Authenticated Setup of Virtual Links with Quality-of-Service Guarantees

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 control protocols that mostly neglect security considerations. In this work, we present our proposal for a Virtual Link Setup Protocol (VLSP) that is designed as a modular extension to a standardized and extensible 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.**

I. Introduction

Network virtualization is a promising abstraction technique that allows for optimizing the utilization of network resources 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 have examined the subject closely, such as the Global Environment for Network Innovations (GENI) [1], the NSF NeTS FIND Initiative [2], or the AKARI Architecture Design Project [3]. Within these research projects, various prototypes for network virtualization architectures have been constructed and evaluated. Many approaches are mainly concentrating on providing an experimental facility for network researchers. Contrastingly, the 4WARD project [4] of the EU 7th Framework Programme focused on a more general network virtualization architecture that also considers different business roles and various types of providers. One important aspect was to consider the creation of virtual links with quality-of-service (QoS) guarantees *across* different substrate Infrastructure Provider (InP) *domains*.

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 sender authentication of the involved signaling messages. This permits to verify whether the sender 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 and 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. Section IV evaluates the signaling performance of the proposed Virtual Link Setup Protocol. 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 components, *virtual nodes* and *virtual links*. Virtual nodes appear and act like physical nodes, i.e., they have access to a specific set of resources, run a (network) operating system and other software. But instead of running directly on dedicated physical hardware, they only run in a virtual environment, where a virtualization layer provides access to a set of virtual resources. 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 have an impact on the system's performance (e.g., the speed to forward packets) and the isolation between virtual nodes and virtual networks.

There exists a multitude of different host virtualization technologies that can be used for realizing virtual nodes based on commodity PC hardware. The technologies can be logically separated 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 being used. Moreover, one can imagine dedicated router hardware with 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 [5], Generic Routing Encapsulation (GRE) [6], Layer 2 Tunneling Protocol (L2TP) tunnels [7], or others.

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 influencing 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. Therefore, the provisioning of *Quality-of-Service guarantees* must provide an integral component of any comprehensive network virtualization framework. Since the elasticity of virtual resources is a major advantage of virtual networks, they must also support quick instantiation. Thus manual configuration at scale is not feasible but requires a robust and flexible signaling protocol that allows for automated *on-demand setup of virtual links*. Furthermore, it must be ensured that the setup of virtual links and the reservation of corresponding resources in the substrate is properly *authenticated* in order to protect the setup of virtual links from forged or tampered signaling messages.

Figure 1 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.

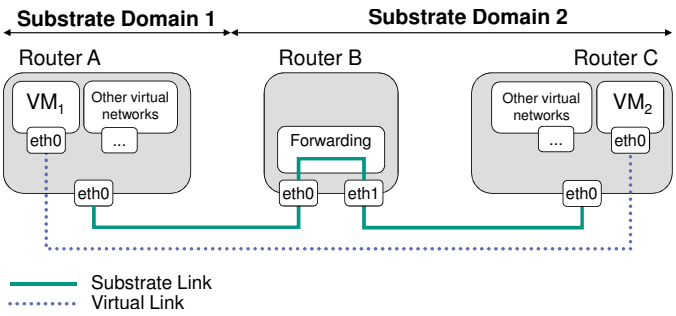

Figure 1. Basic network virtualization example

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 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.

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 Signaling* (NSIS) framework [8], which provides an up-to-date IP-based signaling protocol suite and which can be used for a variety of signaling applications. Even though QoS signaling can already

be accomplished by means of the QoS NSLP protocol [9], 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 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.

Figure 2 gives a conceptual overview of the NSIS protocol architecture and incorporates the VLSP object as an extension of the QoS NSLP.

