

The Underlay Abstraction in the Spontaneous Virtual Networks (SpoVNet) Architecture

Roland Bless, Christian Hübsch, Sebastian Mies, and Oliver P. Waldhorst
Institute of Telematics – Universität Karlsruhe (TH) – 76128 Karlsruhe, Germany
{bless,huebsch,mies,waldhorst}@tm.uka.de

Abstract—Overlay-based services are a popular approach for providing functions like multicast, quality of service or security in the Internet without requiring infrastructure support. This paper presents the *Underlay Abstraction Layer* in the *Spontaneous Virtual Networks (SpoVNet)* architecture that enables easy and flexible creation of such services. Also building on an overlay approach, the Underlay Abstraction provides generic functionality to cope with mobility, multi-homing, and heterogeneity. It manages node mobility by separating node identifiers from network locators and it provides persistent connections by transparently switching locators. Multi-homing is supported by choosing the most appropriate pair of network locators for each connection. In order to cope with network and protocol heterogeneity, it uses dedicated overlay nodes, e.g., for relaying between IPv4 and IPv6 hosts. Since the functionality provided by the Underlay Abstraction can be used by several overlay-based services in parallel, redundant functionality is removed from services and applications.

I. INTRODUCTION

Today's network structure for global communications is characterized by a multitude of heterogeneous access technologies, different protocols, user mobility, and systems possibly connected by multiple access links. Missing in today's Internet is, however, integrated support for multicast, quality-of-service (QoS), and security. In the past, several solutions were developed for these functions, but they are not widely available, because these functions are only partially supported, e.g., QoS in UMTS- or IEEE 802.11e-based access networks or multicast in the core networks of certain Internet Service Providers (ISPs). Since applications cannot rely on the required functionality everywhere, such features are not widely used. Currently, it is not obvious whether next generation networks will offer native support for such functions.

One approach for making missing functionality available without infrastructure support is to deploy *overlay services*. Such services are located in the application layer and consist of several instances of a distributed application running on different end systems. The instances are connected by transport links, making up a logical *overlay network* on top of a physical network infrastructure. Well-known examples for such overlay services constitute the *Narada* protocol [1] for end system

multicast, the overlay-based *OverQoS* architecture [2] for enhancing the best-effort quality of service of today's Internet, and the peer-to-peer VPN solution *Hamachi* [3]. But such overlay-based services have also several drawbacks. First, each overlay service brings its own mechanisms for dealing with underlay characteristics like network dynamics and link failures, or for adapting communication to current traffic conditions. In the best case, mechanisms are replicated in several services, adding redundancy. However, their employment may also be inefficient when different services try to achieve contrary goals when allocating network resources. Second, most overlay services build on a single, homogeneous network protocol (e.g., IPv4) and cannot be easily employed in future networks (e.g., based on IPv6) or even in heterogeneous networks (e.g., combining IPv4 and IPv6). Third, most current approaches are not able to transparently employ native upcoming underlay mechanisms, e.g., QoS or native IP multicast support.

In this paper, we present the *Underlay Abstraction layer* of the Spontaneous Virtual Networks (SpoVNet) Architecture. SpoVNETs enable flexible, adaptive, and spontaneous provisioning of application-oriented and network-oriented services on top of heterogeneous networks. The SpoVNET Underlay Abstraction supports creation of overlay services by providing generic mechanisms for dealing with underlay characteristics and by providing an identifier based addressing scheme. The functionality provided by the underlay abstraction can be used by several overlay services at the same time, thus removing redundancy and more efficient use of network resources. The Underlay Abstraction transparently uses native underlay mechanisms like QoS-support where available. All functionality can be accessed by overlay services and applications using a generic interface.

The Underlay Abstraction comprises two components. As lower component in the SpoVNET stack, the *Base Communication* provides means for connection-less and connection-oriented communication between endpoints identified by sets of network locators. It transparently handles protocol heterogeneity as well as NAT and firewall traversal, maintains locator sets in spite of mobility, and selects appropriate locators for multi-homed nodes. Building upon the Base Communication, the *Base Overlay* provides node identifiers for addressing, implementing an ID/Locator split. The Base Overlay can be

This work is partially supported by the Landesstiftung Baden-Württemberg within the BW-FIT program. We want to thank all SpoVNET project members for useful discussions and remarks.

used to send control and data messages to nodes via key-based routing, but also to establish transport connectivity based on node identifiers.

This paper is organized as follows. An overview of the main concepts and components making up the SpoVNet architecture is provided in Section II. Section III comprehensively describes the Underlay Abstraction layer, with a focus on Base Communication and Base Overlay in Sections III-B and III-C, respectively. Section IV relates the presented concepts to previous work on handling heterogeneity in next generation networks. Finally, concluding remarks are given.

II. THE SPOVNET ARCHITECTURE

A. Project Goals

SpoVNet follows two major objectives. The first is the flexible, adaptive, and spontaneous provisioning of communication services in heterogeneous networks. The second is to enable a seamless transition from today's to future networks. This section discusses how these objectives are achieved by the SpoVNet architecture.

