Evaluating Performance Characteristics of SIP over IPv6

Thomas Hoeher[1], Martin Petraschek[1], Slobodanka Tomic[1], and Michael Hirschbichler[2]
[1]ftw. (Telecommunications Research Center Vienna), Vienna, Austria
Email: {hoeher, petraschek, tomic}@ftw.at
[2]Institute of Broadband Communications, Vienna University of Technology, Vienna, Austria
Email: michael.hirschbichler@tuwien.ac.at

Abstract— **Due to the ongoing massive growth of the global Internet, the rising integration of Voice over IP (VoIP) services and the Fixed Mobile Convergence (FMC), the IPv6 protocol and the Session Initiation Protocol (SIP) are key technologies for the realization of next generation communications.**

For both topics, IPv6 and SIP, a lot of self-contained research has been done. However, the challenge of SIP over IPv6 as well as related issues and performance impacts were not considered so far. In this article, we close this gap and draw attention to theoretical and practical aspects of the integration of SIP and IPv6, referred to as SIPv6. In this context our special interest concerns the interworking of heterogeneous IP networks during the transition from IPv4 to IPv6 and their ramifications on the VoIP service. Inevitably, during this period of co-existence the available transition techniques have an impact on the network and application performance. To quantify this impact, we set up a SIPv6 VoIP testbed and measured the performance penalties introduced by four selected transition techniques. We characterize the performance of transition scenarios compared to native scenarios by presenting measurement results and gained insights. Our study reveals individual pros and cons of transition technologies and their available implementations.

Index Terms— **IPv6, SIP, performance measurements, proxying, tunneling, transition techniques, 6to4, Teredo**

I. Introduction

More than one decade ago the Internet Engineering Task Force (IETF) has started to develop a successor for the most widely deployed network layer protocol in the Internet: the Internet Protocol version 4 (IPv4).

IPv4 was originally developed in 1981 to solve the (internet) routing matters for a small backbone connecting academic and government networks within the United States. Nowadays, the Internet is a worldwide backbone interconnecting thousands of autonomous systems, however the network layer protocol is still IPv4.

As nobody could have foreseen this fulminant growth, the developers of IPv4 decided generously to serve almost four billions of nodes, i.e., half of the today's world population. Nevertheless, for some years now the Internet is on the verge of depletion of IPv4 addresses. Already in 1992 the IETF identified this upcoming exhaustion and started to specify countermeasures such as Network Address Translation (NAT), Variable Length Subnet Mask (VLSM), Classless Interdomain Routing (CIDR), and Internet Protocol version 6 (IPv6). However, the only sustainable solution to cope with IPv4 address space shortage, which does not solely shift the point of depletion is IPv6.

IPv6 was conceived within the IETF working group IPng founded in 1994 with the goal to define the next generation Internet Protocol. Among several proposed alternatives, IPv6 [1] - originally called IPng - has been selected as the successor of IPv4. Today, in a variety of IPv6-related RFCs, complementary topics like security, address and routing schemes, the IPv6 transition, and mobility enhancements are treated.

In general, IPv6 offers some novelties and benefits but still the most convincing argument for introduction is the 128-bit address space. The urgency for the introduction of IPv6 becomes more and more obvious since well known Internet engineers like Tony Hain (Cisco Inc.) and Geoff Huston (APNIC) predict the point of depletion between *autumn 2008* [2] and *summer 2012* [3]. In other words, it is high time to continuously push the global implementation of IPv6.

One of the still missing pieces in the IPv6 puzzle is the evaluation of typical deployment issues such as performance, interoperability, and scalability. In this context, the main criteria at the beginning of IPv6 introduction would be the seamless interworking between IPv4 and IPv6, also considered as *IPv6 transition*. Apparently, the migration from IPv4 to IPv6 could only happen in the course of an incremental transition where both protocols have to co-exist. The duration of this process is not predictable, however it brings up a variety of additional aspects, such as performance and scalability in particular at the application layer.

These open questions motivated our IPv6 research on the transition aspects of SIP (Session Initiation Protocol) [4], the protocol which is currently penetrating and revolutionizing the Internet and its services landscape. But not only the Internet, it is about to change the entire

This contribution is based on "Performance Evaluation of SIPv6 Transitioning," by T. Hoeher, M. Petraschek, S. Tomic and M. Hirschbichler, which appeared in the Proceedings of The Second International Multi-Conference on Computing in Global Information Technology (ICCG 2007) co-located with IPv6TD 2007: The Second International Workshop on IPv6 Today - Technology and Deployment, Guadeloupe, French Caribbean, March 2007. © 2007 IEEE.

This work was supported by the Kplus program of the Austrian Federal Government.

telecommunications industry since it has become accepted as the signaling protocol-of-choice for services in the IP Multimedia Subsystem (IMS).
SIP, also known as rendezvous protocol, was primarily designed to signal multimedia-sessions. The most prominent use case is Voice over IP (VoIP), but SIP goes far beyond call initiation including services like location, presence and messaging. The underlying reason for the success of SIP is its simplicity, openness and versatility following the basic concepts of the Internet.

IPv6 and SIP are key technologies for the realization of the Next Generation Network (NGN). Motivated by this, we performed extensive studies on the joint performance aspects to draw conclusions for the implementation of SIPv6. Our studies conducted in a dedicated SIPv6 testbed include performance evaluation for different IPv6 and SIP interworking scenarios with the focus on the VoIP-signaling capability on SIP which is described within this article as well.

This article is organized as follows. Section II briefly considers the IPv6 transition aspects. The main characteristics of SIP are reviewed in Section III. The requirements for the IPv6 transition of the SIP service (SIPv6) and the investigated strategies are presented in Section IV and Section V, respectively. In section VI we describe the design concept of our testbed architecture. Performance results are discussed in Section VII. Section VIII concludes the article.

II. IPv6

The introduction of IPv6 represents a challenge for the entire Internet industry. Several years ago hardware vendors have already started to equip their new appliances with both IPv4 and IPv6 protocol stacks. Since 2007 even the most popular operating systems are fully supporting and strongly recommending the use of IPv6. The broad support by the industry is mainly caused by the economic growth of Asia and its push to the Internet as well as the decision of DoD (Department of Defense, USA) in summer 2003 to migrate the Pentagon's network to IPv6 until 2008.

