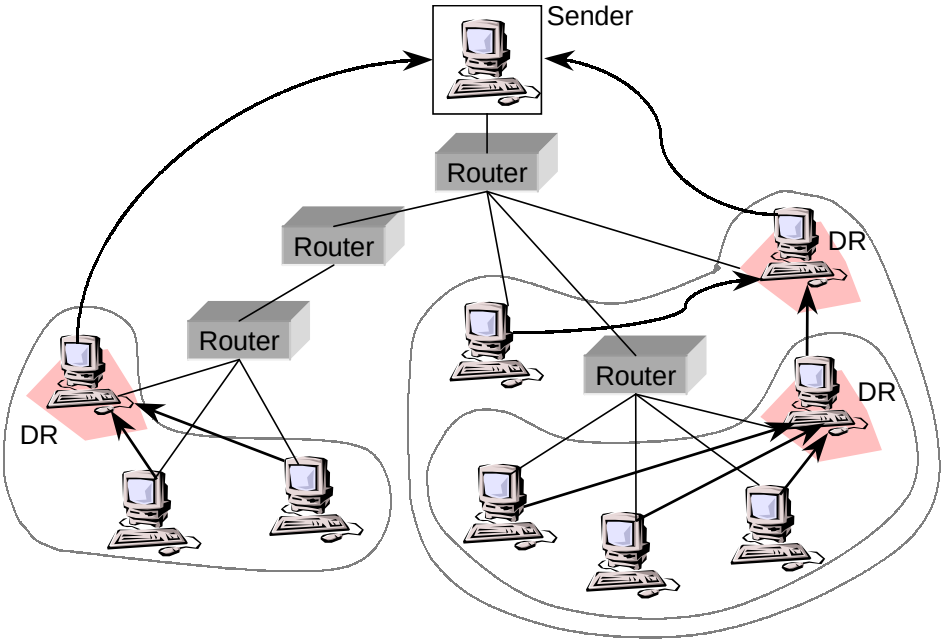

Abbildung 2: Multicast-Baum mit Designated-Receivern

Jede RMTP-Verbindung hat gewisse Verbindungs-Parameter (connection parameters), die von einem Session Manager zur Verfügung gestellt werden, dieser ist allerdings nicht Bestandteil des RMTP-Protokolls. Solche Verbindungsparameter sind unter anderem T_{dally} (Überwachungszeit nach dem Senden des letzten Paketes) oder Cache-Size (Cache-Größe des Senders). Hat nun der Session Manager dem Sender und den Empfängern die Parameter übergeben, aktivieren die Empfänger einen Verbindungskontrollblock und fallen in einen nicht verbundenen Zustand zurück, bis sie Datenpakete vom Sender erhalten. Die gesendeten Nutzdaten haben alle die gleiche Länge, bis auf die letzte Dateneinheit, die kürzer sein kann. Die Nutzdateneinheiten sind vom Typ DATA und die letzte Dateneinheit vom Typ DATA_EOF, damit die Empfänger das Ende des Datentransfers erkennen können. Hat der Sender sein letztes Datenpaket gesendet, wartet er die Zeitspanne T_{dally} ab und löscht danach alle Zustandsinformationen der Verbindung. Diese Zeit wird abgewartet um Quittungen (ACK) zu empfangen, welche Empfänger in periodischen Abständen senden. Hierbei kommen nun auch die Designated Receiver zum Einsatz. Diese Quittungen werden nämlich im Regelfall von den Empfängern nicht zum Sender gesendet, sondern zu dem jeweiligen nächsthöher gelegenen Designated Receiver. Der Sinn dieses Verfahrens ist die Vermeidung einer Quittungsimplosion beim Sender, da die Anzahl der gesendeten Quittungen mit der Anzahl der Sender steigt und der Sender somit Probleme mit der Verarbeitung bekommen kann. Die Designated Receiver senden ihre Quittungen wiederum an den Sender oder den nächst höher gelegenen DR weiter.

Eine Quittung besteht im wesentlichen aus einer Sequenznummer L und einem Bitvektor. Die Sequenznummer gibt diejenige Paketnummer an, bis zu welcher der Empfänger die Pakete mit kleinerer Sequenznummer vollständig erhalten hat. Hierzu muss man wissen, dass für die gesendeten Pakete fortlaufende Nummern vergeben werden. Der Bitvektor zeigt an welche Dateneinheiten schon angekommen sind (1) und welche noch anstehen (0). Hat eine Quittung für L den Wert 20 und der Bitvektor ist 00101000 bedeutet dies, dass bis zur Sequenznummer 19 alle Pakete angekommen sind und die Pakete 20, 21, 23 und 25 bis 27 fehlen. Paket 22 und 24 sind dagegen schon eingetroffen (siehe auch Abbildung 3).

