SpoVNet and Service Overlays

Lukas Schwoebel

***Abstract*—Looking at today's requirements of internet applications and services, we find there are several problems that are not suitably supported by the network architecture such as security matters, multicast, or *Quality of Service (QoS)* [1]. One could find many solutions, but they are often neither natively supported nor universal, thus often specialized for one single case. Furthermore other problems appear such as redundancy due to overlaps and binding on technologies and protocols like the different versions of IP which makes it harder to deploy applications on other devices. *Spontaneous Virtual Networks (SpoVNet)* is an approach to provide a global solution for these problems with transparent support for mobility, multi-homing, QoS, and network-heterogeneity by introducing an additional layer above the transport layer that can be build on by application. SpoVNet also provides a service overlay with different services which can be used by applications as building blocks. In this paper we will discuss the key features of SpoVNet and present the service overlay with main focus on an event-service and group-communication-service.**

I. Introduction

The development of P2P-applications with today's network architecture and infrastructure is a difficult task for several reasons. We take a short glimpse at some of the problems looking at a common application and its challenges: Video streaming and especially its special case, videochat in heterogeneous systems.

In order to deploy this kind of application in a P2P-manner without central instance, the usage of multicast is obvious, as in a video-chat the overhead needs to be as minimal as possible to get the videostream nearly in real-time. From the developers' point of view this is the first big problem that needs to be faced, because multicast is often not available as native solution. Even if a *Internet Service Provider (ISP)* is offering native multicast, the packets will be lost or simply not forwarded beyond its own network. Currently there is no way to propose applications using multicast independently from the ISP – with using a native way on the network-layer. Therefore the idea is self-evident to use *Application Layer Multicast (ALM)*, so employ the solution on the application layer with using the Narada protocol [8] for example.

But we have to face the even bigger problem in heterogeneous systems: Every solution is using its own, homogeneous network protocol and therefore it is hard or even impossible to deal with an exchange of the protocol or communication between heterogeneous devices – meaning the usage of IPv4 or IPv6. Thinking about the news of allocation of the last IPv4-address-block in February 2011 [14], a quick look in the future exposes that we have to face the change from IPv4 to IPv6.

SpoVNet is one approach to solve these and many other problems in an efficient way. It is based right on top of the network layer introducing an *Underlay Abstraction Layer* . The developers present several reasons to use SpoVNet, the main argument is the generic, central interface that is provided to use the network underlay without any required knowledge – as every application and service is using this interface, the available network resources can be used in a more efficient way because only one instance is handling the real network communication. But also problems like mobility, multihoming, network heterogeneity, indirect communication for peers behind firewalls or *Network Address Translation (NAT)* devices [2] and switching between native or application layer based support for QoS or multicast are faced and done seamless and transparent. Furthermore SpoVNet provides basic security and an efficient way for measurements of several attributes.

SpoVNet uses overlays to face most of the key-problems, as they are self-organising and if properly designed also scalable and robust. As there exists no central instance like in traditional systems it is harder to find participants of a P2P application. Therefore an overlay is needed where all participating peers are organised in order to communicate with each other. The overlay is abstracting from the actual network infrastructure underneath, thus a connection between two peers might be a direct link in the overlay but in reality consists of a route involving dozens of routers and links. Also, neighbors in an overlay might be hundreds or even thousands of miles far away from each other. Furthermore most overlays use their own addressing-scheme. While the network-underlay uses IP-addresses and ports most, overlays are based on *identifiers* that are usually indeed based on IP-addresses but nevertheless there is no way to conclude from the ID to the IP-address – which is an abstraction from the network on the one hand and a security mechanism on the other hand.

In this paper we first look at the key concepts of SpoVNet in Section II and give some formal definitions of a SpoVNet instance. Section III describes the main components of SpoVNet with focus on the Base Communication and Base Overlay-layers and a short survey over the *Cross Layer Information Overlay (CLIO)* interface that provides various information about the network and lastly the security component. Section IV covers the Service Overlays provided by SpoVNet and how applications can use them, explaining with the two main examples *Massive Multiplayer Online First Person Shooter (MMOFSP)* and video streaming. Section V presents related concepts and other approaches similar to this project. Finally in the last two sections the drawbacks of SpoVNet are discussed and then features and benefits of SpoVNet are summarised and the future work is presented.

[1] quality requirements in network communication

[2] translation of ip-addresses between different networks

II. SpoVNet overview and definitions

SpoVNet introduces an abstraction layer that separates the actual network-handling and communication from program logic in applications by introducing a generic interface for applications in order to use a seamless and transparent way for *technology-independent* communication.

The main core of SpoVNet consists of two components: The *Base Communication* layer that is providing end-to-end communication, connection-less as well as connection-oriented, and handles NATs and firewalls. The second component is the *Base Overlay* that is responsible for addressing and key-based routing.

At first we formalise some definitions concerning SpoVNet and its overlay: [5]

Definition 1: A set S of SpoVNet-nodes that are cooperating in an application is called an *SpoVNet instance*. S has a Canonical SpoVNet Name $CSN(S)$which is unique for S. Furthermore to identify the instance S there is the Unique Universal Identifier $UUID(S)$, calculated by $chic_S(CSN(S))$

Definition 2 ($chic_S(a)$): cryptographic hash identifier construction function, which is using the *Overlay Routable Cryptographic Hash Identifiers (ORCHID)* scheme [16] dependent on the SpoVNet instance S. For $chic_S(a)$ there is the ORCHID-defined prefix p and a global, SpoVNet specific context id c, furthermore the two functions e_S (extraction-function) and h_S (hash function) that are used by $chic_S(a)$. Therefore, $chic_s(pk)$ is computed as follows: $p|e_s(h_s(c|pk))$[3].

