Authenticated Quality-of-Service Signaling for Virtual Networks

Roland Bless, Martin Röhricht, and Christoph Werle
Institute of Telematics
Karlsruhe Institute of Technology (KIT)
P.O. Box 6980, D–76049 Karlsruhe, Germany
Email: {bless, roehricht, werle}@kit.edu

***Abstract*—While virtual networks have been subjected to detailed analysis, prototypes are usually constructed and instantiated manually or by means of dedicated control protocols that mostly neglect security considerations. As the instantiation and maintenance of virtual networks requires virtual network operators to gain access to existing network resources, a proper sender authentication builds an important aspect for control protocols. In this work, we present a Virtual Link Setup Protocol (VLSP) that is designed as a modular extension to a standardized state-of-the-art signaling protocol suite. We use these signaling protocols to combine an authenticated and on-demand setup of virtual links with the establishment of Quality-of-Service guarantees in the underlying substrate. The solution presented in this paper is not limited to a specific set of virtualization techniques or tunneling mechanisms. We describe the design and implementation of VLSP and evaluate its signaling performance, as well as the overhead that is associated with the instantiation of the virtual links and the authenticity checks.**

I. Introduction

Network virtualization is a promising abstraction technique that allows for optimization of network resource utilization by enabling the concurrent isolated operation of multiple virtual networks on top of a shared physical network infrastructure (the 'substrate'). Furthermore, virtual networks can be used to test and deploy novel network architectures and protocols, which presents a viable approach towards the design and deployment of a future Internet. For these reasons, there has been tremendous interest in network virtualization over the past few years and many large research initiatives, e.g., the Global Environment for Network Innovations (GENI) [2], the NSF NeTS FIND Initiative [3], or the AKARI Architecture Design Project [4], have examined the subject closely. Within these research projects, various prototypes for network virtualization architectures have been developed and evaluated. Many approaches mainly concentrate on providing an experimental facility for network researchers to test novel network architectures based on new protocols. Contrastingly, the 4WARD project [5] of the EU 7th Framework Programme focused on a more general network virtualization architecture. Aiming to reflect real-world business models, the 4WARD network virtualization architecture considers different business roles and various types of providers, which we are going to present in detail in the next section. In this context, an important aspect was to consider the creation of virtual links with quality-of-service (QoS) guarantees *across* different substrate Infrastructure Provider (InP) *domains* in order to build larger or even global virtual networks.

In this work, we present a signaling protocol for the dynamic setup of virtual links with QoS guarantees. We tightly couple the setup of virtual links with an optional QoS reservation and additionally enable authentication of the involved signaling messages. This permits to verify whether the initiator of the virtual link setup request is actually allowed to do so in accordance with the local policy and, if so, to reserve the substrate resources associated with this virtual link. As a common control plane for the deployment of virtual networks on a global scale, we assume an IP-based substrate in conjunction with the IETF Next Steps in Signaling (NSIS) framework as an up-to-date IP-based signaling protocol suite. Our solution is independent of a particular system virtualization technology and can be used with, e.g., XEN or KVM.

The remainder of this paper is organized as follows. In Section II we give a brief overview of the key goals of network virtualization. We provide a description of the 4WARD network virtualization architecture and state the resulting requirements to setup virtual links on demand. We then describe the design and realization of an authenticated Virtual Link Setup protocol that fulfills these requirements in Section III. The evaluation of the proposed Virtual Link Setup Protocol with regard to signaling performance is presented in Section IV. We outline some related work in the context of this paper in Section V before we conclude in Section VI.

II. Network Virtualization

Virtual networks basically consist of two types of components, *virtual nodes* and *virtual links*. Virtual nodes appear and act like physical network nodes, i.e., they are interconnected by various (virtual) links, have access to a specific set of resources, run a (network) operating system and other software. But instead of running

This is an extended and reorganized version of a paper of the title "Authenticated Setup of Virtual Links with Quality-of-Service Guarantees" that appears in the proceedings of ICCCN 2011, Maui, HI, USA [1]. Manuscript received February 15, 2011; revised May 15, 2011; accepted June 15, 2011.

doi:10.4304/jcm.7.1.17-27

directly on dedicated physical hardware, they only run in a virtual environment, where a *virtualization layer* provides access to a set of virtual resources. In general, a virtual resource appears to a user of that resource as if she is the (exclusive) owner of that resource. These virtual resources are composed of logical and physical resources—e.g., CPU, memory, process table, or memory buffers—that do not necessarily directly correspond to the resources of the physical system that hosts the virtual node. This allows a physical node to host multiple virtual nodes simultaneously. Hence, physical resources can be shared between different virtual nodes which may impact the system's performance (e.g., the speed to forward packets) and the isolation between virtual nodes and virtual networks.

There are many different host virtualization technologies that can be used to realize virtual nodes based on commodity PC hardware. The technologies can be categorized into full virtualization (e.g. Linux' Kernel-based Virtual Machine (KVM) or Oracle's Virtualbox), para-virtualization (e.g. XEN), and container-based virtualization (e.g. Linux VServer or OpenVZ). However, a virtual network should conceptually operate independently from the actual virtualization technology in use. Moreover, dedicated router hardware may also provide corresponding virtualization support.

Virtual links are used to interconnect virtual nodes and may likewise be realized by a variety of different substrate technologies, e.g., through VLANs, MPLS, or, for IP-based substrate networks, by one of the existing IP tunneling techniques like IP-in-IP tunnels [6], Generic Routing Encapsulation (GRE) [7], or Layer 2 Tunneling Protocol (L2TP) tunnels [8]. In some cases the virtual link may be even realized by internal communication via memory, e.g., if two virtual nodes are hosted on the same physical host.

Virtual networks may be administrated and operated in a wide variety of ways. In order to allow for a clean separation of responsibility and control, the 4WARD architecture considers three different entities with corresponding interfaces as illustrated in Figure 1.

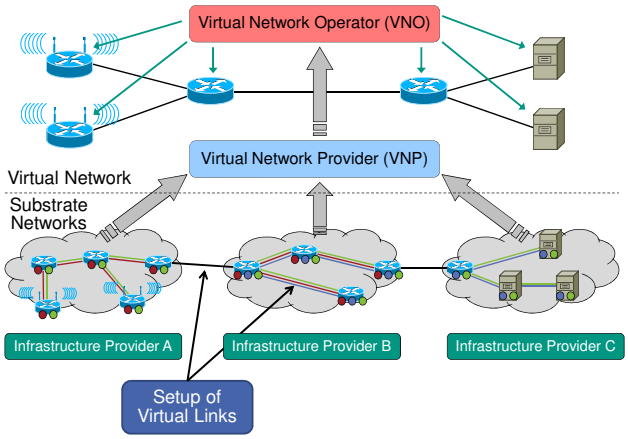

Figure 1. The Network Virtualization Business Model of the 4WARD architecture

Within this framework an *Infrastructure Provider* (InP) is responsible for the operation and management of the actual physical resources that are located in the physical substrate. It can partition its resources into a set of slices that will be made available for the virtual networks and also re-arrange the actual location of its virtual resources, e.g., migrate a virtual node between substrate nodes. The mapping of virtual resources onto substrate resources is usually defined and governed by the InP. Therefore, the InP is able to allocate and create virtual resources such as virtual nodes and virtual links. This article focuses on the on-demand creation and setup of virtual links by InPs. An InP also provides a resource control interface for its virtual resources, so that the entity who operates a virtual resource can perform some basic control functions, e.g., install, reboot, or halt a virtual node.