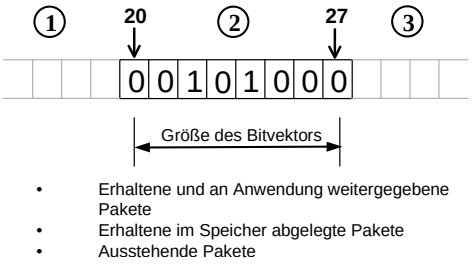

Abbildung 3: Empfängerfenster mit zugehörigem Bitvektor

Ein DR empfängt, wie oben schon erwähnt, die Quittungen von denjenigen Empfängern, die im Multicast-Baum hierarchisch unter ihm liegen. Der Designated Receiver ist für die Verarbeitung der Quittungen verantwortlich und muss die fehlenden Pakete an die Empfänger senden. Dies geschieht durch Unicast oder, falls viele Empfänger das gleiche Paket vermissen durch Multicasting. Hierzu zählt der DR während eines Zeitintervalls mit einem Zähler (C) die Anzahl der Empfänger, die das betreffende Paket nicht bekommen haben und speichert in einer Tabelle ihre Adressen ab. Überschreitet C einen Schwellwert (MCAST_thres) sendet der DR die Daten per Multicast. Dieses Problem des Multicasting ist allerdings nicht so einfach lösbar, da die Pakete nur an die unter dem DR liegenden Teilbaum gesendet werden sollen. Bei IP-Multicasting ist das sogenannte *Subcasting* allerdings nicht vorgesehen, so dass RMTP sich bis zur Verfügbarkeit einer solchen Funktion mit einer Tunneltechnik weiterhilft. Die zu sendenden Daten werden in eine IP-Dateneinheit vom Typ SUBTREE-MCAST gekapselt, die extra für diesen Zweck entwickelt wurde. Bevor der Versand des Paketes erfolgt, wird als Quelladresse die des Senders eingetragen, so dass die im Multicastbaum hierarchisch darunterliegenden Router nur an den hierarchisch unter dem DR liegenden Teilbaum weitersenden. Ein Problem gibt es allerdings im Bereich der Router: Sie müssen die Dateneinheit vom Typ SUBTREE-MCAST wieder entpacken und dazu müssen sie bis jetzt noch nicht implementierte Funktionen erhalten. Dies bedeutet wiederum, dass RMTP momentan nicht im Internet einsetzbar ist!

Ein DR erhält wie die normalen Empfänger seine fehlenden Pakete vom Sender oder einem übergeordneten DR, wenn er dies in seinen Quittungen angibt. Wichtig ist beim periodischen Senden der Quittung, dass die Intervalle weder zu groß noch zu klein sind. Zum einen, um eine zu große Verzögerung bei vermissten Daten zu vermeiden, zum anderen um Daten nicht als fehlend zu melden, obwohl sie gerade „unterwegs“ sind. Deshalb werden bei RMTP die Raten, mit denen ein Empfänger Quittungen sendet, in Abhängigkeit von der Umlaufzeit gesetzt. Hierzu wird in festen Intervallen eine Dateneinheit vom Typ RTT-MEASURE verschickt, um mit deren Hilfe die Umlaufzeit zu berechnen.

Bisher wurde nicht erwähnt, dass RMTP es Empfängern auch ermöglicht, einer laufenden Übertragung beizutreten. Damit das neue Mitglied die bisher gesendeten Daten erhält, muss es sie mit negativen Quittungen anfordern. Dazu müssen allerdings Sender und DR die bis zu diesem Zeitpunkt übertragenen Daten speichern, was problematisch werden kann. Bei der Entwicklung von RMTP wurde für die Flusskontrolle ein fensterbasiertes Verfahren gewählt, das der Sender verwaltet und beim Verbindungsaufbau vom Session Manager zugewiesen bekommt. Diese Verfahren soll auch „langsamere“ Empfänger vor Überlastung verschonen. Bei der Staukontrolle wird eine Art Slow-Start Verfahren verwendet. Hierbei geht RMTP davon aus, dass bei einer hohen Anzahl an Datenverlusten die Netzlast sehr hoch ist und sendet deshalb in längeren Abständen.

Die Vorzüge von RMTP sind die Zuverlässigkeit, obwohl nicht alle Empfänger bekannt sind. Des weiteren sind die Skalierbarkeit und die Fähigkeit, in heterogenen Netzen effektiv zu arbeiten, aufgrund der Verwendung von Designated Receivern hervorzuheben. Die DR sind aber

auch zugleich Probleme bei RMTP, da sie momentan nur statisch eingesetzt werden können. Außerdem kann es passieren, dass manche DR überlastet sind, während andere Nachbar-DR noch Kapazitäten frei haben. Ein noch größeres Problem dürften allerdings die oben schon erwähnten fehlenden Funktionen der Router sein.

3.3 SRM (Scalable Reliable Multicast)

SRM wurde im Dezember 1997 in [FJLM+97] als zuverlässiges und multicastfähiges Rahmenwerk für Light-Weight-Sessions (LWS) und das Application-Level-Framing Modell (ALF) vorgestellt. SRM geht aufgrund der Orientierung am ALF-Konzept einen anderen Weg als viele Multicast-Transportprotokolle. Das Application-Level-Framing Modell wurde 1990 von Clark und Tennenhouse vorgestellt und geht davon aus, dass verschiedene Anwendungen auch verschiedene Anforderungen an das Protokoll haben, besonders beim Multicasting. Deshalb ist es schwierig bzw. unmöglich mit einem Protokoll all diese Anforderungen abzudecken. Als Lösung zu diesem Problem schlägt das ALF-Konzept vor, nur eine Grundfunktionalität von Seiten der Protokolle zur Verfügung zu stellen. Diese Funktionalitäten können dann von einer Anwendung entsprechend ihren Anforderungen erweitert werden. Zum Verständnis sei hier noch erwähnt, dass Light-Weight Sessions (LWS) das Ergebnis der Ausarbeitung und Erweiterung von ALF um einfache auf dem IP-Multicast-Modell basierende Kommunikationsmechanismen ist, wie z.B. Video- und Audioanwendungen.