Definition 3 ($START(S)$): the initiator node of the SpoVNet instance S

Definition 4 ($CNN_S(a)$): Canonical Node Name, name of a specific node a in the instance S that is unique within the instance, furthermore to identify a node a, each has an unique identifier $ID(a)$ within the instance S, calculated by $chic_S(CNN_S(a))$

There are two forms of connectivity between nodes in the SpoVNet overlay:

Definition 5 ($C_d(a,b)$): Direct underlay connectivity: Two nodes are able to send and receive packets directly in the overlay, formally represented by the function $C_d : S \times S \rightarrow \{true, false\}$. For two nodes $a, b \in S$ the connectivity is given by $C_d(a,b) = true \Leftrightarrow$ (a is able to send packets to b directly)

Definition 6 ($C(a,b)$): Indirect connectivity: Nodes might have to rely on other nodes as relays because of NATs or firewalls, so no direct connection is possible. Formalised as function $C : S \times S \rightarrow \{true, false\}$ with two nodes $a, b \in S$ indirect connectivity is given if $C(a,b) = true \Leftrightarrow \exists x_1, x_2, ..., x_m \in S, x_1 = a, ..., x_m = b, C_d(x_1, x_2) \wedge C_d(x_2, x_3) \wedge ...C_d(x_{m-1}, x_m)$ for m $\geq$ 3. Direct underlay connectivity is a special case with m = 2

So node a can prove its ownership of $ID(a)$ by using $CNN_S(a)$ as input for the $chic_S$ function. The same applies for a SpoVNet instance S that proves its ownership of $UUID(S)$ by using $CSN(S)$ as input.

[3]here the | means concatenation of two strings

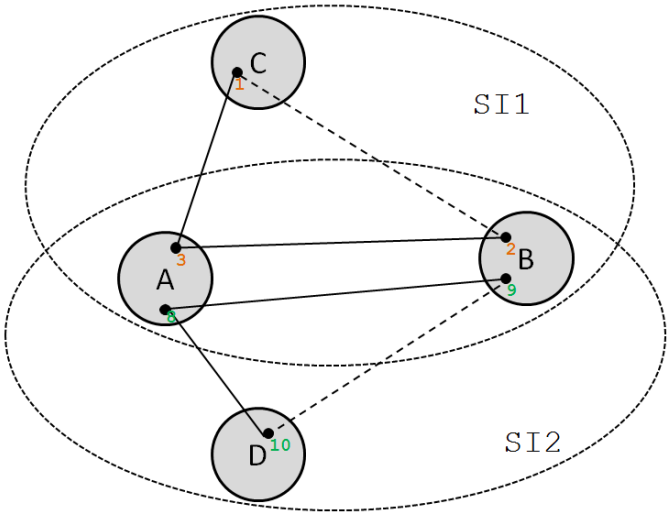

Fig. 1. Illustration of two SpoVNet instances SI1 and SI2 with each three nodes.

Using this definitions we can define *transport links* and the *Base Overlay* in a formal way:

Definition 7 ($T(a,b)$): A transport link $T(a,b)$ is a bidirectional transport connection between the nodes a and b with $a, b \in S$. If $C(a,b) \wedge C(b,a) = true$, such a link can be created and is called *indirect link* if $C_d(a,b) \wedge C_d(b,a) = false$, otherwise it is a *direct link*.

Definition 8 ($B_S(S,E)$): The *Base Overlay* is a connected[4], undirected graph containing all nodes: $B_S = (S, E)$ with S as SpoVNet instance and the set E containing all edges of the graph: $E \subseteq \{\{(a,b\}|a,b \in S, C(a,b) \wedge C(b,a)\}$. Per definition there exists a transport link T(a, b), $\forall a, b \in S$.

In figure 1 an example setting is illustrated with two SpoVNet instances SI1 and SI2. SI1 contains three nodes A, B, and C. Dependend on SI1 node A was assigned ID 3, node B has ID 2, and node C has ID 3. Between A and B, and A and C exist direct links, so $C_d(1,3) = true$ and $C_d(2,3) = true$. Between 1 and 2 there is no direct link but the dashed line implies there exists an indirect link. It is $T(1,2)$, $T(1,3)$, and $T(2,3)$

There is a second SpoVNet instance SI2 containing the node D and also A and B with the respective IDs 8, 9, and 10. D doesn't know anything about the existence of C or the fact, that A and B are joined in another SpoVNet instance. It is $C_d(8,9)$ and $C_d(8,10)$ as well as $C(9,10)$ and at last $T(8,9)$, $T(8,10)$, and $T(9,10)$

We differentiate between three layers: the application-layer, the SpoVNet service-abstraction-layer and the SpoVNet-Base. Figure 2 shows the different components: At the very bottom

[4]In some cases the overlay can split into two or even more parts, but this is beyond the scope of this survey