In order to accomplish objective one, SpoVNet supports the flexible provisioning of communication services by using overlays. Per definition, overlays can be established spontaneously and flexibly, and do not require special support from the network or a given infrastructure. Furthermore, properly designed overlays are self-organizing, scalable and robust. Section II-B discusses which components in the SpoVNet architecture enable easy service provisioning by overlays.

To reach objective two, SpoVNet accommodates the fact that some of the services, which SpoVNet provides in the application layer, will become an integral part of the future network infrastructure. Therefore, the architecture allows a seamless replacement of SpoVNet services with the corresponding network services once they become available. Section II-C discusses this in more detail.

B. Components of the SpoVNet Architecture

A *SpoVNet* (or, to be more precise, a *SpoVNet Instance*) provides communication services for participants that want to cooperate in a common application context. Members of the SpoVNet are logical entities, called *SpoVNet Nodes*, that are interconnected by the means of the SpoVNet software. The latter is running on a *SpoVNet Device* and provides the actual communication services that include at least a basic connectivity so that every SpoVNet Node can reach any other SpoVNet Node within the same SpoVNet. A *SpoVNet Device* is an arbitrary computing device, e.g., a PC or notebook, on which the SpoVNet software is installed and it has at least one network connection. In case of *multi-homing*, i.e., devices with more than one connection, each may use a different network protocol, e.g., one may use IPv4 while another uses IPv6. Devices may listen on each of the network connections for connection attempts by other SpoVNet Devices. The network address and the port a device listens on provide a *locator* for the device within SpoVNet. In general, each device has a non-empty *Locator Set*, i.e., it has one or more locators.

Applications can use the SpoVNet Software in order to create an own SpoVNet Instance or to join an existing instance as a SpoVNet Node. A node that creates a SpoVNet Instance is its *Initiator*. To allow other nodes to join, the Initiator selects a unique name called *Canonical SpoVNet Name* for a SpoVNet instance. For internal use, the string is mapped by a hash function to a bit string of fixed length denoted as *SpoVNet Unique Universal Identifier*. In the following we formalize our definitions to avoid ambiguities:

Definition 1: A set S of SpoVNet Nodes cooperating in a common application context is denoted as *SpoVNet Instance*. For an instance S , $CSN(S)$ denotes the *Canonical SpoVNet Name*, $UUID(S) = chic_S(CSN(S))$ is the *SpoVNet Unique Universal Identifier* and $START(S) \in S$ is the *Initiator* of S .

Each SpoVNet Instance S uses its own internal addressing scheme for the SpoVNet Nodes it contains. Within a SpoVNet Instance, each node has a unique *Canonical Node Name* that is mapped to a unique *Node Identifier*, which is mainly used as base for forwarding and routing. Thus, we have:

Definition 2: Let S be a SpoVNet Instance. Then a *SpoVNet Node* $a \in S$ has a *Canonical Node Name* $CNN_S(a)$ with $\forall a, b \in S, a \neq b : CNN_S(a) \neq CNN_S(b)$, and a *Node Identifier* $ID_S(a) = chic_S(CNN_S(a))$.

Definition 3: A $chic_S$ function is a cryptographic hash identifier construction function, as defined by the ORCHID scheme [4] and dependent on the SpoVNet Instance S . Using a public key pk as input, the ORCHID-defined prefix p , an extraction function e_S , a hash function h_S , and a global SpoVNet-specific context id c , the resulting value is computed as follows: $chic_S(pk) = p|e_S(h_S(c|pk))$ whereat the $|$ operator denotes the concatenation of two bitstrings.

Using this construction scheme, a node a can prove to a node b that it is owner of $ID_S(a)$ by using $CNN_S(a)$ as public key and input to $chic_S$. Furthermore a SpoVNet Instance $UUID(S)$ can authenticate itself, too, as detailed in section III-C3.

As mentioned earlier, SpoVNet uses overlays for flexible service provisioning. Establishing an overlay link between two arbitrary overlay nodes requires that at least one of the nodes can open a connection to the other. Unfortunately, connection establishment between the nodes may be impossible, either because both nodes are located behind firewalls or NAT gateways, or because the nodes use different network or transport layer protocols. In both cases, a common solution is to relay the connection via a different node, which can be reached from behind firewalls or NAT gateways or can relay between the different protocol families, respectively.

Reverting on this idea, we define two different forms of connectivity. *Direct underlay connectivity* means that a node can send packets to another node directly. In contrast, *indirect connectivity* is given if a node can send packets to another node using additional nodes as relays. Formally we can define:

Definition 4: For a SpoVNet Instance S , the *Direct Underlay Connectivity* $C_d : S \times S \rightarrow \{\text{true}, \text{false}\}$ between

two nodes $a, b \in S$ is given by $C_d(a, b) = \text{true} \Leftrightarrow (a \text{ can send packets to } b \text{ directly})$.

Definition 5: For a SpoVNet Instance S , the *Indirect Connectivity* $C : S \times S \rightarrow \{\text{true}, \text{false}\}$ is given by $C(a, b) = (\text{true}) \Leftrightarrow \exists x_1 = a, x_2, \dots, x_m = b \in S : \bigwedge_{i=1}^{m-1} C_d(x_i, x_{i+1})$. For $m > 2$, x_2, \dots, x_{m-1} are called *Relay Nodes*.