An Voraussetzungen für die Anwendung von SRM wird von der untenliegenden Schicht nichts weiter als das normale Basismodell von IP für Multicasting benötigt. Es ist noch anzumerken, dass SRM eigentlich für das sogenannte Whiteboard, welches im MBone Anwendung findet, konzipiert wurde.

Bei SRM handelt es sich um ein empfängerorientiertes Protokoll, da der Empfänger die Behebung von Übertragungsfehlern einleiten muss. Die Erkennung von Fehlern geschieht bei verlorenen Daten mit Hilfe von Sequenznummern und bei fehlerhaft übertragenen Daten mit Prüfsummen. Damit der Verlust von Dateneinheiten nicht unbemerkt bleibt, bis die nachfolgenden Daten eingetroffen sind, senden die einzelnen Gruppenmitglieder regelmäßige, periodische Statusmeldungen, sogenannte *Session Reports*. Die Session Reports dürfen nur einen kleinen Teil der zur Verfügung stehenden Bandbreite beanspruchen (z.B. 5%), der entweder durch ein Reservierungsprotokoll oder durch einen Algorithmus zur Staukontrolle bestimmt wird. Diese von jedem Empfänger versendeten Nachrichten werden per Multicast übertragen und enthalten unter anderem eine Nummer von jeder anderen Empfangsstation. Diese Nummer ist die jeweils höchste Sequenznummer, die der betreffende Empfänger von jedem anderem Empfänger erhalten hat. Nun kann jedes Gruppenmitglied seine Daten mit den empfangenen Session Reports vergleichen und feststellen, ob schon alle Daten angekommen sind. Falls Daten fehlen, sendet die Station einen *Repair Request* per Multicast und fordert damit die Übertragung der fehlenden Daten an. Diese Daten müssen aber nicht vom Sender übertragen werden, sondern können auch von einer anderen Empfangsstation gesendet werden. Um eine Quittungsimplosion zu vermeiden, falls mehrere Empfänger die gleichen Daten nicht erhalten haben, verwendet SRM beim Senden von Repair Requests einen Timer. Dies bedeutet, dass eine Station, die fehlende Daten entdeckt hat, mit dem Senden ihres Repair Request nicht sofort beginnt, sondern eine gewisse Zufallszeit abwartet, die von der Entfernung zum Sender abhängt. Hierdurch wird erreicht, dass Empfänger, die näher am Sender sind, mit größerer Wahrscheinlichkeit zuerst die Wiederübertragung von Daten anfordern. Wenn nun der Request Timer abgelaufen ist, sendet die Station ihren Repair Request und setzt ihren Request Timer hoch, allerdings auf die doppelte Ablaufzeit. Wir können nun zwei Fälle unterscheiden. Im ersten Fall empfängt eine Station Z einen Repair Request von Station X über Daten, die sie selbst noch nicht hat. Daraufhin hält sie ihren Request Timer an und setzt ihn wieder mit doppeltem Wert(siehe Abbildung 4: Timer 1 mit doppeltem Wert setzen). Gehen nach Ablauf

dieses Zeitintervalls bei Station Z weder die Daten noch ein erneuter Repair Request ein, sendet Station Z seinen Repair Request. So wird vermieden, dass unnötig viele Repair Requests gesendet werden. Im zweiten Fall erhält Station Y einen Repair Request von Station X über Daten, die bei ihr schon angekommen sind. Um das mehrfache Senden der fehlenden Daten zu vermeiden werden hier ebenfalls Zeituhren in Abhängigkeit von der Laufzeit zwischen den Stationen gesetzt (in Abbildung 4: Timer 2 setzen). Dies bedeutet, eine Station Y, die die fehlenden Daten von Station X senden könnte, setzt ihren Zeitgeber. Erhält Y nun vor dem Ablauf die geforderten Daten wird der Vorgang abgebrochen. Läuft der Timer hingegen ab sendet Y die fehlenden Daten.