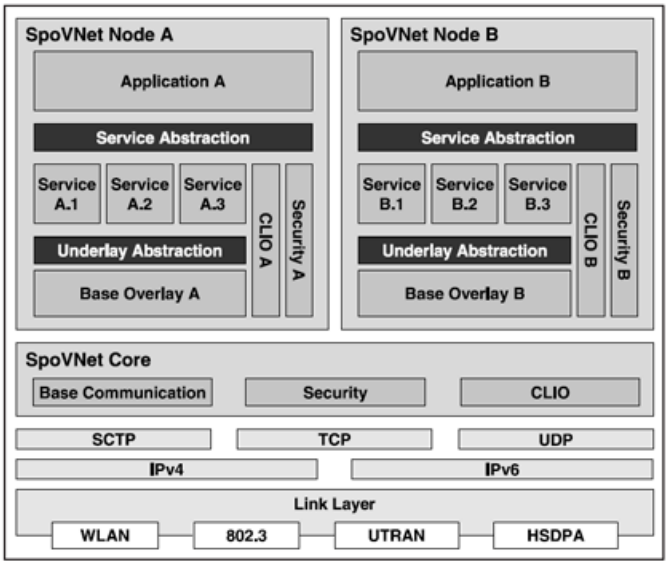

Fig. 2. The SpoVNet Architecture and the components of the SpoVNet Underlay Abstraction [21]

there is the *Link Layer* which consists of the used technologies like WLAN, 802.3[5] or HDSPA, on top of that on the *Network Layer* the versions of IP and on the *Transport Layer* the communication-protocols like *Stream Control Transmission Protocol (SCTP)*[6], TCP or UDP. Based on that standardised layers the *SpoVNet Core* is build, consisting of components used for global purposes – the *Base Communication* responsible for communication between devices, the *Security Component* and *CLIO*. The SpoVNet Core is running once per device. Based on that core, there is the *SpoVNet Node*, the logically representation of a device within a SpoVNet instance. As the figure proposes, every application is requirering its own running instance, so every application is running an own overlay. Part of a node and therefore unique for every SpoVNet instance is the Base Overlay component, the local security and CLIO instance and several services that are used by the current application. As seen in the figure, within the *SpoVNet Node* there are two abstraction layers: The *Underlay Abstraction* for abstracting from physical devices and addresses by using instance-specific identifiers and on top of that the *Service Abstraction* that is providing generic interfaces for special use cases, like multicasting – or access to the information gathered by CLIO. In the following sections every of these components are presented in detail.

III. The Components of SpoVNet

As mentioned, there are different components in SpoVNet, in this section the SpoVNet Underlay abstraction is explained, looking at the components of the SpoVNet Base, the CLIO interface that provides network-information, and the security component.

Base Communication

The Base Communication is the undermost layer of SpoVNet, providing transport services between various endpoints to higher layers. Endpoints in the network are identified by sets of *network locators*. One locator basically contains the address and port a device listens on, as well as the used protocol. The locator-set is non-empty, thus contains one or more locators. Each node provides an Endpoint Descriptor containing the locator-set and further information like relays. In the following the services provided by the Base Communication are described.

As the name already implies, the Base Communication is mainly responsible for providing a way to communicate with other nodes. At first, for communication there are two ways, depending on the purpose of the communication messages can either be sent unreliable or with acknoledgement. The first way is done with *One-Shot-Messages*. They are sent unreliably, unordered but error-free using UDP with checksums. To guarantee that important messages are transmitted even during bandwidth bottlenecks, a priority can be set - e.g. for control-messages. If this is not sufficient, it is possible to create *direct transport links* with usage of SCTP which includes a seperate stream for control data – or, if not available, TCP is used with multiple connections.

Furthermore if needed, basic security and QoS can be provided for communication: A secured transport link can be established by request, using the available technologies like *Transport Layer Security (TLS)*[7] or IPsec[8]. If necessary keys will be provided by a key-exchange using the private/public key-pairs from the above definition of $CNN_S(a)$ or $CSN(S)$. The communication via secured links is transparent, thus "invisible" for upper layers, therefore no adaption in a respective application is necessary. By using cross-layer information and QoS monitoring data provided by the CLIO interface, the created overlays also try to meet certain QoS constrains by adaptation of the overlay structure. The applications signal their needs for QoS to the Base Communication which then chooses appropriate protocols for communication and establishes respective transport links. If it is available, the Base Communication will use native QoS by requesting guarantees for transport links, e.g. by using the *Next Step in Signaling (NSIS)* suite for signaling the needs. Basically the NSIS uses a special protocol for signal messages that can be prioritised [1]. However, if a native QoS concept in the underlay is not available or cannot be used, it is not possible to give strong guarantees but provide probabilistic guarantees only.

Now we have to face a problem in P2P-applications that has to be considered from the beginning, as otherwise the whole communication process might fail. Sometimes it is not possible to reach nodes directly, thus one-shot-messages or direct transportlinks are not possible and therefore secured channels or QoS-constrains don't make sense either. To solve this problem, indirect communication is used with relays – nodes that are used as rendevouz points. There have to be

[5]Ethernet LAN
[6]transport layer protocol like TCP or UDP
[7]successor of SSL, cryptographic protocol
[8]protocol suite for securing the internet protocol

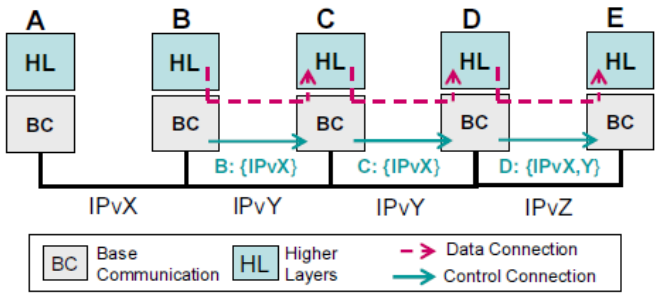

Fig. 3. Discovery of relay nodes simultaneously to data transmission [5]

considered to cases of relays: NAT relays or protocol relays.