A *Virtual Network Provider* (VNP) acts as a customer of different InPs and is responsible for an on-demand instantiation of virtual networks by means of the offered resources of these InPs. It therefore creates virtual topologies that can also be used recursively by or in cooperation with different VNPs to span an even larger virtual network.

The *Virtual Network Operator* (VNO) fulfills the role of a network operator just as in the traditional way but in this case based on the virtual network that is provided by the VNP. The VNP acts as a resource broker for the VNO and assembles the virtual network from resources of possibly multiple InPs. The VNO can then maintain and control the virtual resources in order to offer specific services to its customers.

The reason for the introduction of an intermediate VNP is to allow VNOs the instantiation of large virtual networks that reach from end to end across multiple InPs without having the VNOs to establish business relationships with every involved InP. Therefore, this architecture provides a clean separation of concerns and responsibilities. However, note that these roles can also be combined and interact in a wide variety of different ways. For instance the roles of VNP and VNO can be performed by the same entity. The 4WARD architecture describes an exemplary way of how a virtual network architecture can be constructed and should not necessarily be considered to be the only solution. The solution outlined in this paper is intended to be used at InP level, either inside a single InP or also between different InPs.

It is a key goal of network virtualization to enable the efficient use of resources, which can be achieved by sharing resources between concurrent virtual networks. In order to inhibit virtual networks from affecting each other adversely, it is the responsibility of the underlying substrate to construct, monitor, and maintain these virtual resources and to provide them with a deterministic degree of mutual isolation. A virtual link should behave very similarly to a physical link, i.e., providing guaranteed QoS parameters such as throughput, delay, jitter, packet loss rate, and packet error rate. Therefore, the provisioning of *Quality-of-Service guarantees* and a corresponding

admission control must provide an integral component of any comprehensive network virtualization framework offering virtual links with guarantees.

Since the elasticity of virtual resources is a major advantage of virtual networks, they must also support quick instantiation of virtual resources. Thus manual configuration at scale is not feasible but requires a robust and flexible signaling protocol that allows for an automated *on-demand setup of virtual links*.

Many network virtualization architectures consider the setup of virtual networks across different substrate domains. While the authentication of signaling traffic is already desirable in an intra-domain setting, it becomes indispensable in an inter-provider setting to fend off the setup of virtual links due to forged or tampered signaling messages. Therefore, it must be ensured that the setup of virtual links and the reservation of corresponding resources in the substrate is properly *authenticated*. Authentication of signaling messages means that there exist mechanisms so that both the identity of the virtual link setup request's initiator and the integrity of a signaling message—or at least specified parts of it—can be verified.

Figure 2 shows an exemplary setting in which two virtual nodes (VM_1 and VM_2) have already been created in an inter-provider setting and are running on Router A and Router C in the presence of other virtual nodes that may be interconnected arbitrarily. Router B is a pure substrate router without network virtualization support, i.e., it does not host virtual nodes but supports QoS reservations in the substrate. Similarly, virtual links are handled transparently by Router B and it is not aware of a traversing virtual link. To Router B, the virtual link simply looks like an ordinary data flow between Router A and Router C.

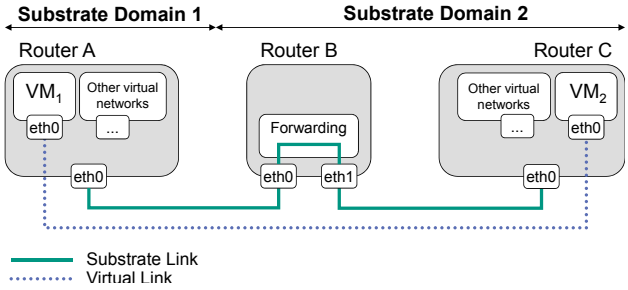

Figure 2. Basic network virtualization example

The request to interconnect the two virtual nodes is triggered by an abstract management entity, which initiates the signaling procedure. To fulfill the previously motivated requirements, i.e., to interconnect the two virtual nodes with a QoS-provisioned virtual link in an authenticated manner, we need

- a description of the QoS requirements for the virtual link
- the locators of the substrate end points of the virtual link
- the identifiers of the virtual link's endpoints, e.g., identifiers of the virtual nodes and their related virtual interfaces
- the (de-)multiplexing method of the virtual link (e.g., a tunnel type, like L2 Tunnel, IP in IP, GRE, ...)
- a key infrastructure that allows the computation of a message authentication code or a digital signature for the signaling messages

With these requirements in mind, the next section discusses the design and realization of such a virtual link setup protocol. For the remainder of this paper we assume an underlying IP-based substrate, since IP is currently the least common denominator on a global scale and a common substrate technology is a precondition to deploy and operate virtual networks across different InP domains.

III. Virtual Link Setup Protocol

In this section, we propose a *Virtual Link Setup Protocol* (VLSP) that permits an authenticated and dynamic setup of virtual links with dedicated Quality-of-Service guarantees. After successful authentication, the VLSP allocates the required resources along the substrate path and connects the virtual link's ends to the virtual nodes' interfaces. Virtual nodes themselves are not aware of the signaling and do not need to run the signaling application. The setup of the virtual link takes place in the substrate and is coordinated by the signaling control entities running solely on the involved substrate nodes.

For the setup of virtual links, the following steps must be performed:

1) Both infrastructure providers, each operating its substrate domain and hosting involved virtual nodes, must acquire the substrate addresses of the opposite end of the virtual links.
2) The substrate addresses are then used by the signaling control entity to establish a virtual link with Quality-of-Service guarantees while verifying the signaling messages' authenticity. The path-coupled signaling that the VLSP uses by default ensures that a feasible substrate path exists between the substrate nodes hosting the virtual nodes.
3) Resource reservation along the substrate path is performed by means of the corresponding Resource Management Functions (RMF) located inside the substrate nodes.
4) Signaling must reach the opposite substrate node's control plane in order to install state for the virtual links.
5) The final step consists of the involved RMFs at the endpoints actually installing state required to connect the substrate tunnel end to the virtual link end (e.g., network interface of the virtual node) and bringing up the virtual link.

A. Quality-of-Service Signaling for Virtual Links

In order to perform Quality-of-Service signaling for virtual links, we rely on the IETF's *Next Steps in Sig-*

naling (NSIS) framework [9], which provides an up-to-date IP-based signaling protocol suite and can be used for a variety of signaling applications. Even though QoS signaling can already be accomplished by means of the QoS NSLP protocol [10], the current NSIS framework does not provide support to setup virtual links. As discussed previously, virtual links need to be tightly coupled with the provisioning of QoS guarantees in order to allow concurrent operation of mutually isolated virtual networks.