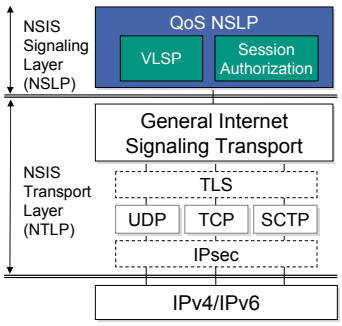

Figure 2. Conceptual overview of the NSIS protocol architecture with the VLSP object and the Session Authorization object

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) [10] 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 [9] is a soft-state protocol that reserves resources along a data path and installs state in nodes on this path accordingly. It conveys a dedicated QoS Specification (QSPEC) [11] object for specifying QoS parameters. Therefore, it abstracts from the actually used QoS mechanisms on the data path like IntServ or DiffServ. 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 an optional so-called *Session Authorization Object* [12] which provides an authorization token that can be used to authenticate NSIS signaling messages on a per user or per session basis [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 3 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.

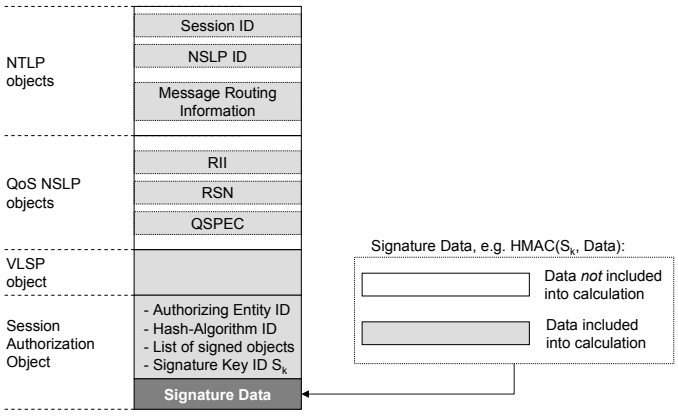

Figure 3. Binding of Session Authorization Object and NSIS message objects

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)
- (De-)Mux method
- Opaque descriptor of the (De-)Mux method

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 4. 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 is only dependent on the used tunneling mechanism.

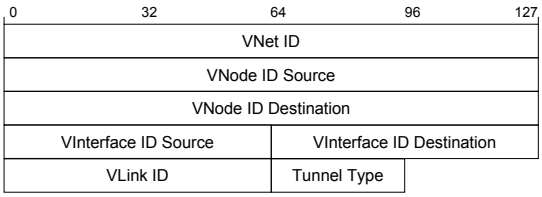

Figure 4. Conceptual overview of the VLSP object

The addressing information carried in the VLSP object enables the endpoints of the virtual link to connect the 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. Thus the VLSP object can be introduced in a backwards compatible fashion by using the NSLP object extensibility flags. These flags instruct intermediate NSIS entities that are not aware of this newly introduced object to leave this object unchanged when forwarding the message.

Figure 5 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 checks the authenticity and integrity of the message and 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.

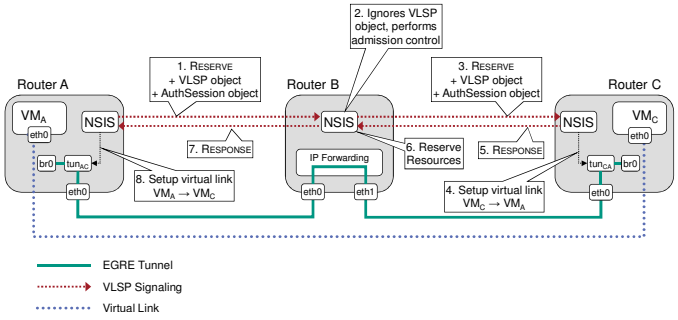

Figure 5. Exemplified scenario of a virtual link setup procedure

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. After that Router C sends a QoS NSLP RESPONSE back to Router B, 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.

Since virtual links are usually used for bidirectional communications, the QoS reservation should also be established in both directions. Therefore, the QoS NSLP receiver, 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 5.

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 of a tunnel for the virtual link.

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

We evaluated the proposed virtual link setup protocol in a testbed environment following the setup depicted in Figure 6. Each node consists of Intel Xeon X3430 quad-core CPUs running at 2.40GHz, 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. 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. Fine-grained measurements were performed by putting reference 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.