When middleboxes are used, usually devices behind them are in a different subnet, often they are using private network-addresses like 10.0.x.y or 192.168.y.z – for communication with nodes in the internet a translation of addresses is needed. This is the responsibility of NAT. In order for SpoVNet nodes to communicate with nodes behind such a middleboxes, a *NAT relay* is needed. A second possible reason to use NAT relays are firewalls that don't allow unlimited access. Nodes have to keep track of possible NAT relays used for their locators. So if a node joins, it contacts a "bootstrap-node", that is either directly connectable and then can be used as a NAT relay itself, or it uses a NAT relay already, that can be used for the newly joining node. So a node discovers a NAT relay automatically on joining and learns about other relays while communicating with other nodes because NAT relays are advertised in the Endpoint Descriptor.

If two nodes want to communicate with each other but are using different protocols like IPv4 or IPv6, a *protocol relay* is needed. NAT relays are asymmetric, allowing transport connections in one direction only - while protocol relays are symmetric. Thus they are not advertised like NAT relays and they are "harder to find". The Base Communication layer uses gossiping to learn about protocol relays, so nodes are sending the lists of their supported locator-protocols. Figure 3 shows the process: node B sends a message to its neighbor C using the protocol "IPvY", simultaneously advertising its support for "IPvX". As soon as C is sending a message to D using "IPvY", too, it is gossiping that it knows a protocol relay for "IPvX" et cetera. The overhead usually is very low as the lists must be synchronised only if they change.

One more problem this underneath layer covers is the usage of multiple locators. Mobile devices can be connected via UMTS, WLAN, LAN, or maybe even Bluethooth, thus are able to change their type of connection if one fails. Due to this Endpoint Descriptors may contain multiple locators, thus multiple supported protocols. This provides redundancy, but for transparent changes of a locator, their liveliness has to be tested. As this would result in communication overhead like a three or even four-way-handshake using TCP or SCTP, alternate connectors are only tested if explicitly required by higher levels. Typically, nodes will hold connections to $O(logN)$ other nodes with N nodes in the respective SpoVNet instance.

On the other hand the locator-change is done seamless, if a mobile device changes its location and therefore a locator change is needed, SpoVNet supports the sustaining of transport links. Either a needed change can be foreseen e.g. with CLIO-information or the node reestablishes its connection on its own. For the rare case, that two devices change their locations at the same time, they can use the overlay to find each other and place themselves in a new position. This is done seamless without the application and the user knowing it.

Base Overlay - layer

On top of the Base Communication layer the Base Overlay is placed. Its main purpose is to resolve node-identifiers $ID_S(CNN_S(a)))$ to Endpoint Descriptors. It provides several functions that are now described:

Every SpoVNet instance S has certain attributes. First of all, as described in Section II in the definition section it has a canonical name $CSN(S)$ in order to allow other nodes to find and join S. Furthermore there exists the instance specific function $chic_S$ for mapping the canonical node name $CNN_S(a)$ of a node a to its identifier $ID_S(a)$. Additionally, it is possible to hide a SpoVNet instance, so it cannot be found by everyone without knowledge about it.

The initiator $START(S)$ is setting these attributes to create a SpoVNet instance. Formally, the initiator sets $S = \{START(S)\}$ and $E = \emptyset$. To join an existing instance, a joining node j has to get to know at least one bootstrap node which is not necessarily part of the instance itself, that delivers a list of initial nodes. One of these nodes is contacted, then it authenticates and authorises the new node and provides the Endpoint Descriptor of j's neighbors, at least one. The joining node gets the bootstrap-nodes by a "discovery message" sent by multicast and containing the $UUID(S)$. If the instance is hidden, a credential is sent, too, in an encrypted message, to trigger the nodes of the hidden SpoVNet to identify themselves. The whole joining process is similar to the join procedure in a Chord-overlay [20].

As in the joining-process an authentication is required, there need to exist cryptographic mechanisms to do so. A joining participant is creating a message and signs it with $CNN_S(a)$, using $ID_S(a)$ as sender address. To verify the ownership of $ID_S(a)$ the receiver can recompute the hash using h_S and compare it with the lower bits of the sender address. The same mechanism works for the whole SpoVNet instance to verify the ownership of $UUID(S)$ by using $CSN(S)$ as public key for $chic_S$. Authorisation for hidden instances works by a new node sending credential to $START(S)$ using key-based routing. The initiator now validates it and sends authorisation credential back to the joining node. To reduce load and for better scaling, $START(S)$ can authorise other nodes to do so, too.

Finally in the Base Overlay it is possible to create transport links, too. In order to avoid applications have to call both the Base Overlay and Base Communication functions for establishing transport links, the process is encapsulated. Thus, the process is in both layers similar, but the Base Overlay uses

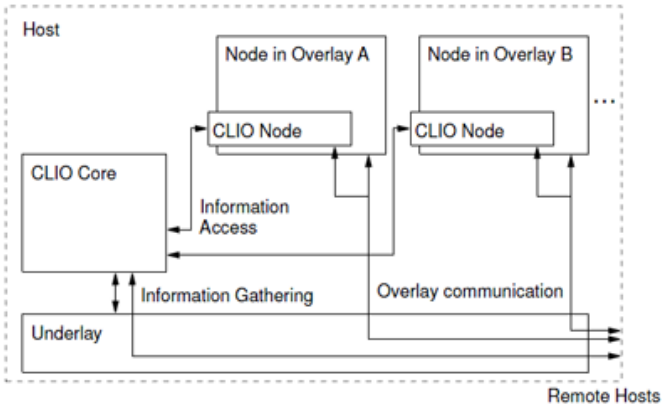

Fig. 4. The CLIO architecture and information flow [10]

node-identifiers instead of Endpoint Descriptors.