Direct underlay connectivity is a special case of indirect connectivity with $m = 2$. Indirect connectivity is *transitive*.

The *Base Communication* (cf. also Section III-B) provides *transport links* between nodes with direct or indirect connectivity. Overlays can be created in a straight forward way without caring for underlay connectivity by using transport links. Formally, we have:

Definition 6: For a SpoVNet Instance S , a *transport link* $T(a, b)$, $a, b \in S$, is a bidirectional transport connection from a to b , it can be created if $C(a, b) \wedge C(b, a)$. If $C_d(a, b) \wedge C_d(b, a)$, the link is called *direct* transport link. Otherwise, it is called *indirect* transport link.

A basic connectivity between all nodes in a SpoVNet instance is provided by the so-called *Base Overlay*, that is described in Section III-C. Formally defined, the Base Overlay consists of the graph connecting all SpoVNet nodes, where the edges are given by transport links:

Definition 7: For a SpoVNet Instance S , the *Base Overlay* $B_S = (S, E)$ is a connected¹, undirected graph that contains all nodes. For the set of edges holds $E \subseteq \{\{a, b\} | a, b \in S, C(a, b) \wedge C(b, a)\}$. For each edge $\{a, b\} \in E$ exists a transport link $T(a, b)$.

This definition provides indirect connectivity for the entire SpoVNet Instance and allows to establish transport links between arbitrary pairs of SpoVNet Nodes. With the Base Overlay it is possible to easily create new transport overlays for services. We identify a service t with its *Service ID* $SID(t)$ that is unique within a SpoVNet Instance.

To illustrate how the SpoVNet components defined in this section can be put into practice, Figure 1 shows a SpoVNet Instance with six SpoVNet Nodes in heterogeneous access networks. The nodes use different network protocols, i.e., IPv4 and IPv6. However, most of the logical links that form the ring-like Base Overlay use direct transport links. Only the connection between the desktop PC (top right) and the PDA (bottom right) involves different network protocols and must use an indirect transport link that is relayed by the notebook (bottom). Within the SpoVNet Instance, two service overlays with disjoint sets of nodes are created.

The following section describes how the individual components of the SpoVNet architecture are organized within a node.

C. Two-layer Abstraction Architecture

SpoVNet's second objective is to enable a seamless transition from current to future networks. To reach this objective,

¹It may actually be the case that B_S gets partitioned into two sets B_{S_1} or B_{S_2} due to link breaks in the underlay, but this is out of scope of this paper.

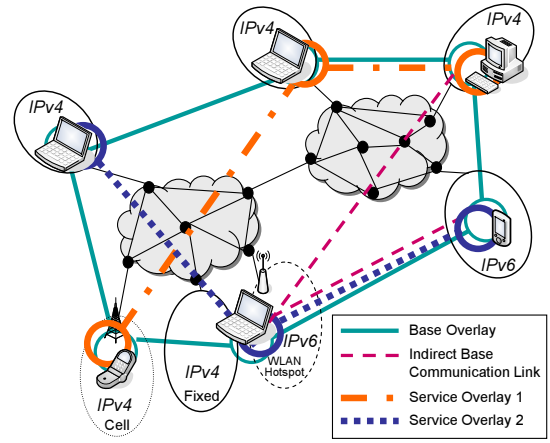


Fig. 1. A SpoVNet instance with two service overlays.

SpoVNet organizes the components defined in Section II-B in a layered architecture with three main layers as shown in Figure 2. The main layers are separated by abstraction layers, which are basically composed by the interfaces that are provided by each layer.

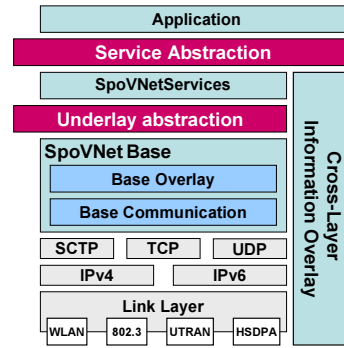


Fig. 2. The SpoVNet Architecture and the components of the SpoVNet Underlay Abstraction.

The topmost layer of the architecture constitutes the *Application layer*. SpoVNet does not define an architecture for applications, although a real-time online game and a video streaming application are developed for demonstration purposes within the project. SpoVNet ensures that applications do not require modification when the network evolves by providing stable interfaces within the *Service Abstraction*. The Service Abstraction ensures that the applications do not access network functionality directly, but by using advanced communication services. Thus, when new functionality is provided by the (evolving) network, it can be adopted by the service transparently for the application. An example constitutes an overlay-based application layer multicast service that might be partially replaced by Source-Specific IP Multicast in future networks. As long as the service interface is not modified, applications can already use advanced network services without requiring any change later.

The *SpoVNet Service Layer* consists of all SpoVNet Services that are implemented by service overlays as described

in Section II-B. Services in this layer are not strictly built one on top of another and are rather constructed side-by-side, so that one service needs not, but may use another service if required. For example, within the SpoVNet project, an event notification service is defined that uses the multicast/multipeer service.

The second abstraction layer is the *Underlay Abstraction*. It is composed from the interfaces provided by both the Base Overlay and the Base Communication. The former provides an Identifier-Locator split, while the latter provides the transport link abstraction. Thus, the Underlay Abstraction hides mobility, heterogeneity and multi-homing.