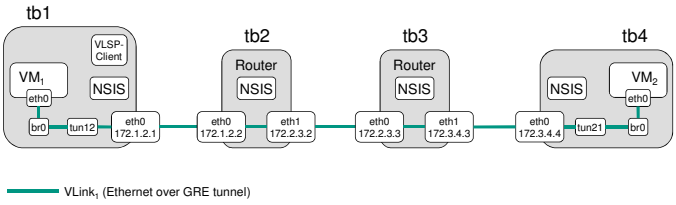

Figure 6. Evaluation setup with four Linux routers and two different virtual machines being connected through an EGRE tunnel

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 [14] with disabled logging output. 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. Potential routers on the path that do not support NSIS would simply act as plain IP routers and therefore not being involved in the QoS signaling.

B. Signaling Performance

We performed 100 separate runs to evaluate the signaling performance of our virtual link setup protocol. The experiments run as follows. 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 peers and corresponding QoS NSLP RESERVE messages carrying the VLSP object. Once the RESERVE reaches tb4, it establishes its virtual link endpoint by calling a shell script. After successful creation of that tunnel, tb4 sends a RESPONSE back towards tb1. Once the NSIS instance on tb1 receives the RESPONSE it also establishes its virtual link's endpoint upon which it finally informs the external program about the success of the operation.

Figure 7 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 37.8 ms on average with a standard deviation of only 0.97 ms.

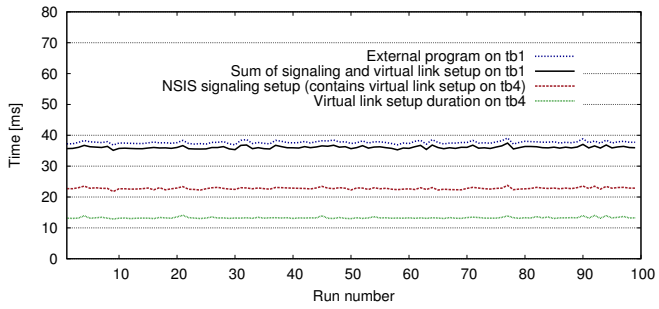

Figure 7. Signaling duration to setup a reservation and establish the corresponding virtual link between tb1 and tb4

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 13.3 ms on average on both, tb1 and tb4 (see lowest green line), with a standard deviation of 0.2 ms. Note, that the signaling time 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 8 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 355.3 ms on average with a standard deviation of 43.1 ms. This relatively high number—compared to the previously discussed setup request—comes 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 180.4 ms on average to call a script 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.

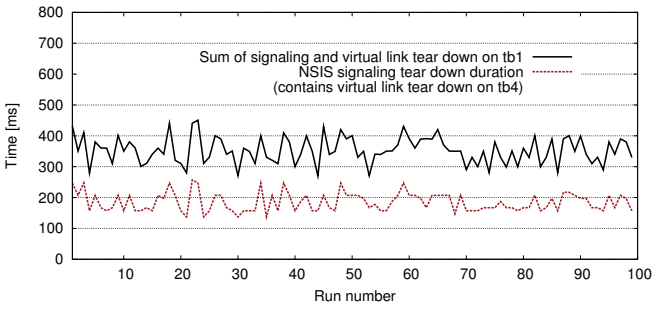

Figure 8. Signaling duration to tear down a reservation and remove the corresponding virtual link between tb1 and tb4

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 inter-process 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.

The results obtained are perfectly in line with the detailed measurements for the plain signaling overhead. As shown in Figure 9 a GIST three-way handshake took 1.37 ms between tb1 and tb2 on average, processing and forwarding of a QoS NSLP RESERVE on tb2 took 1.09 ms on average, and processing and forwarding a RESPONSE on tb2 took 0.75 ms on average.