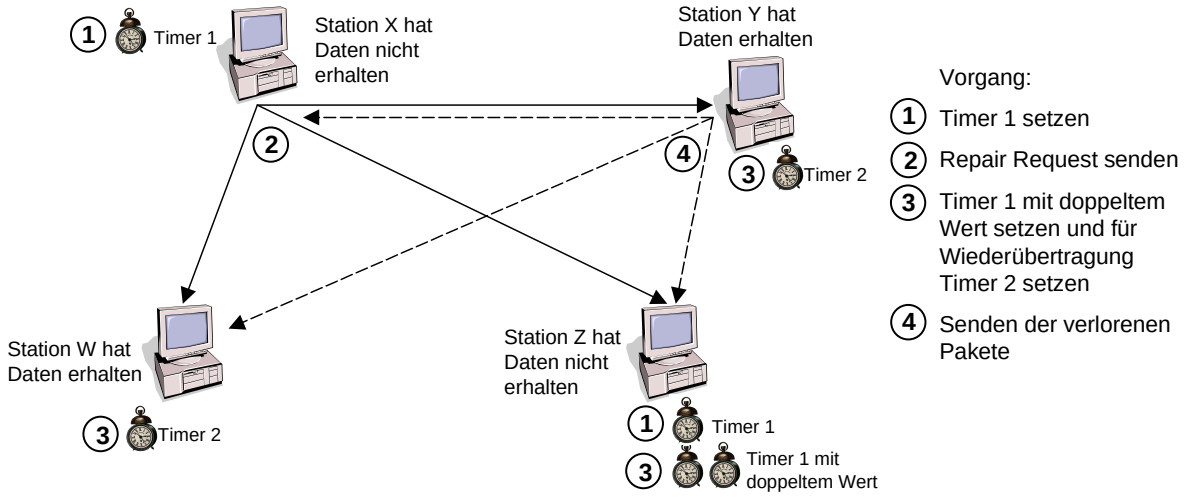

Abbildung 4: Beispiel für das Anfordern von verlorenen Daten

Außer dem soeben aufgezeigten Verfahren für den Verlust von Datenpaketen gibt es noch zwei Verfahren mit Erweiterungen (siehe auch [FJLM+97]), auf die hier aber nicht im Detail eingegangen werden soll. Das eine Verfahren versucht die Performance zu erhöhen indem die jeweiligen Timer mit komplizierteren Algorithmen gesetzt werden, während das andere darauf abzielt, lokale "Recovery-Algorithmen" zu verwenden, indem Anfragen und Repair Requests nur in einer lokalen Umgebung per Multicast verschickt werden. Im Bereich der Lokalen Recovery-Verfahren wird momentan noch weiter gearbeitet.

Zusammenfassend kann man sagen, dass SRM von Grund auf nur eine begrenzte Zuverlässigkeit anbietet und erweiterte Funktionalitäten den jeweiligen Anwendungen überlässt.

3.4 MFTP (Multicast File Transfer Protocol) und MFTP/EC

Wie der Name Multicast File Transfer Protocol schon sagt, ist dieses Protokoll für die zuverlässige Verteilung von Dateien entwickelt worden. Ziel ist es, einer großen Anzahl an Empfängern per Multicast Dateien auszuliefern. Eine typische Anwendung stellt das Verteilen von Software-Updates an Kunden dar. Die Anforderungen an MFTP unterscheiden sich deutlich von anderen Multicast-Protokollen. Die Empfänger senden außer Statusmeldungen keine Informationen an die Quelle. Außerdem bestehen keine Anforderungen bezüglich Echtzeit. Es kommt hier nicht darauf an, ob das Paket ein paar „Sekunden" früher oder später den Empfänger erreicht.

MFTP teilt eine Datei in Pakete auf und bildet aus mehreren Paketen Blöcke gleicher Größe. Jeder Block enthält so viele Pakete, wie Bits in ein IP-Paket mit maximaler Größe passen. Kann z.B. ein Netzwerk IP-Pakete mit einer Länge von bis zu 1500 Bytes weiterleiten, bestehen die Blöcke aus annähernd 12000 Paketen.

In einem ersten Durchlauf versendet der Sender alle Pakete per Multicast an die Gruppe. Die Empfänger schreiben ihrerseits alle erhaltenen Daten in eine Datei. Kommt ein Paket nicht an, lässt ein Empfänger an der entsprechenden Stelle ausreichend Platz, um später die Lücke mit dem nachgesendeten „Ersatzpaket“ zu füllen. Fehlt in einem Block ein Paket, sendet der Empfänger per Unicast eine negative Empfangsbestätigung (NACK) zurück. Die negative Empfangsbestätigung enthält den Status jedes Paketes eines Blocks (entweder empfangen oder ausstehend). Um eine Quittungsimplosion zu vermeiden, verzögert der Empfänger das Senden um bis zu zwei Sekunden. Der Sender seinerseits sammelt alle NACK's und bestimmt diejenigen Datenpakete, die nicht angekommen sind. Danach wird der zweite Durchlauf mit dem Senden aller nicht angekommen Pakete gestartet. Der Sender wartet wieder auf negative Empfangsbestätigungen und führt den Vorgang solange fort, bis die Empfänger alle Pakete erhalten haben. Hat ein Empfänger alle Daten erhalten verlässt er die Multicastgruppe. Die Fehlerbehebung findet erst statt, nachdem alle regulären Daten gesendet wurden. Dies stellt hier kein Problem dar, da man später an beliebige Stellen der Datei springen kann um fehlende Pakete einzufügen.