A direct consequence of employing abstraction layers is that both SpoVNet services and applications are “blind” for events within the network, which are purposely hidden by the abstractions. However, it might be beneficial if a service or application can autonomously adapt to changing underlay conditions by applying its own application-specific logic. For example, an application layer multicast service based on a tree-like overlay structure might want to remove a certain SpoVNet Node from a central tree position with high load if it roams from a high bandwidth fixed network connection (e.g., T1) to a low bandwidth cellular wireless connection (e.g., GPRS). Therefore, SpoVNet services and applications can be underlay-aware if required. This is assured by a component denoted *Cross Layer Information Overlay (CLIO)* that collects cross layer information in a distributed manner and provides them in an abstract way to applications and services. How components within the Underlay Abstraction use cross layer information provided by CLIO is discussed in the next section.

III. THE SPOVNET UNDERLAY ABSTRACTION

A. Overview

The Underlay Abstraction allows easy provisioning of advanced application- and network-oriented services. It supplies an identifier-based addressing scheme and basic connectivity by employing overlay techniques. Control messages are passed between SpoVNet Nodes in order to maintain the (overlay) network structure or to create transport links between nodes according to the transport requirements. Data messages can also be passed via the overlay at least.

The advantages for applications and services by using this abstraction are hiding of multi-homing, protocol heterogeneity, NAT/Firewall traversal, and mobility if requested. Furthermore, it provides basic security services and QoS support.

The Underlay Abstraction consists of two main components: the *Base Communication* and on top of that the *Base Overlay* (cf. Figure 2). The former acts on locators and handles transport protocol connectivity whereas the latter organizes the basic connectivity with addressing based on Node Identities. The Base Overlay is also used to exchange locator-related information that is required by the Base Communication.

B. Base Communication

The Base Communication offers transport services between SpoVNet Nodes to higher layers, e.g., for the Base Overlay as

well as for services or applications. The transport services provide the transmission of datagram messages (i.e., stateless one-shot messages) and the creation of direct or indirect transport links with given transport attributes (e.g., reliable/unreliable, secure/unsecure, QoS parameters, max. message size). The target of a message or link is identified by an *Endpoint Descriptor* that contains multiple locators (i.e., transport addresses and port numbers) as well as potential NAT relays. The Base Communication can generate an Endpoint Descriptor for the SpoVNet Device it is running on. On the other Hand, Endpoint Descriptors can be exchanged by nodes and are used for addressing in the Base Communication.

To overcome heterogeneity it is needed to transparently bridge between different protocols as well as NAT/Firewall traversal by the assistance of relays. There are two types of relays:

- *NAT relays* – publicly reachable nodes that can help to traverse NATs or firewalls
- *Protocol relays* – nodes that can relay to another network protocol (e.g., IPv4 to IPv6)

Local policy determines whether a node is able and willing to serve as a relay. The capability is then indicated in the Endpoint Descriptor so that nodes learn about potential relays automatically. More information on relay discovery is given in Section III-B4.

We start with a description of services that the Base Communication provides. Basically, messages can be sent with different transfer attributes, e.g., reliable/unreliable, secure/unsecure, with QoS requirements. We start with a special case of datagram messages that are transmitted unreliable and without establishing state. For other messages, it may be required to establish transport connections, e.g., for reliable or secure message transmission. This is, however, chosen automatically by the Base Communication corresponding to the given transfer attributes.

1) *Sending One-Shot messages*: One-shot messages are sent to the receiver unreliable, unordered, and usually error-free (e.g., UDP with checksum enabled). Furthermore, a priority can be specified in order to allow for preemption during bandwidth bottlenecks, e.g., more important control packets may preempt lower priority data packets. Appropriate locators are transparently selected.

2) *Creating direct transport links*: The application, service or Base Overlay specify requirements for message transfer that lead to creation of direct transport links by the Base Communication. The transport link is established using the best locator (see Section III-B5) according to the link requirements, data measured by CLIO or chosen by a heuristic.

When reliable transfer for data messages of the Base Overlay, or of services and applications on higher layers is requested, a transport link may be established as follows:

- a) If available, an SCTP connection is created using one stream for Base Communication control data and another stream for higher layer data (e.g., for data forwarded within the Base Overlay structure).

- b) If SCTP is not supported, multiple TCP connections are established.
- c) For all further links either an additional SCTP stream or TCP connection is used for higher layer data.
- d) A *transport link handle* is returned to the Base Overlay or higher layers, respectively.

Using different SCTP streams or different TCP connections is important, because data messages should not impede control messages for the Base Communication due to a single flow control mechanism. We prefer SCTP, since it is message oriented and provides separate flow and error control for different streams. Optionally, its multi-homing feature may be used, but this is beyond the scope of this paper.