The emerging global trend towards IPv6 is lasting observably, however deployment issues especially concerning the required co-existence and interworking between IPv4 and IPv6 [5] hinders faster penetration. In this context, the buzzword *IPv6 transition* describes the incremental migration towards native IPv6. A time-frame for this process is today not foreseeable.

IETF has specified a variety of IPv6 transition technologies in order to tackle various use case scenarios in different transition stages. These technologies can be grouped into three categories [6]:

(1) *Dual-Stack* - Each IP-aware network device is equipped with a dual protocol stack, meaning IPv4- and IPv6-enabled.
(2) *Tunnelling* - To cross native IPv4-domains IPv6 traffic is encapsulated in IPv4 packets. Tunnels are established between two endpoints either manually or automatically, depending on the chosen method (e.g. 6to4, Teredo, or static tunnels).
(3) *Translation* - On the edge between an IPv4 and an IPv6 domain one protocol must be translated into the other and vice versa. For this interconnection several approaches exist including direct translation (NAT-PT) or the termination of one IP-leg and the new initiation in the other domain (proxying).

As long as the core Internet and most of the provider networks remain native IPv4-enabled the only method to interconnect two IPv6 nodes is to tunnel the traffic. Nowadays, it is probably the most popular method to interconnect IPv6 islands. Dual-stack networks enhance flexibility and reachability, however the lack of IPv4-address is still present as each node ideally should serve both protocols. In corporate networks typically more than ninety percent of the deployed network infrastructure is already equipped with dual-stack capability. The use case for translation is currently not given as there are almost no networks or endpoints which only rely on native IPv6. However, a scenario with IPv6-only nodes will probably become real caused by the continuous growth of the Internet. The number of mobile devices being permanently connected to Internet has started to grow, but the space of available IPv4-address will be soon exhausted. Of course, there are several kludges deployed to shift the point of depletion, but with the ongoing growth their management will become increasingly complex. Once again the peer-to-peer nature of the Internet is gaining more and more attention - in particular from the service providers - and this fact argues against the further use of intermediate solutions which only provide an extended IPv4 space.

III. Characteristics of SIP

The Session Initiation Protocol (SIP) has been designed to signal multimedia- and multiparty-services. Considering VoIP, SIP is basically used for the initiation, modification and termination of call sessions. In general, a session is a virtual connection between two or more endpoints used to transport real-time media. For that purpose RTP (Real-time Transport Protocol) [7], the most popular protocol for the media transportation is used.

The classical SIP architecture (as depicted in Figure 1), also known as SIP trapezoid, includes the following components:

User Agent (UA) - describes the logical function of the terminal equipment. Depending on the role during the session, active or passive, a UA acts either as client (UAC) or server (UAS), respectively.

(Out-/Inbound) Proxy Server - routes SIP signaling information (proxy server). It typically integrates also the functionalities of *Registrar* and *Redirect Server*. The registrar associates the current terminal representation (IP-address) to the user's AoR (Address-of-Record) and stores this information in the location database.

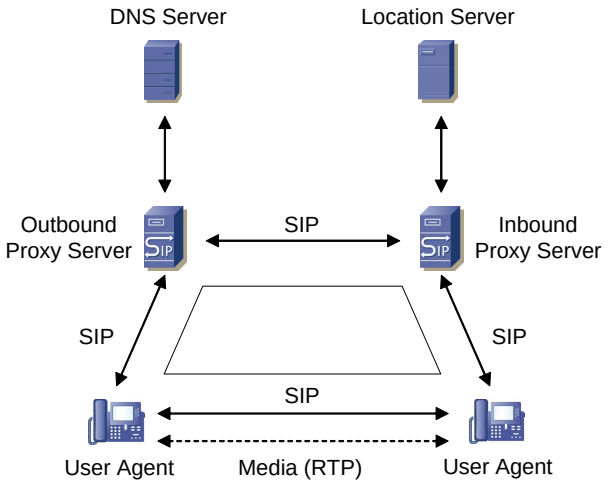

Figure 1. SIP trapezoid

Based on the registrar-function nomadicity for users, i.e., roaming can be provided. Functionally the redirect server acts as UAS which informs the calling party that the requested user is now reachable on the following SIP URI.

DNS (Domain Name Service) Server - maps domain names into IP-addresses and provides information about public domain-services. Relevant DNS-records for SIP and IPv6 are NAPTR (Network Authority Pointer), SRV (Service), A (IPv4 Address), and AAAA (IPv6 Address).

Location Server (LS) - keeps information about user location as provided during the registration process, in a database.

In order to understand SIP and its mode of operation it is necessary to consider the basic concepts of dialogue, transaction, and session.
The SIP dialogue uniquely identifies a temporary relation between two endpoints established during call setup and released after call completion. The term transaction is taken from HTTP (Hyper Text Transfer Protocol) and describes the common request/response process which can be used either inside or outside of an established SIP dialogue. Supplementary, the text-based coding as well as the header structure shows also strong similarities to HTTP. To conclude this part of definition, a session relates to transported media and its corresponding parameters. The setup of a media session happens synchronously with the initiation of the controlling SIP dialogue.

The basic functionality of SIP is built up on six methods (REGISTER, INVITE, ACK, BYE, OPTION, and CANCEL) where each of them triggers a transaction. In the course of a usual SIP call life cycle (shown in Figure 2) four of the mentioned methods play a decisive role. Once a user turns on her terminal equipment the user agent must register with its serving proxy server. For this purpose the method REGISTER is used to authenticate the user agent and to announce that the user is ready to receive calls. The public availability of a user is provided by the proxy server which is the first point-of-contact when accessing a SIP domain. During the registering procedure the physical reachability (IP-address) is stored in LS so that the proxy server is able to route calls to the user's terminal equipment.
If both caller and callee are registered, the method INVITE can be used to setup a call. The required sequence for a successful call establishment is INVITE - 200 OK - ACK. During this message exchange the media session parameters including codec, IP-address, and port number are negotiated as well. ACK is also one of the six methods however it is a unidirectional request only used for confirmation. Once ACK is received the call is set up successfully and media is transported directly between both involved user endpoints. Note that basically all SIP messages coming after session negotiation can be sent directly without passing the proxy servers. To enforce that all signaling traffic traverse the participating proxy servers the *record route* functionality of SIP can be applied. To tear down a VoIP call a transaction is released by sending the terminating BYE method. Further details about SIP can be found in [4].