CLIO [10], [13], [21]

Every application and service using SpoVNet is creating its own overlay, as its requirements for QoS may differ widely from others. For example video-streaming via multicast doesn't meet the same requirements as distributed online games. So the different overlays need different optimisation techniques in order to meet the given requirements. These techniques require accurate information about the network – in SpoVNet this problem is addressed by introducing a special component called *Cross-Layer Information for Overlays (CLIO)*.

This service is splitted into two parts as proposed in [10]. Figure 4 shows the architecture, there is the CLIO core component that is responsible for the logic – like the measurements, monitoring, pre-processing and storing data. On the other hand there is the node component, that is basically an interface to access information from the core. On a device once per application and therefore once per overlay, an own SpoVNet node is running as shown in figure 2 already – and therefore once per application and overlay an own CLIO-node is running, too. The CLIO core gathers information about each participating device and its bandwidth but due to security reasons the data is collected in underlay-notation, so the real IP-address and ports etc. are used while the generic CLIO interface gets translated information, only. So each overlay gets information in the respective overlay notation. Hence it is impossible for a malicious node to gain knowledge about other overlays running on a specific host – otherwise this could be used to find hidden overlays.

The CLIO node-interface is organised as a *XML Remote Procedure Call (RPC)* interface, so it is a standardised connector that can easily be implemented.

The CLIO-core gathers and provides many data about the network and the nodes, in the actual implementation *UNISONO* there are about 50 dynamic and static properties of the underlay measured, whereas static information means connection types like WLAN or UMTS or multicast/broadcast capabilities. Values like bandwidth, number of nodes or loss-rate are dynamic properties. For nodes, there are values gathered like bandwidth, latency, CPU load and system properties like number of CPU-cores.

To keep the impact on the system as minimal as possible, data is reused whenever possible, e.g. by aggregating same or similar queries. Because data is collected only by request, the resource usage is nearly zero if CLIO is running but the device is not logged into any overlay.

The information CLIO measures could be critical if someone wants to exploit the system. Therefore the communication is secured with authentication, encryption and integrity-protection. There exists a Policy Engine that defines which measurements are allowed, the access to information about the measurements is defined via *Access Control Lists (ACL)*. Using limits and local policies the authors of [13] assume that for small networks about 10.000 nodes this protection can be sufficient. For bigger networks, the amount of malicious nodes can be too big, therefore attacks like *Denial of Service (DoS)* are a big problem.

Security Component

Facing current solutions and the internet architecture, security is kind of a patchwork – SpoVNet pursues *integrated security*. Similar to CLIO a security component is running once per node and joined overlay, maintaining and organising secure connections and keys. With this, the applications and services based on SpoVNet don't need to implement security aspects on their own. This implies two main benefits:

- Removing redundancy and therefore avoiding errors and failures
- Easy access for all components that are based on SpoVNet and its Underlay Abstraction

The first purpose of the Security Component is to provide secure communication between nodes. If a save channel is required, the component use the Base Overlay identifiers for key-exchange and saves the keys. The keys are saved in this component and used for secure connections that are maintained by the Base Communication layer. An appropriate communication protocol is chosen automatically. If no protocol like TLS is available on the device, the Security Component provides its own connection security, if possible based on available security mechanisms.

Furthermore the component provides an interface for sensitive operations like bootstrapping, joining, leaving. Also policies specified by a creator of a SpoVNet instance are managed like authorisation, authentification and confidentiality. Applications and services are bound to this interface in order to enforce the specified policies. Nevertheless, if secure functionality is required beyond the functionality of the Security Component, it has to be implemented by the applications themselves, using the basic functionality of the Security Component.

IV. The Service Overlay and Applications

With the Service Abstraction, SpoVNet proposes a layer with an extensible library of services that can be used like building blocks to build applications on top of them. The

services are organised as overlays, one for each service. In this section at first two services are presented that are currently provided by SpoVNet as example implementations and then two applications that are using these services.

Group Communication Service

The first service is trying to address the multicast problem in P2P. As native multicast in the network is not widely available and is usually limited to the network of the respective ISP most solutions are using the ALM concept. As there exists no standard implementation and such concepts like security or QoS are difficult to achieve, SpoVNet is offering a Group Communication Service or *Multicast/Multipeer-Overlay (MCP-O)* to be used for higher level applications. This is a scalable, efficient multicast-service where QoS and security can be enabled. Scalability and efficiency is achieved by the use of clustering, based on the idea of NICE [2] which is a protocol for scalable ALM, especially for low-bandwidth participants. The main idea is to build hierarchical clusters where equal peers respectively to their properties like bandwidth and ping between each other are organised in one cluster. With clustering the management traffic is small, because multicast-control-messages don't need to be sent group-wide any more. The idea of NICE is extended through the use of CLIO information to achieve better clustering, also members who support native IP multicast are put into the same cluster.

Basically the clusters are organised like trees, for each cluster there exists a cluster leader that is used for the actual data distribution within the specific cluster

To achieve security, there are several mechanisms. Confidentiality is provided by encryption with cluster keys, *Timed Efficient Stream Loss-Tolerant Authentication (TESLA)* [17] covers authenticity and integrity efficiently without required trust between sender and receiver - and is able to protect receivers against denial of service attacks in certain circumstances. It uses mainly symmetric cryptography for its mechanisms. However, it requires synchronised clocks for each party to work.