Therefore, we integrate the QoS signaling in the substrate with the setup of virtual links and extend the existing QoS NSLP protocol to include signaling information for virtual links. This approach provides the following two advantages: First, the integration of both tasks reduces the signaling time required in comparison to two distinct signaling operations and second, by using the QoS NSLP as a basis, we do not need to create an entirely new NSIS signaling layer protocol that would otherwise provide a near-identical set of features.

Infrastructure providers must agree on a common method and signaling protocol for setting up virtual links across different substrate InP domains. We therefore assume that InPs agree a priori (e.g., negotiated out-of-band via peering agreements) to use NSIS with the VLSP extension for setting up virtual links between substrate domains and to run NSIS on the involved substrate nodes. Intermediate domains may support QoS NSLP only as the VLSP extension is only required on virtual node hosting substrate nodes.

Figure 3 gives a conceptual overview of the NSIS protocol architecture and shows the incorporated VLSP object as an extension of the QoS NSLP. The Session Authorization object is an optional security object that can be employed by any NSLP.

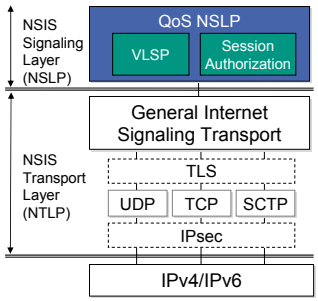

Figure 3. Conceptual overview of the NSIS protocol architecture with the VLSP object and the Session Authorization object

The lower layer, called NSIS Transport Layer Protocol (NTLP), is responsible for the correct routing and transport of signaling messages between two adjacent NSIS nodes and employs already existing transport protocols like UDP, TCP, TCP with TLS, or SCTP. The General Internet Signaling Transport (GIST) [11] protocol fulfills the requirements of an NTLP.

The upper layer, called NSIS Signaling Layer Protocol (NSLP), implements the signaling application logic and operates either from end to end, from edge to end, or from edge to edge. The QoS NSLP [10] is a soft-state protocol that reserves resources along a data path and installs resource reservation state in nodes on this path accordingly. It conveys a dedicated QoS Specification template (QSPEC) [12] object for specifying QoS parameters. Therefore, it abstracts from the actually used QoS mechanisms on the data path like IntServ or DiffServ. For instance each InP domain must determine autonomously how QoS requests can be met by the actually employed QoS mechanisms inside the domain, e.g., which DiffServ class is eligible for conveying the requested data flow.

The QoS NSLP uses *path-coupled signaling* by default, which proves especially advantageous for QoS resource reservations as it allows to install state in exactly those nodes that belong to the data flow's path. Furthermore, it is assured that a working path exists and the signaling path is automatically adapted in case re-routing events occur. Note that path-coupled signaling works for any tunneled solution, where the outer tunnel IP destination address is used as destination address for the signaling messages that discover the signaling path.

B. Authenticated Setup of Virtual Links

One of the most important requirements when dealing with resource reservations and the establishment of virtual links is the ability to authenticate legitimate requests and to protect the integrity of critical parts of a signaling message. The NSIS protocol suite already provides basic security mechanisms inside the NTLP layer, e.g., by employing TCP/TLS for messaging associations between signaling peers. But this protection is limited to adjacent signaling peers and cannot provide a per-session or per-user protection of signaling message. The latter is required if authorization decisions depend on a user identity rather than on a node identity. Therefore, the NSIS framework provides an optional so-called *Session Authorization Object* [13] that provides a means to authenticate NSIS signaling messages on a per user or per session basis [14] at NSLP level. The object may carry authentication tokens and can be used for integrity protection of NSLP messages, too. The latter feature was suggested by the authors and directly contributed to [13].

In order to be used for an authenticated on-demand setup of virtual links, the Session Authorization object provides the following information elements: an authorizing entity identifier, start and end time of the authorized session, a list of identifiers for all the objects of this NSIS message that are covered by the signature data, and the signature data itself. The integrity protection can be ensured by means of shared symmetric keys, Kerberos authentication, public key authorization via X.509, or PGP certificates.

Figure 4 shows an exemplary binding of a Session Authorization object and the NSIS message objects. Grey-shaded objects are included into the signature data's calculation in order to be integrity-protected, such as the Session ID, the Message Routing Information, or the QSPEC objects, for instance. Furthermore, the signature

data covers the new VLSP object that is used to setup virtual links, and important parts of the Session Authorization Object itself, such as the ID of the authorizing entity, an ID for the used hash algorithm, the aforementioned list of signed objects, or a key-ID to identify the corresponding signature key. Other parts of the signaling message that are subject to change during transition must not be included into the integrity protection.

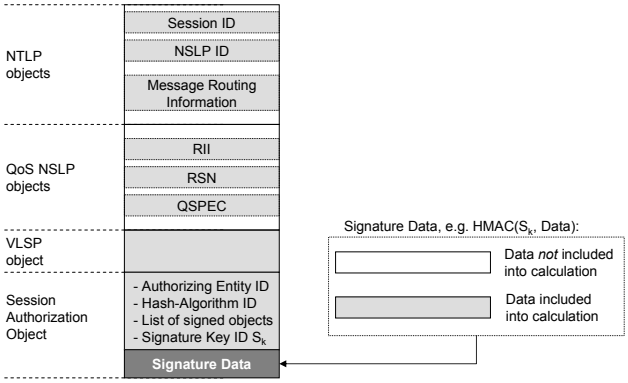

Figure 4. Binding of Session Authorization Object and NSIS message objects

Note that it is also possible to transport more than just one Session Authorization Object with each NSIS message. This allows for the authentication and integrity protection of NSIS messages even across different administrative domains.

C. Implementation Overview

For the setup of virtual links, we extended the existing NSIS QoS NSLP by an optional NSLP object. The newly defined *Virtual Link Setup Protocol* object can be added to QoS NSLP's RESERVE and RESPONSE messages and carries the following additional information:

- *Virtual Network ID:* An identifier for the virtual network for which the virtual link is created and which we assume to be globally unique
- *Virtual Node IDs* of the source and the destination nodes identifying the virtual nodes within the scope of a virtual network
- *Virtual Interface IDs* of the source and the destination nodes, which have only node-local meaning
- *Virtual Link ID* (optional)
- Opaque descriptor of the (De-)Multiplex method, currently the tunnel type.

We used 128 bit identifiers for the Virtual Network ID and the Virtual Node IDs, 64 bit identifiers for the Virtual Interface IDs and the Virtual Link ID, and a 32 bit Substrate Tunnel Type as depicted in Figure 5. Consequently, this object occupies 80 bytes (including the necessary NSLP object header) of a 240 bytes QoS NSLP RESERVE message. The overhead in the data plane depends only on the used tunneling mechanism.