Information related to the call session (like media codec or used RTP ports) is typically carried in the Session Description Protocol (SDP) [8], in the SIP payload of the session establishment message (INVITE) and the subsequent acknowledgement (ACK).

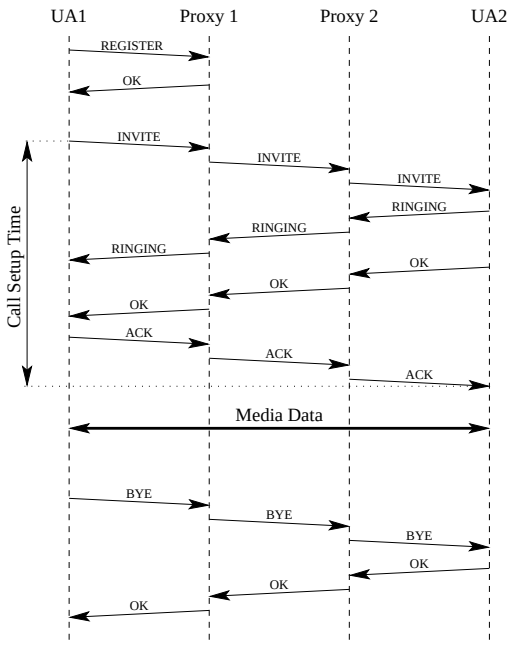

Figure 2. SIP message flow

A. Performance Metrics

For the realization of meaningful performance measurements different metrics can be considered such as CPU utilization, data throughput, and time behavior. The focus of our performance evaluation is the *time behavior*. To quantify the signaling and media transport performance we defined two characteristic values, Call Setup Time and One-Way Delay, respectively (see Figure 2).

Call Setup Time (t_{cs}) denotes the time between the leaving INVITE message (caller's side) and the arriving ACK message (callee's side). This interval is of particular importance, since it will create user annoyance and dissatisfaction if it lasts too long.

One-Way Delay (t_d) represents the average time that RTP packets require to cross the network from one endpoint to the other, meaning the direct path between UA1 and UA2.

In order to be able to conduct accurate time measurements we have synchronized the testbed on a relativ time base using Network Time Protocol (NTP). Based on this step we are able to exploit the sent/received timestamps captured with tcpdump/wireshark (www.wireshark.org). During the test runs all hosts in the testbed are capturing the entire network traffic on their Network Interface Cards (NIC). Since all hosts in the testbed are time synchronized the defined performance metrics, Call Setup Time (t_{cs}) and One-Way Delay (t_d) can be calculated by simply subtracting the sent timevalue T_{sent} from the received timevalue $T_{received}$:

$$t_{cs} = T_{ACK,received} - T_{INV,sent}$$

$$t_d = T_{UA2,received} - T_{UA1,sent}$$

$$t_d = T_{UA1,received} - T_{UA2,sent}$$

For the correlation of measurements we had to unambiguously identify packets that belong together. Meaning, a packet sent from UA1 has to be matched with the corresponding packet received at UA2. This cannot be done by comparing only the IP payload, since the content of the packet might change in transit. In SIP, proxy servers add or remove *Via* entries and decrement the *Max-Forwards* value comparable to Time-To-Live (TTL) in IP. Thus, we had to rely on specific SIP header values to match corresponding packets. In case of SIP signaling the headers Call-ID, To-tag, and From-tag are specified to uniquely identify a SIP dialogue. We used *Call-ID*, *CSeq* and *SIP Method* as these header fields provide additional information for further processing. The Call-ID field is a unique identifier that is randomly generated by the user agent for each call. CSeq (Command Sequence) header contains an integer number which is incremented for each new transaction within a call and can be considered as traditional sequence number. The SIP method denotes the purpose of a SIP transaction like INVITE and REGISTER. For the identification of corresponding RTP packets we used the unique Synchronization Source Identifier (SSRC). Additionally, the ongoing Sequence Number (SN) enables to individually detect each packet of an ongoing stream.

IV. SIP AND IPV6 TRANSITION (SIPV6)

We introduced the notion of *SIPv6* to refer to the integration of SIP and IPv6, and to cover under this term all important interworking aspects. In the following discussion we point out the individual requirements for the IPv6 transition of the SIP service related to the network, signaling, and media layer [9], which we identified as basis for our study:

Network layer: Considering the transition bottom up the IPv6 reachability is one of the fundamental needs. In other words, the SIP-enabled node must be provisioned with IPv6 connectivity either native or by one of the transition technologies. In the beginning the handling of the heterogeneous SIP overlay network could be facilitated by proxy servers with dual-stack capability.

Signaling layer: The aspects on this layer can be classified in those related to (a) SIP signaling, and to (b) DNS (Domain Name System) resolution.

(a) *SIP signaling* - As long as the caller and the callee are using the same version of the Internet Protocol the impact on the transition of SIP signaling is negligible. However, once an IPv4 user wants to call an IPv6 user the introduction of a translating instance becomes necessary. As SIP and its companion for session negotiation, SDP (Session Description Protocol) bear IP-addresses within their header structure the usage of an application-aware translator is required. To handle this issue there are two possible approaches: NAT-PT interworking with a SIP-ALG (Application Layer Gateway), and SIP proxy server acting as B2BUA (Back-To-Back User Agent).

(b) *DNS resolution* - The defined sequence for locating a SIP proxy server includes querying the DNS. At first the NAPTR-record is requested if there is currently no transport protocol (TCP/UDP/TLS) assigned. Based on this information or a predefined transport protocol the SRV-record is queried to obtain the domain-serving proxy server. Concluding, the corresponding A/AAAA-record provides the mapping of the proxy server's domain name to its IP-address. To enable SIP in IPv6 the DNS database has to be appropriately modified and extended.

Media layer: The situation on the media layer is similar to the one on the signaling layer, where the main challenges emerge for the interworking in heterogeneous scenarios - IPv4 calls IPv6 and vice versa. The media channel is negotiated during SIP signaling contained in the SDP payload. In the mentioned case of IPv4/IPv6 interworking the application-aware translator is responsible

to redirect the media channel to a translating intermediary. Such a node, often denoted as media gateway or media relay, repacks media data, mostly transported with UDP/RTP (Real-Time Transport Protocol), between two networks with different versions of the Internet Protocol. The translation process itself is straightforward if the media intermediary gets the correct information about which addresses and ports to use in IPv4 and in IPv6 from the application-aware translator. Furthermore, based on the logical and physical separation of application layer and media layer the balancing of workload among several media gateways is recommended.