3) *Indirect Communication*: Protocol heterogeneity and NATs or firewalls can hinder sending one-shot messages or creating direct transport links. In such scenarios using protocol or NAT relays must be supported by the Base Communication. Relay discovery is described in the following section. At this point, we assume that appropriate relays for the transmission between two nodes are known, establishing communication is straight-forward. A one-shot message from node a to node b , which is not directly reachable, is sent to an appropriate relay c that forwards the message to b . Similar, a transport link from a to b via a relay c can be established by creating a transport link from a to c followed by another one from c to b . However, this solution does not preserve the end-to-end property of the transport links since it requires to split the connection at c . To avoid split connections, the Base Communication uses packet encapsulation for forwarding data from the sender to the receiver. Communication over multiple relays is also possible by concatenating multiple tunnels.

In some situations, relaying for one-shot messages or transport links is not desirable. For instance, a service overlay may rather place a node at a different point in the overlay structure than use an indirect transport link for performance reasons. Thus, the Base Communication will use indirect connections only if requested by higher layers.

4) *Discovering Relays*: The Base Communication automatically detects such relays using different methods for NAT relays and protocol relays, respectively:

Discovering NAT relays — Recall that a node advertises NAT relays within its Endpoint Descriptor. Thus, the node itself must keep track of NAT relays for those locators that are behind NATs or firewalls. For deciding whether to maintain a NAT relay for a locator, the Base Communication provides functionality quite similar to *Simple traversal of UDP over NATs (STUN, [5])*. As described in more detail in Section III-C, joining a SpoVNet instance with a node j requires contacting an initial node i that is already part of the SpoVNet. i must either not be located behind a firewall/NAT or must use a NAT relay r . In the first case, i itself is a potential NAT relay for j , while in the second case r is a potential relay. Thus, j implicitly discovers an initial NAT relay when joining the SpoVNet, and, subsequently, learns about other NAT relays when communicating with other nodes. Thus, sufficient redundancy in NAT relay is automatically achieved.

Discovering protocol relays — In contrast to NAT relays, which are asymmetric in the sense of allowing to open a transport connection only in one direction, protocol relays are symmetric, i.e., they allow for protocol translation in both directions. Thus, protocol relays are not advertised in the Endpoint Descriptor of a node. Rather the Base Communication at each node maintains a list of protocol relays to use for connection establishment.

Nodes learn about protocol relays using a gossiping approach. That is, two nodes that communicate via the Base Communication will synchronize their lists as shown in Figure 3. Synchronization information can either be sent using one-shot messages or using a control link that is established in parallel to a transport link (see Section III-B2) by the Base Communication. These lists must be synchronized only if they changed since the last synchronization.

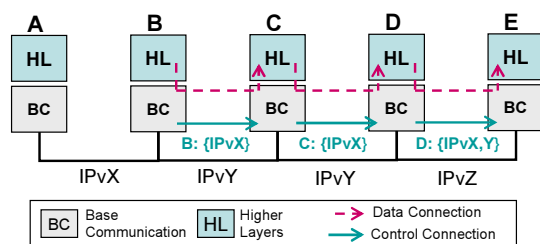


Fig. 3. Discovering relay nodes simultaneously to data transmission.

5) *Handling Multi-homing*: Recall that an Endpoint Descriptor of a node may contain multiple locators. This enables the Base Communication to select an appropriate locator for sending one-shot messages or establishing transport links. Selection can be done by different criteria, e.g., using only links with network protocols supported by the local node to avoid relaying, performance measurements provided by CLIO, or local policies.

Maintaining multiple locators provides redundancy, since in case of a locator failure another locator can be used. Although locator updates are communicated between nodes using the Base Communication control connections (see Section III-B6), transparent locator changes require testing the liveness of the locators in the Endpoint Descriptor. Whether a locator for sending one-shot messages is alive can be easily discovered by sending one-shot heartbeat messages. However, testing liveness of locators for transport links requires a three- or four-way handshake for TCP or SCTP connections, respectively. To avoid unnecessary overhead, alternate transport link connectors are only probed if this is requested by higher layers. In this case, transport links to alternate locators are maintained in parallel to the active transport link. Transport links will be used by overlays at the higher layers. With typical overlay approaches, a node will maintain connections to less than $O(\log N)$ nodes in a SpoVNet Instance with N nodes, making the probing of alternate locators scalable.

6) *Handling Mobility*: SpoVNet offers optional mobility support, since the Base Communication will transparently sustain transport links during locator changes. This support

differentiates between two cases. In case that a mobility-induced handover can be anticipated, e.g., by information received from CLIO, new locators can be exchanged using the Base Communication control connection before the handover takes place. Otherwise, the node that has moved is responsible for reestablishing transport links with other nodes, since with high probability, the locator sets or Endpoint Descriptors of the other nodes will still be valid after the handover. In the rare case that two connected nodes move at the same time, the Base Overlay has to be used for detecting new locators and reestablishing the connection. Finally, according to the SpoVNet philosophy, the Base Communication will use network mobility support, e.g., MobileIP [6], [7], if present.

7) *Providing Basic Security*: If a secure message transfer is requested, a secured transport link will be established automatically. Depending on the set of available mechanisms, e.g. TLS or IPsec, and given, or automatically implied, security requirements, the protocol for securing the transport link will be chosen. Keys, if necessary for the security mechanism, will be provided by key exchanges using the private/public key pairs of $CNN_S(a)$ or $CSN(S)$, respectively, introduced in Section II. Communication over a secured link is transparent for upper layers and does not necessitate adaptation.