The addressing information carried in the VLSP object enables the endpoints of the virtual link to connect the

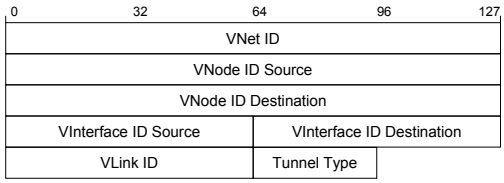

Figure 5. Conceptual overview of the VLSP object

substrate link ends (e.g., tunnel ingress/egress) to the virtual link ends. Since the VLSP object information is only relevant for the virtual link ends, these objects can be safely ignored by intermediate nodes, which pass them on unmodified. Intermediate nodes therefore check the Message Routing Information for the source and destination address of the tunnel flow and do not process the VLSP object, since they are not the end of the virtual link. Furthermore, the VLSP object can be introduced in a backwards compatible fashion by using the NSLP object extensibility flags. These flags are set to 'Forward' in order to instruct intermediate NSIS entities that are not aware of this newly introduced object to pass this object unmodified when forwarding the message. Consequently, NSIS nodes that do not support the VLSP object, are neither confused nor impede deployment of the VLSP.

Figure 6 illustrates the message sequence of a virtual link setup procedure. The request starts with a QoS NSLP RESERVE message that additionally carries a VLSP object to setup the virtual link and a session authorization object to authenticate the request and to protect the integrity of the signaling message. This initial request is directed towards the destination, i.e., Router C in this case. An intermediate NSIS capable router, Router B, intercepts and interprets the signaling message, upon which it performs admission control for the Quality-of-Service request, but ignores the VLSP object as it is not declared as the virtual link's destination. Router B then forwards the possibly adapted resource reservation request with the original VLSP object and the session authorization object.

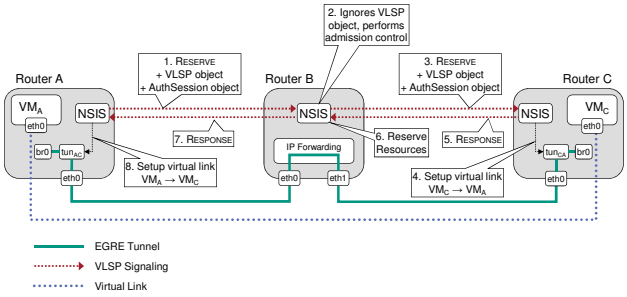

Figure 6. Exemplified scenario of a virtual link setup procedure

Once the signaling message reaches its designated destination, the authenticity and integrity of the message is checked upon which the resource reservation request is processed by the host's resource management function and the virtual link is set up according to the information contained in the VLSP object. This includes attachment of the tunnel end to the virtual interface of the virtual ma-

chine. After that Router C sends a QoS NSLP RESPONSE back to Router B, which finally allocates the admitted resources as indicated by the response, and Router B sends a QoS NSLP RESPONSE back to Router A. Router A can then allocate the necessary resources and establish the virtual link on its side.

We note that the resource reservation is usually unidirectional only, because the substrate routes from A to C and from C to A can actually differ. Since virtual links are usually used for bidirectional communications, the QoS reservation should also be established in both directions. Therefore, we assume that Router A initiates the setup of the direction A $\leftrightarrow$ C and Router C initiates the setup of the direction C $\leftrightarrow$ A. For instance, the QoS NSLP responder, Router C in our example, can optionally initiate a resource reservation in the opposite direction and use the QoS NSLP's Bound Session ID object to create a logical binding between both reservations. The additional RESERVE and RESPONSE messages for this bidirectional reservation are not shown in Figure 6.

The QoS NSLP uses soft states for its resource reservations, thus it refreshes the state by periodically sending RESERVE messages. This accommodates not only for route changes in the substrate, but it can also be used by the virtualization layer to detect failures of virtual links, thereby indicating link breakage to the management and control plane.

IV. EVALUATION

In this section, we evaluate the signaling performance of our Virtual Link Setup Protocol. In addition to the measurements of the overall duration of the virtual link setup—from the initiation of a VLSP request until completion—we performed more fine-grained evaluations regarding the overhead induced by the creation and verification of the session authorization object, as well as by the creation of a tunnel for the virtual link. The herein described evaluation results differ slightly from those presented in [1], because all experiments were performed again with a slightly updated software revision and set up scripts. The most significant differences are related to execution of the local scripts, whereas the signaling related measurements are nearly the same.

Note that we do not measure throughput or forwarding performance as this is highly dependent on the used virtualization technology and it is out of scope for this paper to compare different virtualization technologies. Furthermore, we do not evaluate any QoS-related metrics in the data forwarding path as the VLSP acts only as a signaling protocol.

A. Experimental Setup and Measurement Methodology

We evaluated the proposed virtual link setup protocol in a testbed environment following the setup depicted in Figure 7. Each node consists of Intel Xeon X3430 quad-core CPUs running at 2.40 GHz, 4 GB RAM, and four Intel 82580 Gigabit Ethernet network interfaces, interconnected by a Cisco Catalyst Switch 6500 running CatOS. All nodes used an Ubuntu 10.10 server installation with a 2.6.35 Linux kernel as well as our freely available NSIS-ka implementation [15] at revision r6375. The latency between the endpoints was intentionally kept small (approximately 0.709 ms between tb1 and tb4 measured by 100 ping tests) in order to concentrate measurements on the pure protocol and processing overhead, i.e., no artificial delay was added.

Two kinds of measurement methods were used: code instrumentation using measurement points and network traffic traces (packet capture dumps) generated by tcpdump (pcap library). Fine-grained measurements were performed by putting measurement points into specific places within the code. Once such a reference point is executed, a timestamp is generated from the system clock and recorded in memory. After the entire experiment is finished, the recorded values are written into a file. This avoids the measurements to be affected from file I/O operations. Furthermore, packet capture dumps were made on tb1–tb4 for each of the interfaces. On tb2 and tb3 the dumps for eth0 and eth1 were merged for later analysis. We used a modified Wireshark version that supports NSIS protocols (also available from [15]) in order to extract the relevant NSIS protocol messages and wrote them as Comma Separated Value file for further processing by calculation scripts.

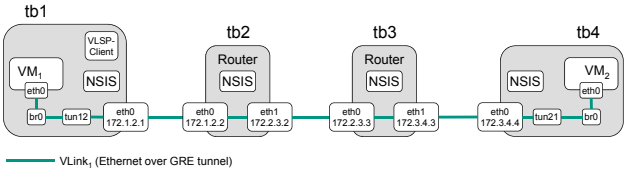

Figure 7. Evaluation setup with four Linux routers and two different virtual machines being connected through an EGRE tunnel

We decided to use Linux' Kernel-based Virtual Machine (KVM) for our tests as KVM is a well-tested and actively maintained virtualization solution that is directly integrated into the Linux kernel. As already mentioned above, the choice of a particular virtualization technology is conceptually independent from and opaque to the virtual link setup protocol. The virtual links are established by means of existing tunneling mechanisms, i.e., we used a Linux software bridge to interconnect each virtual interface with a tunnel endpoint. For evaluation tests, we used Ethernet over GRE (EGRE) tunnels, that are provided by the Linux kernel itself, between VM_1 and VM_2. Note again, that the outlined solution is generic enough to also support different types of tunnels in order to realize virtual links. We chose to use EGRE as this tunnel type allows for plain layer 2 connectivity between the tunnel endpoints, i.e., the virtual machine's virtual interfaces.