V. Investigated SIPv6 Strategies

IPv6 is a mature and proven network protocol. It well considers aspects like security, mobility, routing, and addressing. However, one of the major barriers for wide deployment are the uncertainties concerning IPv6 transition and the co-existence of two Internet Protocols. Although the potential difficulties as well as the developed transition techniques are commonly understood the application-specific concerns still remain. For example, for real-time applications no meaningful performance findings are available. For this reason we decided to quantify the ramifications on VoIP caused by the deployment of IPv6 transition techniques. As there exits a variety of developed methods we selected a promising subset that covers each of the transition categories in order to obtain comparable results. This section introduces the deployed strategies and their implementation as well as discusses individual pros and cons.

1) Dual-Stack: Related to our performance evaluation the dual-stack scenarios, i.e., native IPv4 and native IPv6, provide the reference values to be compared with the other scenarios.

From the implementation point of view almost all off-the-shelf computers and operating systems support dual-stack by default. This means that in turn we can rely on hardware/software implementations which are best suited to serve as reference values. To realize SIP communication the involved dual-stack hosts were provided with an IPv4 as well as an IPv6 address and corresponding DNS entries (A-/AAAA-records).

2) Tunneling: The variety of tunneling solutions ranges from manually to automatically configured tunnels as well as tunnel brokers. For our performance study we selected two tunneling mechanisms that have complementary features concering NAT traversal:

6to4 Tunneling [10] is probably the simplest and most popular automatic tunneling mechanism to transport IPv6 packets over an IPv4 infrastructure. The discovery of 6to4 tunnel-endpoints is solved by integrating its IPv4 address within the 6to4 address (see Figure 3). If a host in the IPv6 Internet is addressed then the IPv4 anycast-address 192.88.99.0/24 is used to reach a public 6to4 relay router. A 6to4 address is identified by its unique IPv6 prefix: 2002::/16. The IPv6 packet is embedded in the IPv4 payload identified with the protocol-type 41. Summed up, 6to4 is easy and convenient to use and it

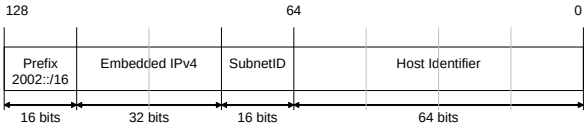

Figure 3. Structure of an 6to4 address

provides a lot of benefits, however some issues require special attention. Although the IPv4 anycast mechanism performs well, up to several seconds can elapse till a working 6to4 relay is found unless there are enough 6to4 relays deployed and they grant service to third parties. Deploying automatic tunneling mechanisms raises always the issue of *asymmetric routing*. This means that incoming and outgoing packets can travel along different paths. As a consequence, both one-way packet delay and routing capability might strongly vary depending on the direction. In addition, the capability for interdomain multicasting is still absent.

All of the above mentioned flaws can be overcome with appropriate countermeasures but 6to4 tunneling essentially fails if Network Address Translation (NAT) is in use. One exception exists, namely if the NAT device additionally implements a 6to4 tunnel-endpoint, though it is rarely used. Because of this inherent lack a fallback solution which allows to serve customers behind NAT devices needs to be implemented as well. The tunneling technique named Teredo [11] is probably the most convincing solution to cope with this issue, and is described next.

Teredo Tunneling [11], originally developed by Microsoft, is an automatic tunneling mechanism that tackles the NAT issue. In other words, by deploying Teredo the hosts which are located behind a device that does network address translation, can still be reached with IPv6. To solve the private numbering issue Teredo introduces the following architecture which includes Teredo client, Teredo server, and Teredo relay.

Teredo client is installed on a host that is placed behind a NAT, has IPv4 Internet connectivity and intends to contact an IPv6 node. *Teredo server* is a server publicly available in the IPv4 Internet. It negotiates required tunnel parameters with the Teredo client in order to overcome the NAT barrier. Typically, it works statelessly without consuming much bandwidth. *Teredo relay* represents the gateway to the IPv6 Internet. This means that a Teredo client sends its tunneled traffic to a Teredo relay which unwraps and forwards it to the addressed IPv6 node. The available bandwidth for the mentioned forwarding service towards the IPv6 Internet is directly proportional to the number of working Teredo relays. The same statement is also valid for 6to4 relays.

For the purpose of NAT traversal the Teredo client initially contacts a Teredo server to learn the deployed NAT type as well as its public representation including IP-address and port number. NAT implementations can

be divided into the full-cone, restricted-cone, port-restricted-cone, and symmetric type.
The protocol applied during the initial negotiation is a simplified form of the STUN (Simple Traversal of User Datagram Protocol (UDP) Trough Network Address Translators (NATs)) protocol. Principally, this method is based on the exchange of messages between private and public realm to detect the mapping behavior and the public representation. In case of a symmetric NAT even STUN is doomed to fail. A remedy for this situation provides the deployment of TURN (Traversal Using Relay NAT). Further considerations are out of scope of this article.
Regarding Teredo, as depicted in Figure 4, the IPv6 Teredo address includes the defined prefix 2001:0::/32, the server's and the client's public IPv4-address, the mapped port number, and flags indicating the present NAT type. Teredo is a relatively complex protocol,