8) *Providing QoS Support*: Due to space limitations we sketch the concepts for supporting QoS in SpoVNet only briefly. On the one hand, transport overlays try to meet given QoS constraints by corresponding adaptation of their overlay structure. Cross-layer information and QoS monitoring data provided by CLIO is used for this purpose and allows for building dissemination structures optimized to the application's special needs. Overlays or applications signal their QoS demands to the Base Communication which then chooses appropriate protocols for direct transport link establishment. On the other hand, although the overlays are underlay-aware, strong guarantees cannot be given without native QoS support in the underlay. If the underlay supports QoS, the Base Communication will use it, e.g., by requesting QoS guarantees for a direct transport link within DiffServ domains using NSIS signaling [8] where available. This requires a mapping of SpoVNet QoS parameters to underlay QoS parameters.

C. Base Overlay

The Base Overlay provides basic connectivity between all nodes in a SpoVNet Instance. In contrast to the Base Communication, that uses underlay specific locators for addressing, the Base Overlay uses SpoVNet-specific node identifiers or optionally canonical node names according to Def. 2. Thus, a main purpose of the Base Overlay is resolving node identifiers to locators, or more specifically, Endpoint Descriptors. For that purpose, the Base Overlay uses key-based routing techniques within a structured overlay topology. Based on this functionality the Base Overlay enables services and applications to send datagram messages or create transport links to arbitrary nodes that are addressed by node identifiers. In fact, together with the Base Communication, the Base Overlay hides most of

the complexity of the underlay from services and applications, and, thus, enables easy creation of service overlays.

Since the Base Overlay organizes nodes within a SpoVNet Instance, creating or joining an instance is synonymous with creating or joining the Base Overlay. Furthermore, the Base Overlay provides mechanisms for authentication and authorization that can be applied during the join process. Thus, the Base Overlay is a key component for establishing trust among nodes in a SpoVNet instance.

The Base Overlay is implemented using transport links provided by the Base Communication. It may use an arbitrary overlay routing scheme (like Chord [9]). On top of such routing scheme, the Base Overlay can also support standard DHT operations. The remainder of this section describes the Base Overlay's main functionality in more detail.

1) *Creating a SpoVNet Instance*: A SpoVNet Instance S is associated with certain attributes. First of all, it has a canonical SpoVNet name $CSN(S)$ (cf. Def. 1) to allow nodes to join S . As aspects related to key-based routing, attributes constitute the overlay routing scheme and the ORCHID-based function $chic_S$ that maps the canonical node name $CNN_S(a)$ of a node $a \in S$ to the node identifier $ID_S(a)$ (cf. Def. 2). Security related aspects include authentication and authorization mechanisms (see Section III-C3). Additionally, a SpoVNet Instance may be *hidden*, i.e., a credential is even required to discover its existence. All these attributes are set by the initiator $START(S)$ when it creates instance S .

Formally, for the initial set of nodes participating in a newly created SpoVNet Instance holds $S = \{START(S)\}$. The Base Overlay (cf. Def. 7) is a graph with a single node $START(S)$ and an empty set of edges $E = \emptyset$. From the technical point of view, the initiator $START(S)$ sets up all overlay routing tables for an overlay with a single node.

2) *Joining a SpoVNet Instance*: A joining node j has to discover a set of *Bootstrap Nodes* B first. Bootstrap Nodes are not necessarily part of S , but will provide a list of *initial nodes* $i \in S$ that are configured to assist other nodes with joining, i.e., integrating them into the Base Overlay. Next, j will connect to an initial node i that executes the authentication mechanism used by S to verify j 's identity. Furthermore it is i 's responsibility to perform authorization on j . After successful authentication and authorization i provides the Endpoint Descriptor of at least one of j 's neighbors within the overlay structure so that j can finally join at its dedicated correct position.

In more detail, joining a SpoVNet Instance S requires the following steps: *Getting Initial Nodes* — To discover bootstrap nodes, the joining node j sends a *discovery message* to a well known multicast or anycast address. Discovery does not need to be Internet wide. It is rather sufficient to discover bootstrap nodes in the local multicast or anycast domain. Thus, the insufficient deployment of multicast / anycast does not hinder the bootstrapping process. The discovery message contains the $UUID(S)$ (cf. Def. 1) of the SpoVNet Instance S j wants to join. Furthermore, in case of hidden SpoVNets, these items are encrypted and the message may contain a credential that

triggers the nodes of a hidden SpoVNet to identify themselves (the encryption key is related to the credential). This prevents eavesdroppers from detecting members of a specific hidden SpoVNet. It will only be known that there is a node running a SpoVNet Instance. A bootstrap node $b \in S$ that receives the discovery message replies with a *discovery response message* containing Endpoint Descriptors of initial nodes in S .

Alternatively to the decentralized approach described above, the Endpoint Descriptors of a set of bootstrap nodes may be supplied by other (out-of-band) mechanisms, e.g., by downloading them from a central server.