Staukontrolle wird momentan durch Setzen eines Schwellwertes ausgeführt. Übersteigt die Anzahl der verlorenen Pakete bei einem Empfänger diesen Schwellwert, muss er die Gruppe verlassen. In zukünftigen Versionen soll der Sender in diesem Fall eine zweite Verbindung auf einem extra Kanal aufbauen und die Daten dort mit einer niedrigeren Rate senden.

MFTP/EC stellt eine Erweiterung des Multicast File Transfer Protocols mit Erasure Correction (EC) dar. Es wird versucht Methoden für eine effizientere Fehlerbehandlung einzuführen. Hierzu teilen bei MFTP/EC Sender und Empfänger die zu übertragenden Daten in Gruppen mit k Paketen ein (k sei eine kleine natürliche Zahl). Es werden ca. 12000 Gruppen gebildet. Nach der Übertragung der kompletten Datei senden die Empfänger negative Empfangsbestätigungen der einzelnen „Gruppenpakete“ an den Sender, die anzeigen ob alle Pakete einer Gruppe korrekt angekommen sind oder nicht. Im nächsten Durchlauf verschickt der Sender per Multicast ein Paket mit redundanten Daten an jede Gruppe. Mit Hilfe dieser redundanten Pakete können Empfänger ihre verlorenen Pakete rekonstruieren. Die Durchläufe werden so lange wiederholt, bis alle Pakete korrekt bei den Empfängern sind. Die genauere Beschreibung des Reparierens mit redundanten Paketen kann auch unter [HoHä98] nachgelesen werden.

3.5 SCE (Single Connection Emulation)

Single Connection Emulation ist kein neues Multicast-Transport-Protokoll, sondern stellt eine Erweiterung der existierenden Protokollarchitekturen mit dem Ziel zuverlässiges Multicasting bereitzustellen, dar. Betrachtet man die drei oberen Schichten der Internet-Protokollsuite, bietet die Netzwerkschicht einen verbindungslosen Dienst ohne Garantien für Zuverlässigkeit (best-effort). SCE lässt diese Schicht unverändert und nutzt deren Multicastfähigkeit (Multicastgruppenadressen, Aufbauen von Multicastbäumen, ...). Die Transportschicht der Internetfamilie hat mit TCP ein verbindungsorientiertes und mit UDP ein verbindungsloses Protokoll. Allerdings sind beide, wie oben schon erwähnt, in dieser Form nicht für den Multicastverkehr geeignet. Hier setzt SCE ein, wobei die bereits im Protokoll für zuverlässigen Unicastverkehr vorhandenen Funktionen weitergenutzt werden sollen. Allerdings werden sie durch die Einführung der SCE-Layer erweitert, um das Protokoll an die Multicastfähigkeit der Netzwerkschicht anzupassen. SCE fasst die Empfangsbestätigungen sowie andere Kontrollpakete zusammen und reicht sie an die überlagernde verbindungsorientierte Transportschicht weiter. Außerdem kann SCE direkt mit der Anwendungsschicht kommunizieren, um spezifische Multicastvariablen zu überwachen. Aus Sicht von Schicht 4 (Anwendungsschicht) bietet die erweiterte Architektur jetzt eine zuverlässige Transport-Multicastschnittstelle.

Der für die Anwendungsschicht gebotene zuverlässige Multicast-Transportdienst (Reliable Multicast Transport Service, RMTS) hat eine Schnittstelle zu den Diensten des bereits exis-

tierenden Transportprotokolls und eine weitere zu den Multicast-Diensten der SCE-Schicht. Der hier verwendete verbindungsorientierte Datenaustausch erfordert den Aufbau einer Verbindung und nach dem Datentransfer wieder den Abbau. Der Verbindungsaufbau ist beim Unicast-Verkehr erfolgreich, wenn die Verbindung von der gegenüberliegenden Seite akzeptiert wird. Dies ist bei SCE schwieriger, da es typischerweise viele Empfänger gibt. Die Entwickler haben hier mehrere Möglichkeiten für die Akzeptanz einer Verbindung offengelassen. Eine Möglichkeit ist, dass mindestens n Empfänger die Verbindung annehmen (n ist eine vorgegebene Zahl). Die Adressierung der Empfänger erfolgt wie bei TCP über die IP-Adresse und die Transportschichtadresse (normalerweise Port-Nummer).