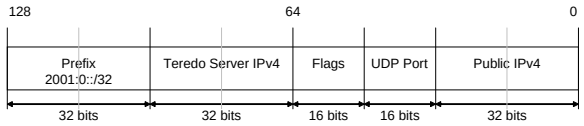

Figure 4. Structure of a Teredo address

however the big advantage of this tunneling technique is that it operates even if NATs are deployed. Still, there are additional problems to be considered. First of all each Teredo client requires a serving/dedicated Teredo server. This need sounds not so difficult to satisfy, on the other hand when configuring our testbed we figured out that it is currently hard to find a working and globally available Teredo server which is operated by a third party and publicly accessible.
An inherent weakness of the Teredo server is that it offers a single-point-of-failure as almost all traffic for the initial communication to and from the Teredo client has to traverse the same server.
During our performance measurements we also observed the occurrence of *route starvation* along the path between a native IPv6 host and a Teredo client. Route starvation means that IPv6 packets destined for a Teredo Client which uses the defined prefix 2001:0::/32 were silently discarded caused by missing links in the routing tables. Altough, based on announced routes and routing decisions the nearest Teredo relay should be contacted to deliver the tunneled IPv6 traffic, in our tests we have traced that the routing process frequently fails. One external router which discards our traffic seemed to be misconfigured.
Before being discarded the packets are routed by a number of routers which have no direct link to a router that is able to serve 2001:0::/32 (Teredo relay) and consequently use their default route. This fact leads to the conclusion that Teredo tunneling is currently faced with low penetration and deployment issues. In our case we solved the problem by installing a static route towards a Teredo relay operated by us. The experience showed that Teredo requires a given infrastructure, ideally made available by the Internet access provider, to guarantee a working tunneling service.
Finally, we have to point out that Teredo compared with 6to4 can serve only one host per negotiated Teredo address. This is caused by the individual global mapping of each host residing in the private network.
Our performance evaluation was conducted using the default 6to4 implementation provided by the Linux kernel as well as Miredo which is an open-source realization of the Teredo protocol.

3) Proxying: The third strategy for IPv6 transition deals with the translation, in particular with an approach referred to as proxying [12]. From the SIP perspective a plain straightforward translation on the network layer is not sufficient. SIP signaling even bears IP-addresses within the application layer headers which additionally require an appropriate treatment. It is conceivable to realize this task with the deployment of a stateful SIP application layer gateway (SIP-ALG). This approach is working following the find-and-replace principle to adapt SIP headers, however without knowledge about SIP logic and the corresponding state machine. The more preferable strategy is to deploy a SIP proxy server (Proxy Gateway) which acts as back-to-back user agent (B2BUA). Classically a B2BUA implements two UAs standing back-to-back where each one is responsible for its call-leg. In other words, a B2BUA crossing call is terminated on the incoming interface of the first UA and re-originated on the outgoing interface of the second UA. The required information for the call management is passed between both instances. By mapping this concept on IPv6 transition one UA is responsible for IPv4 and the other one for IPv6. In Figure 5 a typical proxying scenario is presented. The SIP Gateway is a logical entity that integrates a Proxy Gateway and a Media Gateway.
Similar to signaling, the media traffic transported in RTP must also be translated at the boundary between IPv4 and IPv6. For that purpose a so called media gateway or media relay simply performs network layer translation. Between Proxy Gateway and Media Gateway there is a communication channel, usually implemented as UDP socket with a proprietary control protocol. As the Media Gateway is basically signaling-agnostic this interface is required so that the Proxy Gateway can control the media session. In other words, the Proxy Gateway must assign addresses and ports on which the Media Gateway has to listen on and to forward RTP streams. From the deployment point of view in small networks the co-location of Proxy Gateway and Media Gateway in one network node is normally well suited. However, from a certain number of VoIP users it is recommended to distribute several media gateways (controlled by one proxy gateway) within the network in order to balance RTP work load.
At the time when we started configuring our testbed we found only two open-source proxying implementations compliant with our requirements. Because of this limited

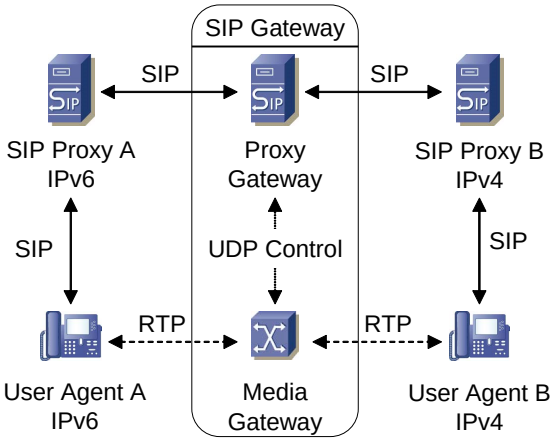

Figure 5. Typical Proxying Scenario

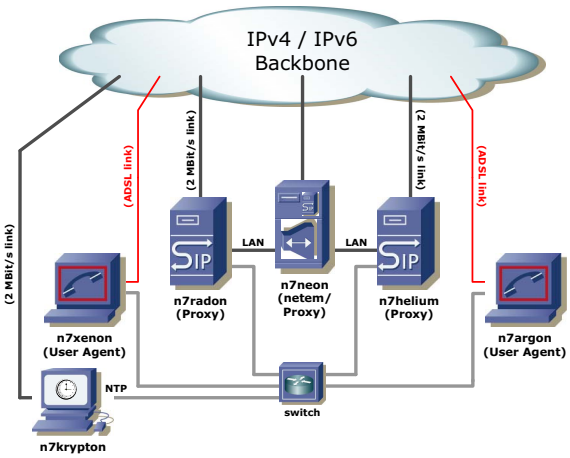

Figure 6. Universal Testbed Architecture

offer we therefore solely evaluated the following two Proxy Gateways: Mini SIP Proxy (MSP) developed by Fraunhofer FOKUS [13] and SIP Express Router (SER) with the extension module nathelper. Regarding the Media Gateway, MSP already provides an integrated SIP gateway solution including a Media Gateway named UDP forwarding daemon (ufwdd). The Portaone RTP Proxy is used to support SER for media interworking.

VI. Measurement Architecture

The focus of the performance evaluation involves native, tunneling and proxying scenarios for SIP (as discussed in [14]). In order to fulfill the different architectural requirements a universal testbed approach was developed that offers a maximum of flexibility.

In general, the testbed architecture follows a SIP architecture which reflects a realistic VoIP scenario with two involved domains. Thereby two UAs (caller and callee) and the corresponding proxy servers (Inbound and Outbound) form a so called SIP Trapezoid. In the depicted testbed infrastructure (see Figure 6) *n7argon* and *n7xenon* host the user agents and *n7radon* and *n7helium* act as proxy servers. For the automation of VoIP testing SIPp, an open-source call flow generator for SIP, acts as UA. Based on scenarios described in XML-format, SIPp establishes and releases a given number of (concurrent) sessions depending on the configured call rate (calls per second, cps). As proxy server, the open-source implementation SER (SIP Express Router) was chosen. In order to use the proxy servers as flexible as possible they are configured as dual-stack nodes serving simultaneously the IPv4 as well as the IPv6 domain. For the purpose of *SIPv6 transitioning* [9] we need one additional node (*n7neon*) which plays a multi-functional role. Depending on the scenario it is used to function as a *tunnel-endpoint* for 6to4 [10] and Teredo [11] or it implements a *translating SIP Gateway* using SER or Mini SIP Proxy (MSP) [13] for IPv4/IPv6 interworking. Furthermore, this node provides domain-specific services like DNS, DHCP, or IPv6 router advertisements.