In order for all four nodes to support the QoS signaling and establishment of virtual links with QoS guarantees, each node ran an instance of the freely available NSIS-ka implementation [15] with disabled logging output. The VLSP extension as well as the Session Authorization

Object have been included the main trunk of the software so that the experiments described in this work can be repeated and verified by other parties.

Nodes tb1 and tb4 used the NSIS suite to establish a QoS reservation and setup the virtual links, whereas the intermediate nodes tb2 and tb3 were only involved in the QoS resource reservation but not in the interpretation of the VLSP object as described earlier.

B. Signaling Performance

We performed 100 separate runs to evaluate the signaling performance of our virtual link setup protocol. A detailed sequence of an experiment run is shown in Figure 8. First, an external program on tb1 issues a request to setup a virtual link between VM_1 and VM_2 via a UNIX Domain Socket interface towards the NSIS instance. After that, the NSIS signaling entity starts a VLSP request towards tb4. This signaling request consists of a GIST three-way handshake between each adjacent NSIS peer and corresponding QoS NSLP RESERVE messages carrying the VLSP object and Session Authorization Object. Once the RESERVE reaches tb4, it establishes its virtual link endpoint by calling a shell script (execution time period denoted as [C]). After successful creation of that tunnel, tb4 sends a RESPONSE back towards tb1. Once the NSIS instance on tb1 receives the RESPONSE (note time period [B]) it also establishes its virtual link's endpoint (calling a corresponding script for GRE tunnel setup, note time period [A]) upon which it finally informs the external program about the success of the operation.

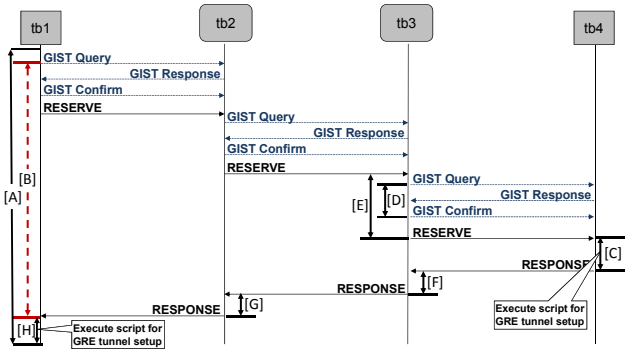

Figure 8. Message Sequence Diagram and Measured Periods

Figure 9 illustrates the duration of a virtual link setup request for 100 separate runs. The entire time from initiating a request until the external program receives the notification of a successful reservation and an established virtual link took 56.8 ms on average with a standard deviation of only 1.96 ms.

As outlined above, virtual links were established by calling a shell script from the NSIS-ka instance that performs the necessary Linux commands to setup an EGRE tunnel for this specific virtual link and connect it to the corresponding bridge of the virtual node. The time to setup the virtual links took 22.7 ms on average on tb1 and 23.0 ms on tb4 (see lowest green line), with

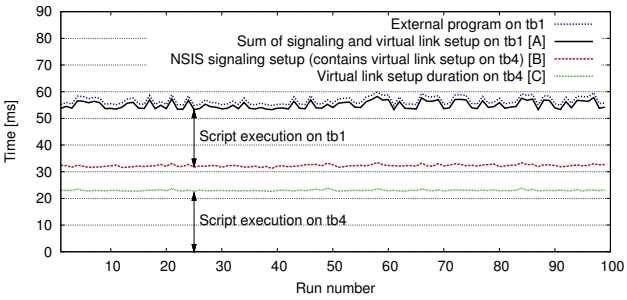

Figure 9. Signaling duration to setup a reservation and establish the corresponding virtual link between tb1 and tb4

a standard deviation of 1.3 ms and 0.23 ms respectively. Note that the signaling time [B] between the initial GIST QUERY and the corresponding QoS NSLP RESPONSE, as shown by the blue line, already contains the time required to setup the virtual link on tb4. Once the QoS NSLP RESPONSE reaches tb1, it reserves the resources and sets up the virtual link accordingly. Therefore, we can determine, that the difference between the blue and the green line accounts for the plain signaling overhead. The second line from the top is the sum of the time required for the NSIS signaling and the time required to setup the virtual link on tb1. The difference between this black line and the top red line accounts for the communication overhead induced by the UNIX Domain Socket interface between the NSIS instance and the external program.

Figure 10 shows the measurement results in case reservations and virtual links are torn down. This is again initiated from an external program on tb1. The entire time from initiating this request until the final confirmation is sent to the external program took 348.5 ms on average with a standard deviation of 57.7 ms. This relatively high number—compared to the previously discussed setup request—stems from the rather high costs that are associated with the removal of a virtual link, i.e. the EGRE tunnel in our case. It took 189.8 ms on average to call a script on tb1 that detaches endpoints from a tunnel and then removes the tunnel from the system. Freeing tunnel resources, however, is more costly than setting them up due to additional checks on resource usage and clean up of structures.

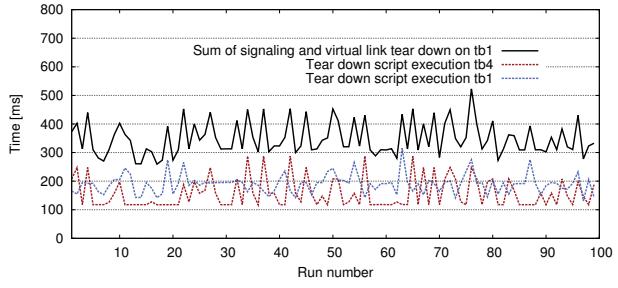

Figure 10. Signaling duration to tear down a reservation and to remove the corresponding virtual link between tb1 and tb4

The virtual link on the initiator's side (tb1) was not removed at the beginning of the tear down request, but

rather once the corresponding RESPONSE message from tb4 has been received. The signaling time is measured from the emitting tearing RESERVE on tb1, until the corresponding RESPONSE reaches tb1 and already contains the removal of the virtual link on tb4. In this case, the plain signaling overhead is negligible compared to the costs that are associated with the removal of an EGRE tunnel. The same applies for the overhead of the communication with the external program via the UNIX Domain Socket interface. Compared to the sum of the signaling overhead and the time required to remove the virtual link on tb1, the difference between this sum and the total time measured on the external program is negligible.