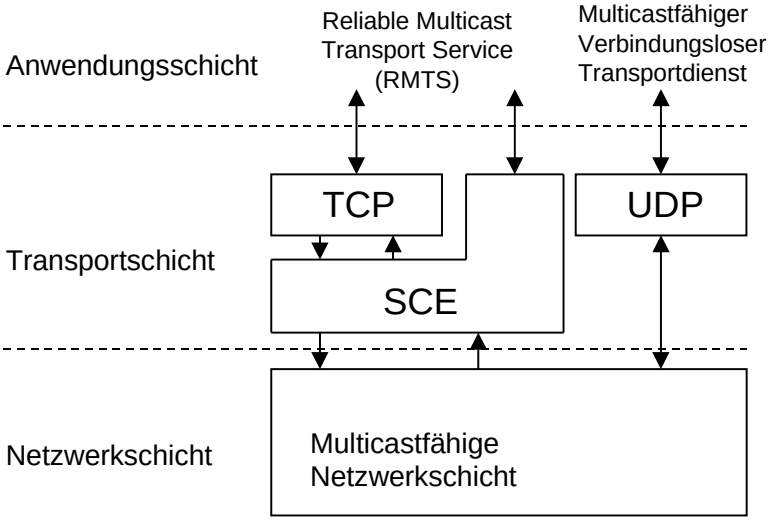

Abbildung 5: Netzwerkarchitektur bei Benutzung von SCE

Ist die Verbindung aufgebaut bekommt der Sender über Kontrollpakete den jeweiligen Status der Empfänger mitgeteilt. Hierzu hält SCE für jeden Teilnehmer spezifische Daten in einer Datenbank. Die Datensätze beinhalten unter anderem die Startsequenznummer, die Fenstergröße oder den Wert der letzten gesendeten Empfangsbestätigung. Anhand der empfangenen Antworten wird die Datenbank ständig aktualisiert. Des weiteren werden verbindungsspezifische Daten, wie Zieladresse oder Portnummer gespeichert.

Sendet die Quelle ein Paket wird es über die Transportschicht an die Single Connection Emulation Layer weitergeleitet. Von hier geht es über die Netzwerkschicht per Multicast an die Gruppe weiter. In umgekehrter Richtung sammelt SCE alle Empfangsbestätigungen ein. SCE wartet eine bestimmte Zeit und sendet nach deren Ablauf, falls eine angemessene Zahl an Empfangsbestätigungen eingegangen ist, eine einzelne Empfangsbestätigung an die Transportschicht. Sind zu wenig Bestätigungen eingegangen wird nichts an die Transportschicht weitergereicht. Somit läuft bei TCP der Timer ab und TCP geht davon aus, dass die Übertragung nicht erfolgreich war. Es wird erneut ein Sendeversuch unternommen. Hier wird deutlich, dass TCP die Aufgabe der Wiederübertragung von verlorengegangenen oder fehlerhaften Paketen übernimmt. Dieser Mechanismus funktioniert nur, wenn die Retransmission-Timer-Zeit von TCP größer ist als die Ablaufzeit des SCE-Timers. Sonst würde TCP immer mit der Wiederübertragung von Paketen beginnen, obwohl diese in Wirklichkeit korrekt bei den Empfängern ankommen.

In typischen Multicast-Anwendungen fließen die Informationen vom Sender zu den Empfängern. Je nach Anwendung haben die Empfänger das Bedürfnis bei der Konversation Informationen zurückzusenden. Dies ist in SCE nicht möglich, da über RMTS nur in eine Richtung Informationen weitergegeben werden können. Lediglich Kontrollinformationen können zurückfließen. Die einzige Möglichkeit um Daten zum Sender zu übertragen ist der Aufbau

einer separaten Unicast-Transport-Schicht-Verbindung. Ein weiteres Defizit, aus TCP resultierend, ist die fehlende Gruppendynamik. Normalerweise wird es den Empfängern ermöglicht während einer Anwendung der Gruppe beizutreten oder sie zu verlassen. In SCE ist das spätere Beitreten zu einer Session aufgrund des TCP-basierten Verbindungsaufbau nicht mehr möglich. Allerdings besteht die Möglichkeit eine Gruppe zu verlassen. Hierbei kann der Sender bzw. SCE das Austreten eines Gruppenmitgliedes erkennen, falls er eine gewisse Zeitspanne keine Empfangsbestätigungen erhält. Es besteht aber somit die Gefahr, dass ein Empfänger gar nicht die Gruppe verlassen wollte, sondern Pakete über längere Zeit durch Netzstau nicht ankommen konnten.

In der Praxis hat SCE momentan und auch in der Zukunft keine Bedeutung. Dies liegt an den oben schon aufgeführten Gründen und an der fehlenden Skalierbarkeit. Wird die Gruppe zu groß führen die Empfangsbestätigungen beim Sender zwangsweise zu einer Quittungsimplosion. Positiv gilt nochmals hervorzuheben, dass SCE keine neuen Protokolle einführt und keine Veränderungen an den existierenden Internetprotokollen (TCP/IP) vornimmt, sondern nur eine Art Zwischenschicht einschiebt.

3.6 STORM (STructure-Oriented Resilient Multicast)

Mit dem Protokoll STORM wird ein neues Modell des Multicasting vorgeschlagen, das sich *resilient-multicast* nennt. In diesem Modell hat jeder Empfänger die Flexibilität zwischen Echtzeitanforderungen (Real-Time) und Zuverlässigkeit zu wählen.