Contacting initial nodes — After a set I of initial nodes has been discovered, the joining node j chooses an initial node $i \in I$ that will help to integrate j into the SpoVNet Instance S . Since the Endpoint Descriptor of i is known from the bootstrap process, j can use the Base Communication to authenticate its identity and send authorization credentials needed by i to invoke the authorization mechanism of the SpoVNet Instance (see Section III-C3). If j is successfully authorized by i , an *authorization credential* is passed to j by i . The credential is used as proof of authentication and authorization when j establishes connections to other nodes within S .

Integrating into the Overlay — To integrate itself into the Base Overlay, j has to determine its correct position within the overlay structure. Using a simplified description of the join process for ease of exposition, this is done by routing a message to $ID(j)$. Since j is not yet part of the overlay, the node identifier will not be found, but the routing will terminate at a node k that is responsible for the value represented by $ID(j)$. k is an initial neighbor of j within the overlay structure and can help j to determine other neighbors.

To implement such an algorithm within SpoVNet, the joining node j will route a *descriptor query message* towards $ID(j)$ using the key-based routing functionality provided by the Base Overlay. Note that j can access this functionality via the initial node i , that is part of the overlay. k will return a *descriptor response message* with its own Endpoint Descriptor to j via key-based routing towards i . Subsequently, j can use the Base Communication to establish a transport link to k using the Endpoint Descriptor, and proceed with integrating itself into the overlay with the help of k .

3) *Authentication and Authorization Mechanism*: To prove ownership of $ID_S(a)$ node a sends a message signed with $CNN_S(a)$ to node b , using $ID_S(a)$ as sender address. Node b recomputes the hash using h_S , compares it with with the lower bits of the sender address $ID_S(a)$ and verifies the signature. The same mechanism can be used by a SpoVNet Instance $UUID(S)$ to authenticate itself using $CSN(S)$ as public key for the *chic_S* function (cf. Def. 3). This way ownership of the $UUID(S)$ and $ID_S(a)$, respectively, can be proven and thus spoofing attacks prevented efficiently.

Authorization is enabled using a centralized and, for scalability and load balancing reasons, a decentralized scheme. With centralized authorization, the credential of the joining node j is sent to initiator $START(S)$ using key-based routing. $START(S)$ validates the authorization of j and generates

the authorization credential that is passed to j by key-based routing through the initial node i . In the decentralized authorization scheme $START(S)$ may authorize other nodes within the SpoVNet Instance to perform authentication, e.g. by signing their CNN_S with its private key part of $CSN(S)$.

4) *Creating Transport Links from Services and Applications*: With support of the Base Overlay’s ability to resolve node identifiers to Endpoint Descriptors, applications and services are capable to send messages and create transport links to remote nodes. However, they would have to call both functionality from the Base Overlay *and* the Base Communication on each operation. To encapsulate the entire process, the Base Overlay provides functionality for sending datagrams and establishing transport links quite similar to the Base Communication. The major difference is that when calling the Base Overlay’s functionality, addressing relates to node identifiers rather than Endpoint Descriptors.

Applications and services can use the Base Overlay to establish a transport link to a certain node n addressed by $CNN_S(n)$ or $ID_S(n)$, respectively. As a first step to establish the link, the Base Overlay sends a descriptor query message over the Base Overlay using key-based routing, and receives a descriptor reply message as described in Section III-C2. The Endpoint Descriptor contained in the reply message is used to set up a transport link using the Base Communication. The transport link handle is returned to the higher layers. During key-based routing of the descriptor, request and reply messages the Base Communication will discover protocol relays on the overlay path as described in Section III-B4. This ensures that relays between the endpoints of the transport link are always known, so that the link can be easily established by the Base Communication. Transport links created by the Base Overlay have the same QoS- and security parameters as transport links created by the Base Communication.

Similar to handling transport links, the Base Overlay can provide the functionality for sending a datagram message to a node addressed by the node identifier. This can be achieved by resolving the node identifier to the Endpoint Descriptor and use the latter for calling the Base Communication’s functionality to send a datagram. However, the process of key-based routing for resolution of the node identifier may induce more overhead than the datagram delivery itself. Thus, the Base Overlay also provides functionality to deliver datagrams by key-based routing without performing an explicit resolution of the target’s node identifier.

IV. RELATED WORK

Many diverse solutions to specific problems imposed by heterogeneity, mobility and multi-homing are currently developed within the IETF, e.g., MobileIP [6], [7], SHIM6 [10], and HIP [11]. While most of these solutions focus on a specific aspect of the problem space, the *Host Identity Protocol (HIP)* has objectives quite similar to the SpoVNet Underlay Abstraction and, thus, shares a part of its functionality. HIP also uses an ID/Locator split and a binding of cryptographic identities to addresses. Most of the functionality provided by HIP is located

in the SpoVNet Base Communication. The SpoVNet underlay abstraction, however, uses a more integrated approach since it also includes the Base Overlay for locating nodes and resolving node identifiers to locators, whereas HIP employs an external mechanism like DNS or a global DHT.

One approach for providing node connectivity based on overlay networks is the *Internet Indirection Infrastructure* i^3 [12]. i^3 uses nodes within an overlay as rendezvous points for transmitting data from a sender to a receiver. That is, i^3 uses indirect connections, whereas the SpoVNet Underlay Abstraction uses direct connections between two nodes wherever possible. Furthermore, i^3 is not designed to cope with heterogeneity in the network layer.