A. IPv4/IPv6 Backbone

The performance results presented in this paper are showing measurements conducted within a closed local network. But we are currently also doing measurements directing the traffic through the network of *incumbent Telekom Austria*. To serve these both purposes we developed a very flexible backbone infrastructure. The cloud shown in Figure 6 represents a logical backbone network that can either be the local network or the infrastructure of Telekom Austria. This solution enables a quick and simple reconfiguration depending on whether the local or the real-world backbone should be used.

B. Setting up Transition Scenarios

In principle, the developed architecture allows to cover almost every combination of *IPv6 Transition techniques*. By flexibly mapping them on the SIP Trapezoid different IPv4/IPv6 interworking scenarios are viable.

The performance measurements presented in this paper are realized in the designed testbed where each scenario requires a corresponding test setup.

Native - The native scenario needs only the two UAs and their corresponding proxy servers operating uniformly with either IPv4 or IPv6.

Tunneling - For the tunneling setup the user agent *n7argon* possesses merely an IPv4 address. To interwork with native IPv6 nodes *n7argon* establishes a tunnel (6to4/Teredo) with *n7neon* which acts as terminating tunnel-endpoint. All other nodes including both proxy servers run IPv6.

Proxying - In this scenario the left domain (*n7xenon and n7radon*) deploys native IPv4 and the right one (*n7helium and n7argon*) runs only IPv6. The proxying setup introduces an interconnecting proxy gateway that translates signaling information and a media gateway responsible for media translation. Both functionalities are implemented on *n7neon*.

C. Synchronization Aspects

Generally, distributed testbed architecture introduces well known challenges including one-way measurements,

clock synchronization, and accuracy/deviation. One approach for synchronizing a network is using NTP (Network Time Protocol). The protocol is based on a hierarchical structure starting from an atomic clock as top-level UTC (Universal Time Coordinated) reference and applying a server/client model to propagate time information. Additionally, it is also possible to deploy external time triggers, e.g. GPS (Global Positioning System) or DCF77.

For the purpose of our performance evaluation a slightly different requirement is given. It is not mandatory to share UTC time within the testbed but to have accurately synchronized nodes with a common (relative) time base (accuracy $\leq 10\mu$s). To satisfy this need, the testbed architecture is supported by a shadow network which is only used to transport NTP traffic.
In this shadow network *n7krypton* acts as the reference time server which synchronizes within a poll interval between 16 and 64 seconds with the time reference of the University of Vienna. This frequent synchronization reduces the drift of the local clock. The remaining nodes are synchronizing with *n7krypton* using a basic interval of 16 seconds however supported by *burst*; once synchronization is scheduled the clock is serially adjusted eight times at an interval of 2 seconds.

On the basis of the mentioned measures we achieved an *accuracy of* $\pm$ *25*μ*s* which offers a sufficient resolution for the performance evaluation presented in this paper.

VII. Performance Results

This section presents results of the performance measurements conducted using the LAN setup. The evaluation of signaling delay compares t_{cs} (Call Setup Time, see Figure 2) between

- ***Native IPv4 - Native IPv6***
- ***6to4 Tunneling - Teredo***
- ***SER - MSP***

To generate a realistic workload 10 cps (calls per second) are established and each call conveys a 12 seconds audio-file which is mirrored at the callee's side. One test cycle per scenario comprises overall 100 calls, this means that after 10 seconds 100 concurrent calls are set up. This peak lasts 2 seconds and consumes a maximum bandwidth of about 9 - 12 MBit/s depending on the chosen transition scenario (neglecting signaling traffic). The reasons for the varying bandwidth are caused by different IP-header lengths and the tunnel overhead for 6to4 and Teredo.

The first part of the performance evaluation deals with the *Call Setup Time* comparing native, tunneling and proxying. The subsequent section considers media transport and concludes the performance results showing the *One-Way Delay* of RTP packets.

A. Signaling delay

As depicted in Figure 2, t_{cs} is the time it takes to establish a SIP call in the 3-way-handshake (INVITE-OK-ACK). The results in this section represent a set of 100 calls for each scenario. Figures 7 - 12 sum up the values of t_{cs} in form of a *histogram* supported with a *quantile*. The quantile indicates the percentage of values within a given Call Setup Time. The evaluation is further done by comparing t_{cs} at 80 percent of the quantile.

The comparison between both native scenarios (IPv4 and IPv6) reveals an explicit advantage of IPv4, especially obvious if considering the quantile at 80 percent. IPv4 hits this level at about 4ms where IPv6 needs almost 5ms, i.e., a 25 percent longer delay (compare Figure 7 and 8). This result could be explained with the 128-bit IPv6 address compared with 32-bit IPv4 address. In general, there is no deducible difference between the implemented IPv4- and IPv6-stacks.
Comparing both Tunneling mechanisms, 6to4 has an obvious advantage. The difference is about 1ms if following the quantiles (see Figure 9 and 10). The decisive reason might be the implementation, as 6to4 runs in kernel-space and Teredo is an open source user-space implementation. Furthermore, Teredo utilizes a mechanism to clarify in advance if a Teredo address is currently assigned which also consumes time. At this point it must be mentioned that using tunneling mechanisms requires preconditions which guarantee that all tunneling components are accessible. This issue especially addresses the availability of relays which provide the connectivity to the IPv6 Internet.
Prior to starting discussion the proxying scenarios, it has to be pointed out that for SER and MSP we use a different scaling of the X-axis, Figure 11 and 12. This is required as especially MSP exceeds the determined scale. It must be stressed that MSP is only a proof-of-concept implementation conducted in the 6NET project and not designed for high performance. However, very popular and widely accepted SIP Express Router causes a four times longer delay compared to the scenarios considered previously. The larger Call Setup Time can be easily explained as this approach introduces an additional proxy server for IPv4/IPv6 interconnection. This proxy must translate SIP (and SDP) between IPv4-/IPv6-domain and has also to control a media relay.
Generally, SERs 80-percent-delay of 20ms introduced in a real inter-domain VoIP scenario represents an acceptable delay in terms of the *150ms time budget* that defines a soft upper limit for adequate voice quality between User Agents. Concerning MSP, further tests were conducted with smaller workload and once the call rate falls below 5cps a compact but five times longer delay than that of SER can be observed.