As far as reliability is concerned, a decision is needed whether to use WLAN multicast or ALM. WLAN multicast is efficient through the use of radio broadcast, but collisions can't be detected as there are no acknowledgments that are needed for reliable data distribution. To decide whether to use WLAN radio with some acknowledgment overhead or ALM, the protocol parameters are determined out of data provided by CLIO about the network structure like WLAN utilisation and used WLAN standard.

Event Service

With the *Event Service (ES)* SpoVNet provides a possibility for producers and consumers to publish and receive information. Consumers are able to subscribe to events respectively to their interests, similar to traditional content-based publish/subscribe systems [6].

To work in a efficient way, there are several roles for nodes contributing to ES: broker, subscriber, publisher, or correlator, whereas a node can change its role or act with respect to multiple roles. Subscriber, publisher, and broker can be found in common publish/subscribe systems whereas correlators are newly introduced.

At first we take a short look at the *broker*: Subscribers register for events at a broker. The set of brokers form an own *broker overlay* where the messages are filtered and forwarded and furthermore connectivity between subscribers and publishers is ensured. There are several algorithms to ensure QoS requirements like in [4]. The overlay is formed like a content-based distribution tree, in which the brokers can move bottom-up for achieving their own QoS requirements.

Lastly there are correlators forming a correlator overlay. Technically they are subscribers to basic events and publisher of composite events. Therefore they are constantly executing a correlation detection to determine events that can be composed. The rules for a correlation can be defined by the respective applications that are using ES. Basically a correlator is maintaining a list of publishers for each subscriber it is attending on. If two or more publications are occurring at the same time, the correlator is composing them. Due to performance and efficiency not every subscription is handled by a correlator. This is done only if the overhead of sending several single event-messages is bigger than the composition by a correlator. Relevant factors are link properties and capabilities of nodes.

Applications

To prove that SpoVNet is able to use these two services efficiently, not only in theory but under realistic conditions, the developers present two example applications that are considered to be high-demanding on efficiency issues. These are now presented shortly.

MMOFPS: There are many approaches to implement MMOFPSs in a P2P manner, one prototype is chosen to show SpoVNets capability of addressing the key problems. Whereas many decentralised approaches exist [7], SpoVNet tries to address all challenges instead of only a subset: scalability, game-dynamics and consistency. The implementation is using ES and MCP-O for updates concerning the *Area of Interest (AoI)*[9] or other criteria like friend or foe membership. Also, the quality of event transmission is guaranteed like evaluation and delivery of shot or movement events. Thus, applications can rely on the SpoVNet services that are capable of achieving probabilistic guarantees.

Video streaming: This application also uses MCP-O for scalability and ES for adaptivity to underlay constrains and, in addition, the mobility support of the SpoVNet Base Overlay. The developers employed *Scalable Video Coding (SVC)* [18] for video encoding which is using the h264/AVC standard. For distributed video streaming on mobile devices several cases have to be considered: If all peers are equal in bandwidth and latency, the MCP-O works perfectly for transmission – until one or more nodes suffer from a significant change of bandwidth which is usually a loss of bandwidth, the other way round is not that critical in the first place. Then there are two

[9]the direct surrounding area of a player, usually the set of players that are in a radius of some meters around

possible ways to handle this: either the concerned device can send an event using the ES to request the creation of a new multicast group with different message load. Another option is the support of clustering in MCP-O for devices with equal bandwidth. Here, the group leader can simply truncate video data as supported by SVC and distribute a stream with lower resolution or bitrate inside the new cluster.

V. Related Work

There exist many solutions for specific problems concerning heterogeneity, mobility and multi-homing, most of them focus on one problem, though. Nevertheless there are some examples that try similar approaches like SpoVNet that are now described.

First, the *Host Identity Protocol (HIP)* [15] covers most of the problems SpoVNet does. There's an ID/locator separation, a binding of cryptographic identities to addresses. The HIP covers pretty much only the Base Communication layer, whereas SpoVNet is more an integrated approach, using the Base Overlay layer, too, for locating nodes and resolving IDs to locators. Thus HIP is more comparable to an external mechanism like DNS or a global DHT.

The *Internet Indirection Infrastructure* i^3 [19] uses nodes as rendezvous points, so data is transmitted indirectly, whereas SpoVNet tries to establish direct connections, whenever possible. Thus this approach could be compared remotely with the use of supernodes like in Skype or KaZzaA. Heterogeneity with IPv4 and IPv6 is not covered anyways.

Ambient Networks[10] as a subproject of SATO builds upon the network-layer. IPv4, IPv6, or the Ambient Networks Node Identity (NID) layer are possible - multihoming, mobility, and heterogeneity are only supported by the latter. Both the NID and SpoVNet are hiding technical details of the network from higher levels, but there are different approaches. NID splits the network in small locator domains, e.g. all IPv4 and IPv6 participants which are organised in a tree-like hierarchy. Sending messages from one node to another is always done by routing through this "tree" and there is only one global "tree" for all applications and services. SpoVNet uses the overlay only during setup-phase and then establishes direct connections, considering the overlay-structure only, if a replacement is needed.