Table I summarizes the evaluation results. In both cases, for the establishment and the removal of virtual links, we see that the overall time, seen from the external program, is composed of the total NSIS signaling duration plus the virtual link setup on tb1 plus an overhead for the interprocess communication. The NSIS signaling duration itself contains the time required to setup or remove a virtual link on the receiver's side, i.e. tb4.

Table I
EVALUATION RESULTS FOR THE ESTABLISHMENT AND REMOVAL OF VIRTUAL LINKS FOR 100 RUNS

Establishment of a virtual link with QoS guarantee [ms]		Avg	StdDev	95% Conf. Int.
External program	[A]	56.8	1.96	[56.40, 57.16]
Virtual link setup on tb1	[H]	22.7	1.30	[22.47, 23.00]
Virtual link setup on tb4	[C]	23.0	0.23	[22.96, 23.06]
Total NSIS signaling duration	[B]	32.4	1.10	[32.17, 32.60]

Removal of a virtual link and tear down of QoS reservation [ms]	Avg	StdDev	95% Conf. Int.
External program	348.5	57.7	[337.03, 359.94]
Virtual link tear down on tb1	189.8	33.4	[183.16, 196.43]
Virtual link tear down on tb4	152.5	51.9	[142.25, 162.83]
Total NSIS signaling duration	157.2	51.8	[146.92, 167.50]

The results obtained are perfectly in line with the detailed measurements for the plain signaling overhead. As shown in Figure 11 a GIST three-way handshake took 1.25 ms between tb3 and tb4 (cf. [D] in Figure 8) on average, processing and forwarding of a QoS NSLP RESERVE on tb3 (calculated by [E]-[D]) took 1.15 ms on average, and processing and forwarding a RESPONSE on tb3 took 0.75 ms on average.

The signaling between tb1 and tb4 consists of three GIST three-way handshakes (3 × 1.25 ms = 3.75 ms), three times processing a RESERVE (3 × 1.15 ms = 3.45 ms) and three times processing and forwarding of a RESPONSE (3 × 0.75 ms = 2.25 ms). Therefore, the plain signaling overhead between an initial GIST QUERY and the final QoS NSLP RESPONSE equals to approximately 3.75 ms + 3.45 ms + 2.25 ms = 9.45 ms which equals

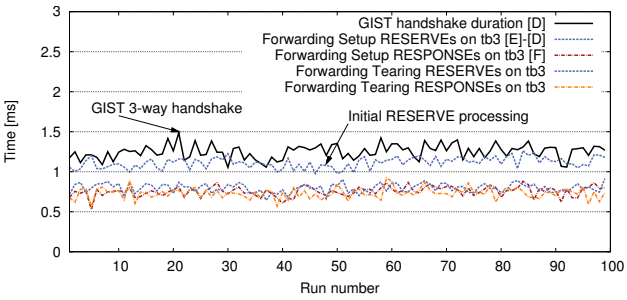

Figure 11. Plain signaling overhead on tb3

almost exactly the time calculated from the results gained from Table I with 32.4 ms – 23.0 ms = 9.4 ms.

The evaluation results clearly demonstrate that the necessary signaling to setup and tear down virtual links with Quality-of-Service guarantees can be performed very quickly. Especially with regard to the critical factor of a quick instantiation of virtual links upon a user's request, the provided NSIS-based solution performed quite well with overall performance costs of less than 60 ms in our setup. Furthermore, we showed that the NSIS signaling and QoS resource reservation only account for a marginal amount of the total time consumed to setup or remove a virtual link, since the main time is spent on local actions (script execution) to set up the virtual link. In case a virtual link is established, the plain NSIS signaling accounts for 9.45 ms on average which translates to about 17% of the total time required. In case a user requests a virtual link to be torn down, the plain NSIS signaling accounts for 4.7 ms on average which translates to only 1.2% of the total time required to tear down the virtual link.

The results can be extrapolated into a formula for the setup delay between nodes $n_0, \ldots, n_h$ as follows:

$$d_{\text{setup}} = 2t_s \; + \sum_{i=0}^{h-1} (g_i + q_i + r_i) \; + q_h + r_h$$

whereby t_s is the tunnel setup time on one system, h is the number of substrate links (or hops) between the tunnel ends n_0 and n_h, g_i is the duration of the GIST three-way handshake between nodes i and $i+1$, q_i, r_i are the durations for the QoS NSLP processing (including admission control) for a RESERVE and RESPONSE respectively. g_i depends on the 'distance' (e.g., round-trip time) between nodes i and $i+1$ while q_i and r_i depend on the local processing performance on each node i.

On the basis of our previously described measurements (and assumption of identical values per node) we estimate the setup delay in a ten node case (eight intermediate nodes and two end nodes) coarsely to

$$\begin{aligned} d_{\text{setup}} &= 2 \cdot 23 + \sum_{i=1}^{9} (1.25 + 1.15 + 0.75) + 1.19\,\text{ms} \\ &= 75.54\,\text{ms} \end{aligned}$$

The value of 1.19 ms for $q_h + r_h$ was calculated by subtracting the script execution time (cf. [C] in Figure 8) of 23 ms from the average time between the incoming RESERVE and the outgoing RESPONSE of 24.19 ms measured by tcpdump.

So one can expect that setting up a virtual link even across a larger distance is usually well below 100 ms and still dominated by the time for execution of the setup scripts.

The session authorization object from [13] was also implemented and secured the signaling message exchange by applying an HMAC-based signature. The signature key for the HMAC was pre-shared and installed on tb1 and tb4 prior to the signaling message exchange. Figure 12 and Table II show the measurement results of 100 consecutive runs. An HMAC signature generation takes 29 μs on average at tb1, the corresponding HMAC verification takes 33 μs at tb4. The HMAC verification is somewhat slower, because some basic parsing must be performed in order to collect all necessary objects that are integrity protected. Strong variations as present in Figure 12 in may stem from system interrupts.

In our implementation the HMAC verification takes place as early as possible, i.e., directly after receipt of a GIST PDU. The latter is not fully parsed, only object headers and types are analyzed to determine whether an HMAC verification is necessary. In case an HMAC verification must be performed, i.e., a Session Authorization Object of type `HMAC_Signed` is present containing a corresponding authorizing entity, the PDU parsing is continued only after a successful HMAC verification, otherwise the PDU gets discarded.

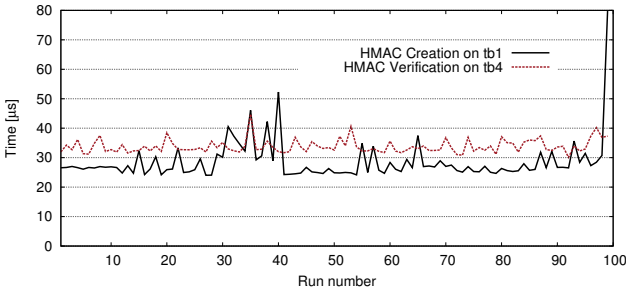

Figure 12. HMAC Creation and Verification

The session authorization object comprised 104 bytes of data and included also the VLSP object contents in its message authentication digest. These results show that integrity protection of virtual link setup signaling is possible with small computational overhead.