B. Media data delay

This subsection compares the six scenarios in terms of *one-way packet delay*. Each of the above mentioned call scenarios signals a SIP session in order to convey an audio-file that last 12 seconds. The audio is coded using G.711[1] at a packetization interval of 20 ms which results in a packet size (RTP payload) of 160 Bytes. Per second 50 RTP packets are sent, this yields 600 packets

[1]G.711 uses a rate of 8000 samples/per second and 8 bits per sample

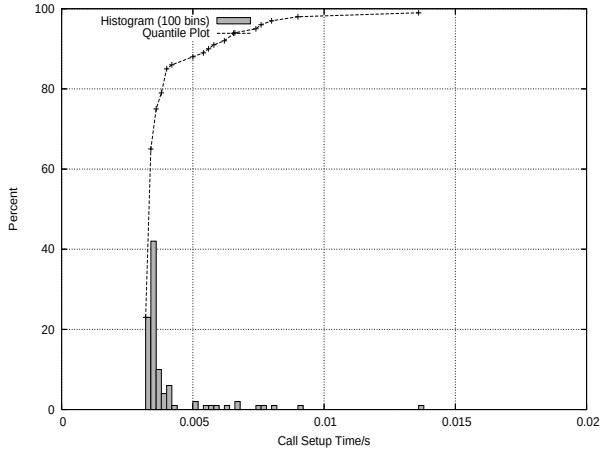

Figure 7. Native IPv4

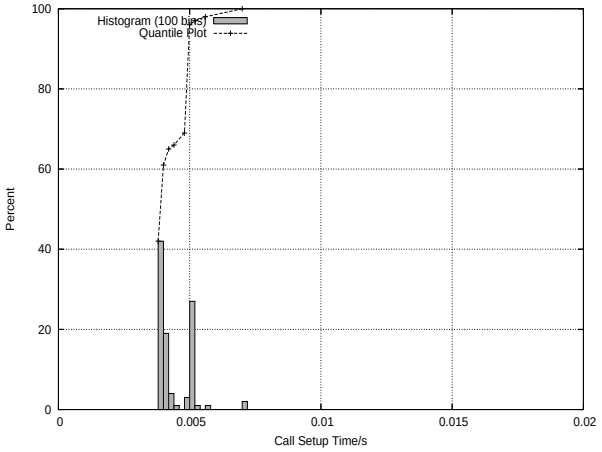

Figure 8. Native IPv6

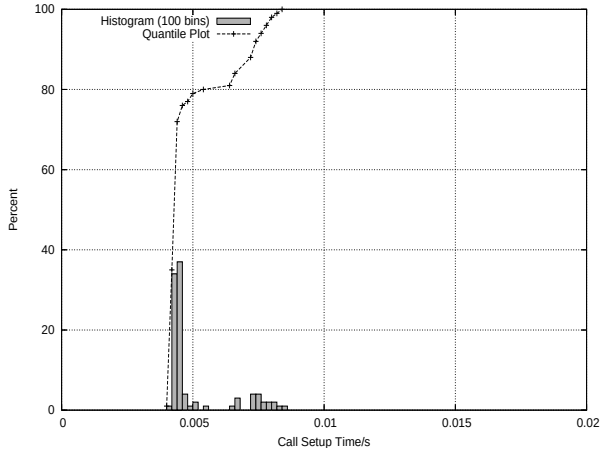

Figure 9. 6to4 Tunneling

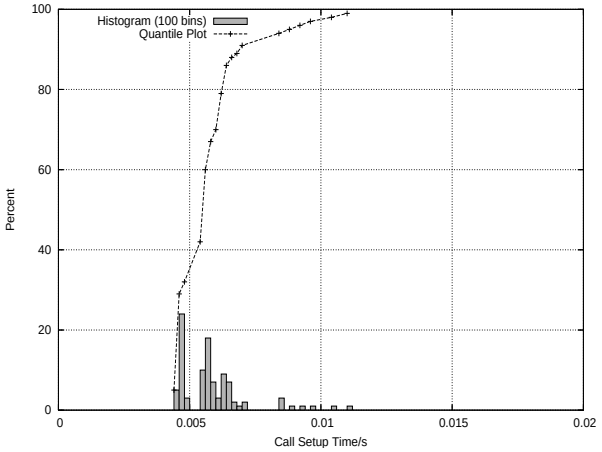

Figure 10. Teredo

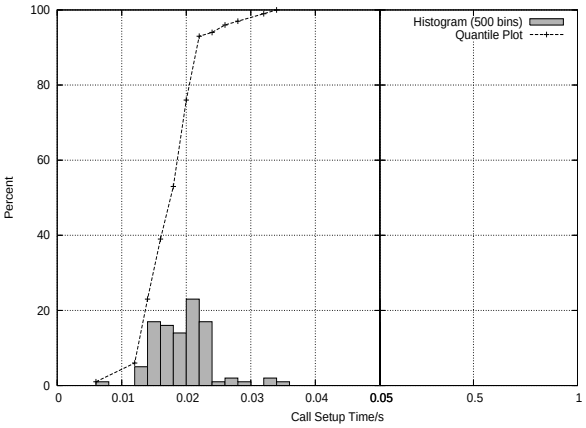

Figure 11. SIP Express Router (SER)

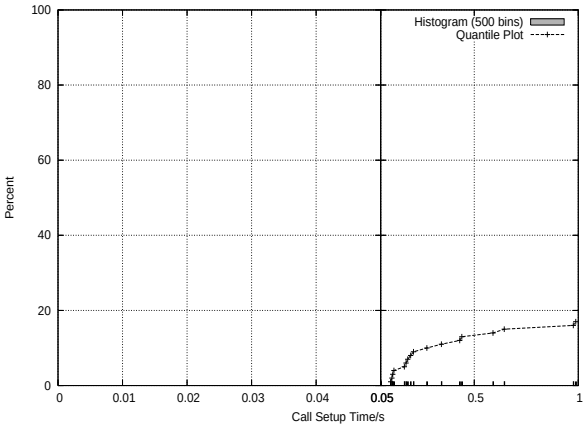

Figure 12. Mini SIP Proxy

in 12 seconds. A VoIP call establishes two audio channels (outgoing/incoming) and so the overall number of audio-packets per call is 1200. Within one scenario 100 calls are established (with a rate of 10cps). The gained set of 12000 RTP packets is used to derive meaningful statistical values.