Last but not least there is exist a project with the aim to standardise P2P-applications called *JXTA* [3]. The motivation of JXTA is pretty much the same as of SpoVNet, to create a standardised protocol for applications. The project was founded by Sun Microsystems under leadership of Bill Joy and Mike Clary. JXTA consists of three layers, a core, the services and the application-layer. There are different overlays as well with optional restricted access if needed. Also there are concepts for network-heterogeneity, multicast and security. The main drawback of JXTA is the missing implementation as it only defines a standard and leaves the implementing to the users. Nevertheless it should be possible to communicate with all nodes that are employing the protocol. Furthermore several specifications are open, thus only a loose common base exists. Mainly because of the loose restrictions and the fact that its not bounded to any platform and programming language, JXTA seems like a very promising player. Currently it lacks of stable implementations and applications, though.

[10]no reference at this point as the project website is offline, referring to reference [13] in [5]

VI. Discussion

As SpoVNet and its implementation ariba were developed for P2P purposes it is obvious that using it with the traditional *Client-Server* pattern might not be very effective. So the advertised beneficials of SpoVNet such as multihoming, heterogeneous networks, et cetera apparently can't be usefully deployed for centralised applications.

The main problem with SpoVNet is the uncertainness if it is able to meet all the solutions it proposes. In August 2009 it was shown using a testing network that ariba is able to communicate via protocol relays between different devices using several technologies [11]. However the demonstration did not prove anything about the scalabity, efficiency, security, or overall stability and furthermore it was a static demo-installation and therefore didn't reveal if SpoVNet and ariba are able to meet the requirements of highly dynamic P2P networks – the main purpose of SpoVNet as advertised by the developers.

The current literature and papers provided by the developers is at some points a bit of a fudge as soon as it comes to concrete details about security or capability. Sometimes there are rough estimations given but often there are no details at all. So either there are no concepts yet or they are not suitable elaborated to be presented. So at this point, we can make our own estimations only.

If we are looking at an example for a P2P-application with many participants, the security aspect gets more and more important. *Safebook* [9] is a good example for showing how important security is, respectively to several matters. It is obvious in order to be attractive for users that this fully distributed social network requires a high amount of users to be accepted. At this point security is a big problem, as SpoVNet is relying on measurements of CLIO in order to build and optimise its overlays. The information gathered and provided by CLIO is very sensitive as it can be easily exploited in order to harm the whole network. The developers provide several security mechanisms to secure CLIO but as shown in first simulations they estimate that in bigger networks with more than 10.000 nodes some features have to be disabled in order to meet security requirements. As SpoVNet is separating itself from all other solutions because of the integrated CLIO-support for optimising the network, it is unclear if the overlay can still be optimised with a restriction of the CLIO functionality. In the case of Safebook it is obvious that this application is laid out for a big amount of users exceeding 10.000 participants.

Overall, the CLIO component seems to be problematic. The developers propose CLIO as one of the main reason that separates SpoVNet from other approaches, because the organisation of the overlay can be optimised with the use of CLIO-information. It is uncertain if the promised requirements can be met without CLIO. Even if there are approaches to

secure CLIO, it seems to be easy to exploit information as a malicious node. There are no detailed concepts for handling the delivery of false data for example. But furthermore, even though physical addresses are mapped to IDs in order to hide network-details and therefore provide a bit privacy for participating nodes, it could be possible to identify and keep track of devices beyond a respective overlay and independently of their IP-addresses and node-ids, just by using the CLIO-data – as there are over 50 items measured, providing a basis for a nearly unique profile.

Altogether SpoVNet is a good approach, but at this moment there is no stable implementation available and above all there is no prove of its capabilities in big networks, in particular how the security concepts are evolving. In theory for fully distributed applications it seems to be superior to related solutions, but as soon as one component in the application requires a centralised approach, there are some drawbacks as SpoVNet is not designed to integrate centralized components – if they are not running a SpoVNet instance.

VII. Requirements and Future Work

SpoVNet does not have any special requirements for a device's hardware or the network other than the current standard – meaning every device able to run a current Linux distribution or any mobile device running Android is possibly capable for SpoVNet. The applications based on SpoVNet are bound to the interfaces, the both abstraction layers Service Abstraction and Underlay Abstraction.

As SpoVNet is only the theoretical definition of the whole service, there exists an actual implementation to show SpoVNets capability for a realistic deployment called *ariba framework* [12].

The ariba API offers two interfaces for a service or application developer. The first one is for creating, joining and leaving a specific SpoVNet-instance, while the other one defines communication between nodes in a joined instance. Currently, ariba offers an open-source implementation running on Linux and Maemo-based Nokia handhelds.

In future, the developers of ariba aim for improving the framework and extending it with further protocols and security components. Furthermore as it becomes apparent that services and applications running with the ariba framework should be evaluated with simulations before being deployed in the internet, ariba will be extended by offering a way of simulation support with the P2P simulator OverSim. Some open issues are under research like a possible partitioning of the overlay and optimisation of relay-paths. For a realistic chance of wide deployment of *ariba* it is essential that legacy applications are transparently supported which is currently examined. Last but not least the developers are working on a portation to further platforms, especially infrastructure routers in order to achieve maximum efficiency.

VIII. Conclusion

In this paper we looked at the key concepts and the components of SpoVNet and how P2P-applications can benefit from introducing an underlay abstraction as SpoVNet proposes. Two example applications that are considered to be high demanding were presented to prove SpoVNets capabilities. We discussed related approaches and compared them to SpoVNet – and at last open issues were discussed that are not suitable solved yet.