The *Ambient Networks* project shares many of SpoVNet's overall objectives, in particular its *SATO* sub-project [13]. SATO enables the flexible creation of *service aware transport overlays*. While SpoVNet service overlays are built on top the transport layer, SATOs built upon the network layer. Suitable network layer protocols are IPv4, IPv6 or the Ambient Networks *Node Identity (NID)* layer [14], where only the latter can handle heterogeneity, mobility and multi-homing. While SATO provides functional building blocks for the composition of new overlay services (e.g., caching, transcoding), the Underlay Abstraction supports overlay service creation by providing transport connectivity based on Node IDs. Thus, the concepts developed by SATO and SpoVNet can complement each other.

The NID architecture and the Underlay Abstraction both hide heterogeneity and mobility from higher layers. However, the approaches are quite different. NID splits the participating nodes into *locator domains (LDs)* with consistent internal addressing and routing system (e.g., all IPv6 nodes or all IPv4 nodes in the same NAT-ed subnet). LDs are connected by *NID routers* and organized in a tree-like hierarchy. Messages routed between two NID nodes in different LDs must always follow the hierarchy, whereas only the initial message between two SpoVNet nodes is routed along the Base Overlay and a direct transport connection is created for subsequent messages. Furthermore, NID provides a new network layer based on the node ID name space, whereas the Underlay Abstraction provides transport links based on SpoVNet Node IDs.

As fundamental difference to other approaches SpoVNet uses a separate name space per SpoVNet instance whereas, e.g., HIP or NID provide a global name space that is shared and commonly used by different applications. Thus, SpoVNet offers a better isolation: each SpoVNet instance is associated with its own SID, probably hidden and protected by an access control mechanism. This makes it more difficult for attackers to launch certain attacks like sybil attacks or DoS attacks.

V. CONCLUSION

This paper presents the Underlay Abstraction as part of the *SpoVNet* architecture, an efficient way of meeting today's and future requirements for providing flexible communication. The architecture deals with problems such as bringing together heterogeneous access technologies, different protocols,

user mobility and multi-homing while also considering QoS-demands, group communication concerns and security issues.

The Underlay Abstraction allows for an easy provisioning of advanced application- and network-oriented services by supplying an identifier-based addressing scheme and basic connectivity by employing overlay techniques. The Base Communication offers transport services between SpoVNet Nodes to higher services and is able to overcome communication barriers like NATs/firewalls or different protocols by transparently using relay nodes. Multi-homing and mobility are handled by maintaining locator sets and choosing appropriate connection endpoints. The Base Overlay connects all nodes in a SpoVNet Instance, using node identifiers and applying an overlay routing scheme. Thus, its main purpose lies in resolving these identifiers to Endpoint Descriptors, enabling services and applications to send datagram messages or create transport links to other nodes. With its Underlay Abstraction, the SpoVNet architecture is able to hide all major network issues from higher services and applications, enabling them to be provisioned flexibly, adaptively and spontaneously on top of heterogeneous networks.

REFERENCES

- [1] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," *IEEE Selected Areas in Communications*, vol. 20, no. 8, pp. 1456–1471, 2002.
- [2] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "OverQoS: An Overlay Based Architecture for Enhancing Internet QoS," in *Proc. 1st Symp. on Networked Systems Design and Implementation (NSDI 2004)*, San Francisco, CA, USA, 2004, pp. 71–84.
- [3] "Hamachi website," <http://www.hamachi.cc/>.
- [4] P. Nikander, J. Laganier, and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)," RFC 4843, Apr. 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4843.txt>
- [5] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," RFC 3489, Mar. 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3489.txt>
- [6] C. Perkins, "IP Mobility Support for IPv4," RFC 3344, Aug. 2002, updated by RFC 4721. [Online]. Available: <http://www.ietf.org/rfc/rfc3344.txt>
- [7] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6," RFC 3775, Jun. 2004. [Online]. Available: <http://www.ietf.org/rfc/rfc3775.txt>
- [8] J. Manner, G. Karagiannis, and A. McDonald, "NSLP for Quality-of-Service Signaling," Internet-Draft, Feb. 2008, draft-ietf-nsis-qos-nslp-16.txt. [Online]. Available: <http://tools.ietf.org/id/draft-ietf-nsis-qos-nslp-16.txt>
- [9] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, Feb. 2003.
- [10] E. Nordmark and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6," Internet Draft, Feb. 2008, draft-ietf-shim6-protocol-10.txt. [Online]. Available: <http://tools.ietf.org/id/draft-ietf-shim6-protocol-10.txt>
- [11] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) Architecture," RFC 4423, May 2006. [Online]. Available: <http://www.ietf.org/rfc/rfc4423.txt>
- [12] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 205–218, 2004.
- [13] Ambient Networks, "D12-F.1 System Design of SATO & ASI," Dec. 2006. [Online]. Available: http://www.ambient-networks.org/Files/deliverables/D12-F.1_PU.pdf
- [14] B. Ahlgren, J. Arkko, L. Eggert, and J. Rajahalme, "A Node Identity Networking Architecture," in *9th IEEE Global Internet Symposium*, April 28-29 2006, Barcelona, Spain.