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	37.8	0.98	[37.62, 38.01]
Virtual link setup on tb1	13.3	0.27	[13.23, 13.33]
Virtual link setup on tb4	13.3	0.23	[13.23, 13.32]
Total NSIS signaling duration	22.8	0.40	[22.74, 22.91]

Removal of a virtual link and tear down of QoS reservation [ms]	Avg	StdDev	95% Conf. Int.
External program	355.3	43.2	[346.75, 363.89]
Virtual link tear down on tb1	168.2	26.9	[162.25, 173.59]
Virtual link tear down on tb4	180.4	30.2	[174.39, 186.38]
Total NSIS signaling duration	185.4	30.2	[179.43, 191.40]

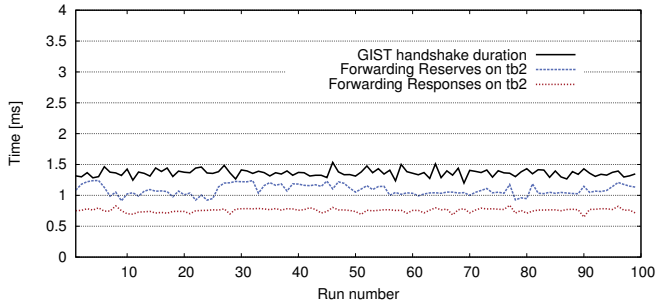

Figure 9. Plain signaling overhead on tb2

The signaling between tb1 and tb4 consists of three times GIST's three-way handshake (3×1.37 ms = 4.11 ms), three times processing a RESERVE (3×1.09 ms = 3.27 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 4.11 ms + 3.27 ms + 2.25 ms = 9.63 ms which equals almost exactly the time calculated from the results gained from Table I with 22.8 ms – 13.3 ms = 9.5 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 performance costs of less than 40 ms in our setup. Furthermore, we showed that the NSIS signaling itself only accounts for a marginal amount of the total time consumed to setup or remove a virtual link. In case a virtual link is established, the plain NSIS signaling accounts for 9.5 ms on average which translates to about 25% of the total time required. In case a user requests a virtual link to be torn down, the plain NSIS signaling accounts for 5 ms on average which translates to only 1.4% of the total time required to tear down the virtual link.

The session authorization object from [12] 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. Table II shows the measurement results of 100 consecutive runs. An HMAC signature generation takes 31 μs on average at tb1, the corresponding HMAC verification takes 40 μs at tb4. 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	31.478	29.615	46.211	2.633
HMAC verification	39.731	36.096	64.754	3.222

V. RELATED WORK

One of the most popular network virtualization research systems is the X-Bone [15] which deploys and manages virtual networks. The 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. The 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 don't promote any particular overlay addressing scheme that can be used within this virtual network. Instead, the virtual links can be used as-is which allows for general purpose use of such virtual links. 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 [16]. 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 [17] which also uses Beagle as its resource allocation protocol and includes a VNS component called supranet [18] for dynamic overlay deployment. This component requires, however, OS modifications in order to use custom tunneling and Quality-of-Service support. Very much like in the X-Bone we aim at avoiding any operating system or application modifications that are necessary in order to use our virtual links.

Bandwidth shares between virtual links are of particular importance for virtual networks. In order to overcome inefficient static division of resources for virtual networks, the DaVinci architecture [19] uses optimization theory to efficiently share underlying network resources. In our approach we don't 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 [20] Feamster et al. propose a high-level design of an architecture for concurrent virtual networks by separating infrastructure from service providers. The architecture allows for the support of 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 builds Trellis [21], [22]. Virtual links are realized via a Ethernet GRE (EGRE) tunneling mechanism which allows for the appearance of 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. [4] 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 the resource reservation basically takes only a small amount of time, the setup and tear down of virtual links take longer 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 can be additionally secured by using the session authorization object.

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 the substrate protocol for GRE and L2TP. We are currently extending our approach by using NSIS also for the node setup thus working towards a more integrated solution.

VII. Acknowledgments

Part of this work was performed within the 4WARD project, which is funded by the European Union in the 7th Framework Programme (FP7), and 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/).

References

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

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

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

[4] 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

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

[6] 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

[7] 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

[8] 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.

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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.

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

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.