In table I are SpoVNets capabilities sumarised and compared to other approaches, mentioned in the related work section. Following attributes were considered: *cryptographic identifiers* means the seperation between real addresses and cryptographic IDs, *secure communication* means the support for establishing secured transportlinks like TLS, *protocol heterogeneity* means the capability of the given approach to handle communication between nodes with IPv4 and IPv6, *multihoming* represents the capability to handle a seamless change between different locations, *multicast* means the support of multicast-mechanisms, *private overlays* implies the possibility to create secured overlays with restricted access, *QoS / performance* means the capability of the approach to somehow meet given performance and speed-requirements. *Platform independent* implies that the approach is independent from platforms and progamming languages and last but not least the *actual implementation*-attribute shows if there is a implementation ready that can be used.

If an approach fits to one attribute it is marked with "X". There might be some restrictions, though, but it is only considered if there exist concepts anyhow.

References

[1] Matthias Altorfer, Daniel Heuberger, and Manuel Innerhofer. NSIS - Signaling in IP Networks, accessed on 10, May 2011. http://www.csg.uzh.ch/teaching/ss06/comsys/extern/talk6.pdf, 2006. III

[2] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. *Scalable Application Layer Multicast*. Proceedings of SIG-COMM 2002, 2002. IV

[3] Robert Becker, Franziska Liebsch, Yann-Rudolf Michel, and Daniel Müller. Jxta: Einführung und Überblick. http://www.ag-nbi.de/lehre/04/S_P2PNET/Ausarbeitungen/JXTA.pdf, 2004. V

[4] S. Behnel, L. Fiege, and G. Mühl. *On Quality-of-Service and Publish-Subscribe*. Processings 26th IEEE Int'l Conference Workshops on Distributed Computing Systems, 2006. IV

[5] Roland Bless, Christian Hübsch, Sebastian Mies, and Oliver Waldhorst. *The Unterlay Abstraction in the Spontaneous Virtual Networks (SpoVNet) Architecture*. Proceedings of 4th EuroNGI Conferent on Next Generation Internet Networks 2008, 2008. II, 3, 10

[6] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf. *Design and Evaluation of a Wide-area Event Notification Service*. ACM Trans. Comput. Syst. 19(3), 2001. IV

[7] Shao-Chen Chang, Tsu-Han Chen, Shun-Yun Hu, and Guan-Ming Liao. P2P-based Virtual Environment Resarch, accessed on 30, Apr 2011. http://vast.sourceforge.net/relatedwork.php, 2011. IV

[8] Yang-Hua Chu, Sanjay G. Rao, Srinivasan Seshan, and Hui Zhang. *A Case for End System Multicast*. IEEE Journal on selected areas in communications 2002, 2002. I

[9] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. *Safebook: a Privacy Preserving Online Social Network Leveraging on Real-Life Trust*. IEEE Communications Magazine, Vol 47, N.12, 2009. VI

[10] Dirk Haage, Ralph Holz, Heiko Niedermayer, and Pavel Laskov. *CLIO - a cross-layer information service for overlay network optimization*. 4th GI/ITG KuVS Workshop on The Future Internet and 2nd Workshop on Economic Traffic Management (ETF), 2009. 4, III

[11] Christian Hübsch, Christoph P. Mayer, Sebastian Mies, Roland Bless, Oliver P. Waldhorst, , and Martina Zitterbart. *Reconnecting the Internet with ariba: Self-Organizing Provisioning of End-to-End Connectivity in Heterogeneous Networks*. SIG-COMM 2009, 2009. VI

TABLE I
SPOVNET IN COMPARISON TO EXISTING APPROACHES

	HIP	i^3	Ambient Networks	JXTA	SpoVNet
IP/ID-split	X	X	X	X	X
secure communication				X	X
protocol heterogeneity			X	X	X
multihoming	X	X	X	X	X
multicast		X		X	X
private overlays				X	X
QoS / performance		X		X	X

[12] Christian Hübsch, Christoph P. Mayer, and Oliver P. Waldhorst. *The Ariba Framework for Application Development Using Service Overlays*. Praxis der Informationsverarbeitung und Kommunikation (PIK) vol. 33, 2010. VII

[13] Ralph Holz and Dirk Haage. *CLIO/UNISONO: practical distributed and overlay-wide network measurements*. Kommunikation in Verteilten Systemen (KiVS) 2009, 2009. III, III

[14] LWN.net. The last IPv4 address blocks allocated, accessed on 30, Apr 2011. http://lwn.net/Articles/425919/, 2011. I

[15] L. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. http://ietf.org/rfc/rfc4423.txt, 2011. V

[16] P. Nikander, J. Laganier, and F. Dupont. An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID). http://www.ietf.org/rfc/rfc4843.txt, 2007. 2

[17] A. Perrig, D. Song, R. Canetti, J.D. Tygar, and B. Briscoe. Timed Efficient Stream Loss-Tolerant Authentication (TESLA). http://www.ietf.org/rfc/rfc4082.txt, 2005. IV

[18] H. Schwarz, D. Marpe, and T. Wiegand. *Overview of the Scalable Video Coding Extension of the H. 264/AVC Standard*. IEEE Trans. on Circuits and Systems for Video Technology 17(9), 2007. IV

[19] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. *Internet indirection infrastructure*. IEEE/ACM Trans. Netw. Volume 12 No. 2, 2004. V

[20] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*. ACM SIG-COMM 2001, 2001. III

[21] Oliver Waldhorst, Christian Blankenhorn, Dirk Haage, Ralph Holz, Gerland G. Koch, Boris Koldehofe, Fleming Lamp, Christoph P. Mayer, and Sebastian Mies. *Spontaneous Virtual Networks: On the Road Towards the Internet's Next Generation*. it - Information Technology Special Issue on Next Generation Internet Vol. 50(6), 2008. 2, III