Table I represents arithmetic mean, median and standard deviation of each scenario. Arithmetic mean and median are very similar whereas median is more robust against outliers. The standard deviation defines the deviation (root-mean-square) of the values from their arithmetic mean. Native IPv4 and native IPv6 behave generally

TABLE I.
COMPARING ONE-WAY DELAY

Scenario	Mean	Median	Deviation
Native IPv4	0.188ms	0.132ms	0.146ms
Native IPv6	0.190ms	0.139ms	0.230ms
6to4 Tunneling	0.200ms	0.180ms	0.292ms
Teredo	0.964ms	0.742ms	0.818ms
RTP Proxy	10.017ms	10.660ms	6.130ms
ufwdd	0.201ms	0.169ms	0.158ms

almost identically, however IPv6 has an indiscernible longer delay. This could be argued with the larger IP-header but the difference is so small that it is within uncertainty of measurement.

Between both tunneling mechanisms an evident difference is obviously present. A factor of about 4.5 represents unambiguously the weakness between a kernel- and a user-space implementation.

Once again it must to be pointed out that for the one-way delay evaluation only media traffic was considered, thus the proxying solutions are reduced on the entity which does media translation. The Portaone RTP Proxy (SER) shows an unexpected slow packet processing compared with the other scenarios, but it must be stressed that no packet loss occurs. On the other hand, ufwdd (MSP) introduces only a small translation delay but with higher workload (as gradually generated during a test case) an increasing packet loss was observable. This effect cannot clearly be reasoned but it is imaginable that too short buffer sizes are applied. However, the throughput performance of ufwdd is impressive and almost comparable with native or 6to4 scenarios.

VIII. CONCLUSIONS

IPv6 transition in general is well covered and specified but there are still application-specific aspects which have to be considered. In terms of SIP for instance, transition can be best handled with the Proxying solution which is an enhanced application-specific approach. In the context of VoIP we are faced with strict limitations concerning delay, jitter and packet loss and so we must stress the need for performance characterization and evaluation for IPv6 transition. This is also the main motivation for the experimental approach we represented in this paper.

ACKNOWLEDGMENT

We thank the members of the ftw. N7-project, *KapschCarrierCom, Siemens, Telekom Austria and Vienna University of Technology* for their vital support and the useful collaboration in order to write this journal contribution.

REFERENCES

[1] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 1883, Dec. 1995.
[2] T. Hain, "A Pragmatic Report on IPv4 Address Space Consumption," *The Internet Protocol Journal*, vol. 8, no. 3, September 2005.
[3] G. Huston, "IPv4 Address Report," http://www.potaroo.net/tools/ipv4/, Tech. Rep., October 2006.
[4] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, June 2002.
[5] M. Tatipamula, P. Grossetete, and H. Esaki, "IPv6 integration and coexistence strategies for next-generation networks," *IEEE Communications Magazine*, vol. 42, no. 1, pp. 88–96, January 2004.
[6] J. Wiljakka, "Analysis on IPv6 Transition in Third Generation Partnership Project (3GPP) Networks," RFC 4215, Oct. 2005.
[7] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 3550, July 2003.
[8] M. Handley and V. Jacobson, "SDP: Session Description Protocol," RFC 2327, Apr. 1998.
[9] G. Camarillo, K. E. Malki, and V. Gurbani, "IPv6 Transition in the Session Initiation Protocol (SIP)," IETF," Internet-Draft, February 2006.
[10] B. Carpenter and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds," RFC 3056, Feb. 2001.
[11] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," RFC 4380, Feb. 2006.
[12] FhG Fokus, "Report on Integration of SIP and IPv6," 6NET, Tech. Rep., August 2003.
[13] P. O'Hanlon, S. Varakliotis, R. Ruppelt, and J. Fiedler, "Realisation of IPv4/IPv6 VoIP Integration Scenarios," 6NET, Tech. Rep., January 2005.
[14] T. Hoeher, S. Tomic, and R. Mendetter, "SIP collides with IPv6," IEEE, Tech. Rep., July 2006.

Thomas Hoeher received his master degree in Electronics from the University of Applied Sciences Technikum Wien, Austria, in 2003. He is currently finishing a master program in Telecommunications and Internet Technologies.

Since 2002 he is working as researcher at ftw. (Telecommunications Research Center Vienna), Vienna, Austria. His research interests include VoIP/IMS security, IPv6 deployment and location-based services.

Martin Petraschek studied Telecommunications Engineering at the University of Technology of Vienna and received his master degree from the Institute of Computer Technology in 2002. His diploma thesis was about building up a public key infrastructure (PKI) for mobile devices. After working for a satellite Internet provider in Barcelona/Spain he joined ftw. in 2004.

He is currently working on various security projects, dealing with Voice-over-IP security, intrusion detection and prevention systems (IDS/IPS), and IPv6 security.

Slobodanka Tomic graduated from the Faculty of Electrical Engineering in Belgrade, Yugoslavia. In November 2000 she joined the Institute of Broadband Communications at Vienna University of Technology in Austria as a researcher working towards a Ph.D. Prior to that she was working in the industry where she was engaged in several research and development projects focusing on network control and management. Since September 2006, Ms. Tomic is with the Telecommunications Research Center Vienna (ftw.), Austria, where she works as a senior researcher and project manager.

Her research interests include architecture and protocols for new generation ubiquitous networks and services.

Michael Hirschbichler is university assistant and Ph.D. candidate at the Vienna University of Technology, Austria. He received his master degrees in Computer Sciences and Computer Science Management in 2006 and 2007, respectively.

His research interests include the IP Multimedia Subsystem (IMS), Spam over Internet Telephony (SPIT), and Voice over IP(VoIP) security and performance.