Bei der Übertragung von kontinuierlichen Datenströmen (continuous media), für dessen Zweck das Modell hauptsächlich gedacht ist, machten die Entwickler zwei wesentliche Beobachtungen. Zum einen hat in einer Anwendung das Maß an Interaktivität eine hohen Einfluss, da eine Anwendung, die eine hohe Interaktivität hat, die Zustellung der Pakete in Echtzeit verlangt. Daraus folgt, dass Pakete mit einer hohen Verzögerung verworfen, sowie verlorene Pakete nicht wiederübertragen werden können. Im Gegensatz dazu gibt es andere Anwendungen mit wenig Interaktivität, z.B. Ausstrahlung von Live-Übertragungen. Bei der zweiten Beobachtung stellte man fest, wie unterschiedliche Teilnehmer der gleichen Anwendung auch unterschiedliche Anforderungen an die Abspielqualität und die Möglichkeit zur Interaktivität haben. Beispielhaft sei hier die Übertragung einer Diskussion erwähnt, bei der es aktive und passive Teilnehmer gibt. Während die aktiven Teilnehmer eher Echtzeitanforderungen haben, damit sie zu jedem Zeitpunkt in die Diskussion eingreifen können, sind passive Zuhörer dagegen mit einer höheren Zuverlässigkeit besser bedient. Für sie ist es angenehmer, wenn die Daten mit einer gewissen Verzögerung ankommen, dafür aber in besserer Qualität.

Die Herausgeber des Resilient-Modells untermauern die Notwendigkeit für Flexibilität (resilience) mit einigen Argumenten. Bei einer Anwendung, die z.B. ein Whiteboard (wb) benutzt müssen alle Pakete ankommen, während speziell bei kontinuierlichen Medien, Echtzeitanforderungen verlangt sind. Die Datenrate von kontinuierlichen Medien ist relativ hoch und im Normalfall gleich, bei Whiteboard-Applikationen werden dagegen burst-artige Datenströme, aber insgesamt weniger Daten versendet. Auch hält im wb-Fall jeder Teilnehmer die erhalten Daten im Speicher, da der Sender zu vorherigen Seiten springen kann. Somit ist natürlich auch jedes Gruppenmitglied in der Lage Daten, die ein anderes Mitglied nicht bekommen hat zu übertragen. Bei kontinuierlichen Medienströmen ist dies nicht der Fall, die Daten werden nach dem Empfang verworfen.

STORM stellt eine erste Implementierung dieses Modells dar, mit den folgenden zwei Schlüsselideen:

- Gruppenteilnehmer organisieren sich selbst in einer verteilten Struktur und benutzen diese, um nicht empfangene Pakete von Nachbarn zu bekommen.

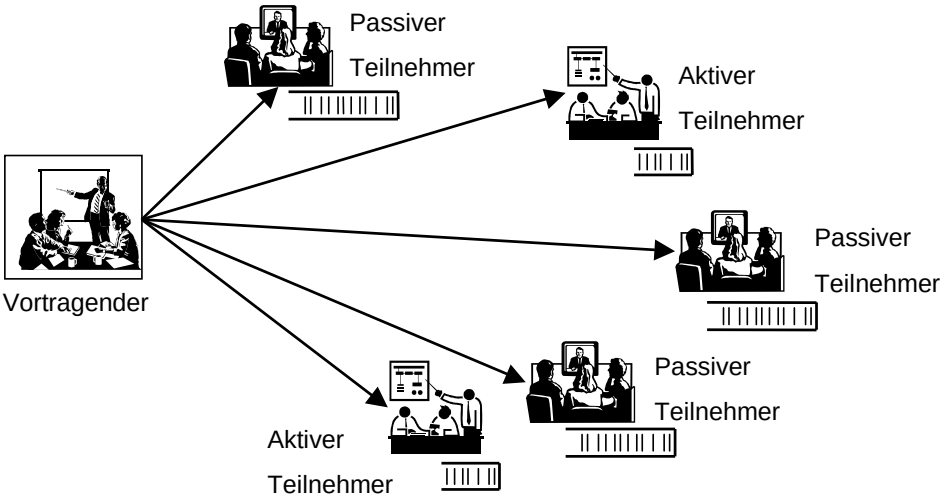

Abbildung 6: Beispiel für einen Vortrag mit mehreren Teilnehmern, wobei die aktiven Teilnehmer einen kleineren Datenspeicher haben (Interaktivität) und die passiven Teilnehmer einen größeren (Zuverlässigkeit)

- Diese verteilte Struktur selbst ist dynamisch.

Das Protokoll wurde für kontinuierliche Medien konstruiert (Audio und Video), bei denen wie oben schon erwähnt ein geringer Verlust an Daten tolerierbar ist. Jeder Empfänger muss einen Speicher haben, in dem er die abzuspielenden Daten kurzfristig vorrätig hält. Treffen nun Daten nach einem Stopzeitpunkt (deadline) im Speicher ein, sind sie nutzlos, da sie schon abgespielt werden sollten. In STORM kann die Größe des Speichers von jedem Empfänger unabhängig bestimmt werden. Es wird bei der Verwendung eines großen Speichers mehr Wert auf Zuverlässigkeit gelegt, bei Benutzung eines kleineren Speichers hingegen mehr Wert auf Interaktivität.