Table II
EVALUATION RESULTS FOR HMAC GENERATION AND VERIFICATION FOR 100 RUNS

Duration of HMAC-based integrity operations [μs]	Avg	Min	Max	StdDev
HMAC creation	28.672	24.026	79.77	7.011
HMAC verification	33.903	30.095	44.746	2.465

Figure 13 shows that the overhead implied by the additional Session Authorization Object and the integrity check is negligible. The overhead includes generation of the Session Authorization Object, creation of the integrity checksum (HMAC) at tb1, and its verification at tb4. The intermediate systems have to parse the Session Authorization Object, but ignore it since they are intermediate nodes. All RESERVE as well as all RESPONSE messages are protected by an additional Session Authorization Object. The shared key between tb1 and tb2 for HMAC calculation was pre-installed.

The upper two curves are based on the internal measurement points and are calculated by subtracting the durations for script execution [C] and [H] (cf. Figure 8) from the overall duration [A]. Since they are overlapping most of the time, the overhead for authenticated signaling is negligible for the overall signaling process. The lower two curves are based on the packet captures and are calculated by subtracting the script execution at tb4 [C] from the duration [B]. They exclude all the local processing at tb1 that precedes the first GIST QUERY and that follows the reception of the RESPONSE. Since they are also overlapping, they confirm that the overhead for authenticated signaling is negligible. The shape of the lower curves fits perfectly with the corresponding upper curves, i.e., there is only a constant offset between them, stemming from additional internal processing that was included in the internal measurement points, but not in the packet captures.

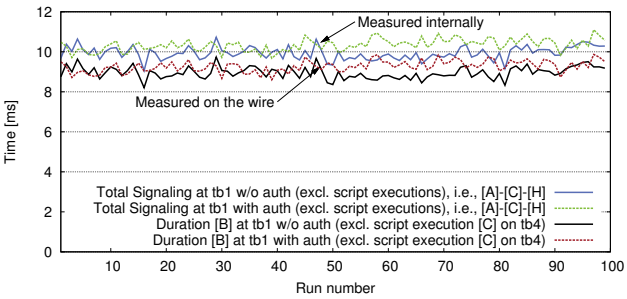

Figure 13. Signaling with and without authentication in comparison

As a conclusion we note that the time required for signaling is very small compared to the time for actually setting up a tunnel end for the virtual link. Furthermore, the signaling can be secured without noticeable overhead.

V. RELATED WORK

One of the most popular network virtualization research systems is the X-Bone[16], which deploys and manages IP-based virtual networks. X-Bone overlay networks use two-folded IP-in-IP tunneling, provide a user-interface for the configuration of overlay networks, and support the use of recursive overlays. X-Bone uses scripting to manage its networks. In this work we focus on the signaling for plain virtual links on top of an IP-based substrate. We therefore do not require any particular overlay addressing scheme that must be used within the virtual network. Instead, the

virtual links can be used as-is which allows for general purpose use of such virtual links, e.g., running IP or other protocols inside the virtual network. Comparable to the X-Bone approach, security can be achieved either for the overlay link itself via IPsec or through another appropriate form of security protection inside the virtual link.

In earlier work, Lim et al. introduced a Virtual Network Service (VNS) architecture that can be used to deploy customizable virtual private networks with QoS guarantees [17]. A dedicated signaling protocol called Beagle is used for resource allocation for virtual links. This QoS support is, however, only enforced on virtual routers and does not perform flow-based signaling along the underlying substrate's data path.

Integrated Quality-of-Service support for virtual networks was also part of several overlay systems, such as Darwin [18] which also uses Beagle as its resource allocation protocol and includes a VNS component called supranet [19] for dynamic overlay deployment. This component requires, however, OS modifications in order to use custom tunneling and Quality-of-Service support. As in the X-Bone approach we aim at avoiding any operating system or application modifications that are necessary in order to use our virtual links.

Bandwidth sharing between virtual links is of particular importance for virtual networks. In order to overcome inefficient static division of resources for virtual networks, the DaVinci architecture [20] uses optimization theory to efficiently share underlying network resources. In our approach we do not aim at an optimum resource sharing between virtual links, but want to provide guaranteed Quality-of-Service resource reservations for virtual links, for which we install and maintain state on network nodes by means of IP-based signaling protocols.

In [21] Feamster et al. propose a high-level design of an architecture for concurrent virtual networks by separating infrastructure from service providers. The architecture supports real end-to-end services. According to the authors, signaling protocols should then be used for the coordination between service and infrastructure providers.

A very promising approach towards a network virtualization platform is Trellis [22], [23]. Virtual links are realized via a Ethernet GRE (EGRE) tunneling mechanism, which allows for direct layer-two link connectivity between any two virtual nodes on top of an existing IP-substrate. The implementation is flexible enough to allow virtual hosts to control their own forwarding tables and still provide isolation of different virtual links by terminating virtual links in the root context rather than in the virtual host containers. We partially follow this approach and do also provide layer-two connectivity between virtual nodes. However, Trellis does not consider signaling mechanisms to control elements in the substrate, e.g., for QoS guarantees or an on-demand setup of virtual links.

Schaffrath et al. [5] recently presented a proposal and initial prototype of a network virtualization architecture. Different from research initiatives like GENI, which focused on the provisioning of an experimental facility, this proposal identifies the different entities and roles that are necessary for a network virtualization architecture. In this proposed architecture, the responsibilities to manage a virtual network are separated between virtual network operators, virtual network providers, and infrastructure providers. Virtual link setup was not explicitly considered or described in detail. The approach, however, fits well to the proposal of our virtual link setup protocol that is triggered by the infrastructure providers on behalf of the requests coming from the virtual network providers.

VI. Conclusion

The herein proposed virtual link setup scheme couples a Quality-of-Service resource reservation in the substrate with the setup of a virtual link between virtual nodes. While resource reservation and its related signaling basically take only a small amount of time, setup and tear down of virtual links take more time due to shell script execution. Instead of using a resource reservation protocol and a separate tunnel setup protocol we combined both protocols leading to a more efficient solution, which is additionally secured by using the session authorization object. Furthermore, since the QoS signaling follows the substrate path it is assured that the virtual link is actually working after setup.

As the Quality-of-Service guarantees are currently bound to the outer tunnel endpoints, IPv6 could provide an even better isolation between different virtual links in terms of QoS, as each virtual node may be easily equipped with a dedicated IPv6 address of the substrate node's subnet. This would allow for an easier flow classification. However, this is future work since the current Linux kernels do unfortunately not yet provide support for IPv6 as substrate protocol for GRE and L2TP. We are currently extending our approach by using NSIS also for virtual node setup thus working towards a more integrated solution.