Die in STORM verwendete Baumstruktur (Recovery-Struktur) um fehlende Pakete zu bekommen unterscheidet sich von der anderer Transportprotokolle. Es gibt nicht wie im Multicast-Routing-Baum einen Unterschied zwischen inneren Knoten und äußeren Knoten. Hier sind alle Knoten Endpunkte, wobei jeder Knoten mehrere Väter hat. Der Vater selbst ist wiederum ein anderer Empfänger oder der Sender selbst. Die Recovery-Struktur wird aufgebaut sobald Empfänger zur Anwendung hinzutreten. Hierzu verwendet ein Empfänger eine Ringsuchtechnik (expanding-ring-search, ERS), mit der er in frage kommende Väter ausfindig macht. Diese ERS-Requests werden periodisch per Multicast versendet, wobei die „Time to live“-Zeit mit jedem gesendeten ERS-Request zunimmt. Gruppenmitglieder können per Unicast auf die Anfragen antworten. Anhand dieser Informationen kann ein Empfänger geeignete Väter ausfindig machen. Ein geeigneter Vater hat zwei wichtige Eigenschaften: Zuerst besitzt er die Möglichkeit angeforderte Pakete zeitgerecht auszuliefern. Zum anderen besteht eine geringe Korrelation zwischen den verlorenen Paketen des Vaters und des Sohnes. Jeder Empfänger prüft in gewissen Abständen die Qualität seiner Väter und sucht sich bei Bedarf andere aus. Somit ergibt sich für STORM eine dynamische Recovery-Struktur.

Die jeweiligen Vaterknoten müssen also am Anfang von jedem Empfänger bestimmt werden. Die Auswahl ist abhängig von den eigenen Anforderungen an die Verzögerung. Wenn ein neues Mitglied der Gruppe beitritt, wählt es die Väter anhand deren Paketverlustrate aus. Stellt ein Empfänger eine Lücke in seinen Paketsequenznummern fest, sendet er per Unicast eine negative Quittung baumaufwärts zu einem Vater. Hat ein Vater eine negative Quittung erhalten und die angeforderten Daten selbst schon erhalten sendet er sie an den Sohn per Unicast.

Jeder Knoten muss zu diesem und anderen Zwecken eine Liste mit folgenden Parametern haben: Tabelle mit Vaterknoten, Qualitätsabschätzung für jeden Vaterknoten, ein Verzögerungshistogramm über alle erhaltenen Pakete, eine bestimmte Anzahl von Timern für Repair Requests, die an Väter verschickt wurden und eine Liste mit Söhnen, für die noch ausstehende Repairpakete geliefert werden müssen. Erhält ein Sohn nach einer Timeout-Zeit keine Antwort auf seine negative Quittung, verschickt er wieder eine negative Quittung, allerdings an einen anderen Vater der Liste. Dies geht solange bis der Sohn das fehlenden Paket erhält, oder das Paket veraltet ist und keine Verwendung mehr hat. Das Ziel der Verwendung von mehreren Vätern, ist das Erreichen einer ausgeglichenen Recovery-Struktur, die auch relativ robust ist.

STORM bietet Möglichkeiten zur Anpassung von Zuverlässigkeit bzw. Interaktivität an die individuellen Wünsche. Besonderer Wert wird auf das schnelle „Reparieren“ von fehlenden Paketen durch die Recovery-Struktur gelegt.

4 Zusammenfassung

Bis zum heutigen Tag konnte sich kein Multicast-Transportprotokoll in der Praxis durchsetzen oder gar etablieren. Es ist damit zu rechnen, dass in Zukunft kein einzelnes Protokoll existieren wird, sondern ein Pool von Protokollen. Dies liegt an den vielen unterschiedlichen Anforderungen, die einzelne Dienste haben. Zum Schluss soll die nachfolgende Tabelle alle in diesem Papier vorgestellten Multicast-Transport-Protokolle gegenüberstellen.

Protokoll	Zuverlässigkeit	Hauptanwendungsgebiete	Recoverystruktur	Probleme
UDP	–	Video, Audio	Sender verantwortlich	verbindungsloses Protokoll
RMTP	+	Verteildienste	lokale RS	Momentan fehlende Funktionen in Routern
SRM	+	Whiteboard	lokale RS	Skalierbarkeit
MFTP	+	File-Transfer	Sender verantwortlich	ineffiziente Fehlerbehandlung
SCE	*o*	Bis jetzt keine	Sender zuständig	Quittungsimplosion
STORM	*o*	Video, Audio	lokale RS	

Tabelle 1: Multicast-Transport-Protokolle in der Übersicht

Literatur

[FJLM+97] Floyd, Van Jacobson, Liu, McCanne und Zhang (Hrsg.). A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. Transactions on networking, IEEE/ACM, Dezember 1997.

[HoHä98] Markus Hofmann und Christoph Hänle (Hrsg.). Performance Comparison of Reliable Multicast Protocols using the Network Simulator ns-2. Proceedings of ieee conference on local computer networks (lcn), IEEE, 1998.

[KrRe00] Gerhard Krüger und Dietrich Reschke. *Lehr- und Übungsbuch Telematik*. Fachbuchverlag Leipzig. 2000.

[PaSa96] Sanjoy Paul und Krishan K. Sabnani (Hrsg.). Reliable Multicast Transport Protocol (RMTP). Ieee infocom'96, IEEE, 1996.

[ZiWi99] Martina Zitterbart und Ralph Wittmann. *Multicast - Protokolle und Anwendungen*. dpunkt.verlag. 1999.