VII. Acknowledgments

Part of this work was conducted within the 4WARD project, which was funded by the European Union in the 7th Framework Programme (FP7), as well as within the G-Lab project, funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (support code 01 BK 0809, G-Lab, http://www.german-lab.de/). Moreover, we acknowledge support by Deutsche Forschungsgemeinschaft and Open Access Publishing Fund of Karlsruhe Institute of Technology.

References

[1] R. Bless, M. Röhricht, and C. Werle, "Authenticated Setup of Virtual Links with Quality-of-Service Guarantees," in *Proceedings of 20th IEEE International Conference on Computer Communications and Networks (ICCCN 2011)*. IEEE, Aug. 2011, pp. 1–8.

[2] GENI, "Global Environment for Network Innovations," Feb. 2011. [Online]. Available: http://www.geni.net

[3] FIND, "Future Internet Design," Feb. 2011. [Online]. Available: http://www.nets-find.net

[4] AKARI, "Architecture Design Project for New Generation Network," Feb. 2011. [Online]. Available: http://akari-project.nict.go.jp/

[5] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy, "Network Virtualization Architecture: Proposal and Initial Prototype," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, ser. VISA '09. New York, NY, USA: ACM, 2009, pp. 63–72. [Online]. Available: http://doi.acm.org/10.1145/1592648.1592659

[6] W. Simpson, "IP in IP Tunneling," RFC 1853 (Informational), Internet Engineering Task Force, Oct. 1995. [Online]. Available: http://www.ietf.org/rfc/rfc1853.txt

[7] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)," RFC 2784 (Proposed Standard), Internet Engineering Task Force, Mar. 2000, updated by RFC 2890. [Online]. Available: http://www.ietf.org/rfc/rfc2784.txt

[8] J. Lau, M. Townsley, and I. Goyret, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)," RFC 3931 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFC 5641. [Online]. Available: http://www.ietf.org/rfc/rfc3931.txt

[9] X. Fu, H. Schulzrinne, A. Bader, D. Hogrefe, C. Kappler, G. Karagiannis, H. Tschofenig, and S. V. den Bosch, "NSIS: A New Extensible IP Signaling Protocol Suite," *Communications Magazine, IEEE*, vol. 43, no. 10, pp. 133–141, October 2005.

[10] J. Manner, G. Karagiannis, and A. McDonald, "NSIS Signaling Layer Protocol (NSLP) for Quality-of-Service Signaling," RFC 5974 (Experimental), Internet Engineering Task Force, Oct. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5974.txt

[11] H. Schulzrinne and R. Hancock, "GIST: General Internet Signalling Transport," RFC 5971 (Experimental), Internet Engineering Task Force, Oct. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5971.txt

[12] G. Ash, A. Bader, C. Kappler, and D. Oran, "QSPEC Template for the Quality-of-Service NSIS Signaling Layer Protocol (NSLP)," RFC 5975 (Experimental), Internet Engineering Task Force, Oct. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5975.txt

[13] J. Manner, M. Stiemerling, H. Tschofenig, and R. Bless, "Authorization for NSIS Signaling Layer Protocols," RFC 5981 (Experimental), Internet Engineering Task Force, Feb. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc5981.txt

[14] R. Bless and M. Röhricht, "Secure Signaling in Next Generation Networks with NSIS," in *Communications, 2009. ICC '09. IEEE International Conference on*. Dresden, Germany: IEEE, Jun. 2009, pp. 1–6.

[15] Institute of Telematics, "NSIS-ka – A free C++ implementation of NSIS protocols," Aug. 2011. [Online]. Available: http://nsis-ka.org/

[16] J. Touch, "Dynamic Internet overlay deployment and management using the X-Bone," *Computer Networks*, vol. 36, no. 2-3, pp. 117–135, Jul. 2001.

[17] L. K. Lim, J. Gao, T. S. E. Ng, P. R. Chandra, P. Steenkiste, and H. Zhang, "Customizable Virtual Private Network Service with QoS," *Computer Networks*, vol. 36, no. 2-3, pp. 137–151, May 2001.

[18] P. Chandra, Y. hua Chu, A. Fisher, J. Gao, C. Kosak, T. S. E. Ng, P. Steenkiste, E. Takahashi, and H. Zhang, "Darwin: Customizable Resource Management for Value-Added Network Services," *Network, IEEE*, vol. 15, no. 1, pp. 22–35, Jan. 2001.

[19] L. Delgrossi and D. Ferrari, "A Virtual Network Service for Integrated-Services Internetwork," in *Proceedings of the 7th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 97)*, St. Louis, MO, USA, May 1997, pp. 291–295.

[20] J. He, R. Zhang-Shen, Y. Li, C.-Y. Lee, J. Rexford, and M. Chiang, "DaVinci: Dynamically Adaptive Virtual Networks for a Customized Internet," in *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*. New York, NY, USA: ACM, Dec. 2008, pp. 1–12.

[21] N. Feamster, L. Gao, and J. Rexford, "How to Lease the Internet in your Spare Time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61–64, Jan. 2007.

[22] S. Bhatia, M. Motiwala, W. Mühlbauer, Y. Mundada, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford, "Trellis: A Platform for Building Flexible, Fast Virtual Networks on Commodity Hardware," in *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*. New York, NY, USA: ACM, Dec. 2008, pp. 1–6.

[23] S. Bhatia, M. Motiwala, W. Mühlbauer, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford, "Hosting Virtual Networks on Commodity Hardware," Department of Computer Science, Georgia Tech, Atlanta, GA, USA, Tech. Rep. GT-CS-07-10, Jan. 2008.

AUTHORS' BIOGRAPHIES

Roland Bless is associate professor and senior researcher at the Institute of Telematics, at the Karlsruhe Institute of Technology. He studied Computer Science (Diplom-Informatik) at the University of Karlsruhe until 1996 and got his PhD (Dr.-Ing.) in the area Quality-of-Service-Management in 2002. His main research interests are in network virtualization, future Internet architectures and protocols, peer-to-peer and overlay networks as well as IPv6. Since 1998 he is active in the Internet standardization within the IETF and gives lectures at KIT on 'Next Generation Internet' and 'Multimedia Communications'. Dr. Bless is member of GI, ACM SIGCOMM, and IEEE ComSoc.

Martin Röhricht graduated in Computer Science (Diplom-Informatik) at the University of Karlsruhe in 2007 and is currently a PhD student at the Institute of Telematics, at the Karlsruhe Institute of Technology (KIT) under the supervision of Prof. Dr. Martina Zitterbart. His main research interests are Quality-of-Service in heterogeneous environments, Internet signaling protocols, and network virtualization. Martin Röhricht is member of GI, ACM SIGCOMM, and IEEE ComSoc.

Christoph Werle graduated in Computer Science (Diplom-Informatik) at University of Karlsruhe in 2008. Under supervision of Prof. Dr. Martina Zitterbart, he is currently working towards a PhD with the Institute of Telematics at the Karlsruhe Institute of Technology. Besides Internet signaling protocols, his research interests include routing protocols, network virtualization, and security in communication systems.