Modeling and performance evaluation of transport protocols for firewall control

Sebastian Kiesel * Michael Scharf

Institute of Communication Networks and Computer Engineering
University of Stuttgart, Germany
{sebastian.kiesel,michael.scharf}@ikr.uni-stuttgart.de

Abstract

Firewalls are a crucial building block for securing IP networks. The usage of out-of-band signaling protocols such as SIP for IP telephony and multimedia applications requires a dynamic control of these firewalls and imposes several challenges. Recently, several firewall control architectures and protocols have been developed. The main focus of this paper is the Simple Middlebox Configuration Protocol (SIMCO), which is a new transaction-based firewall control protocol. Due to the impact on call setup delays, firewall signaling requires small end-to-end delays and thus mandates a careful choice of the transport protocol. Therefore, this paper studies SCTP, TCP and UDP-based transport for SIMCO and compares different configurations that allow to optimize the performance. We present an analytical model to quantify the impact of head-of-line blocking in SCTP and TCP and verify it with measurements. Both the model and measurements reveal that SCTP can significantly reduce the SIMCO response times by leveraging transmission over multiple parallel streams. While already a few SCTP streams can almost completely avoid head-of-line blocking, our results show that TCP- and UDP-based transport may suffer from significantly larger delays.

Key words: Firewall, IETF MIDCOM, IETF NSIS, SCTP, Head-of-line blocking

*Corresponding author

Computer Networks – The International Journal of Computer and Telecommunications Networking,
Volume 51, Issue 11, 8 August 2007, Pages 3232-3251.
doi:10.1016/j.comnet.2006.11.031 http://www.elsevier.com/locate/comnet

1 Introduction

Firewalls are a widely deployed technology to protect networks against unwanted access. The usage of out-of-band signaling in IP networks, namely "Voice over IP" (VoIP) solutions using the Session Initiation Protocol (SIP), poses new challenges to firewalls. Due to the dynamic nature of SIP, firewalls have to take part in the session signaling. The advancement of firewall technology is further driven by the emerging IP telephony platforms, which are also referred to as Next Generation Networks (NGN). These network architectures, such as 3GPP IMS or ETSI TISPAN, are intended to replace the circuit-switched telephone networks by SIP-based VoIP. However, compared to the Internet, these networks have much higher security requirements. Therefore, firewalls will be a major component in these platforms, e. g., for screening at the interconnection points of different operator's networks.

Several options exist for the signaling to such firewalls, such as NSIS or the Simple Middlebox Configuration Protocol (SIMCO), and there are many ongoing research and standardization activities in this field. Thus, the first part of this paper reviews different firewall control architectures and discusses their interaction with SIP-based applications.

Like most signaling applications, firewall control protocols have quite stringent delay requirements because the transaction delay contributes to the call setup delays perceived by users. Therefore, the choice and parametrization of transport layer protocols is of particular importance. The Transmission Control Protocol (TCP) is the default choice for reliable transport in the Internet. Since TCP ensures reliable in-order delivery, end-to-end delays may be increased due to the head-of-line blocking effect when IP packets are lost. This effect is particularly critical on links with high data rates, i. e., between large softswitches and other central IP telephony platform entities such as firewalls.

While head-of-line blocking is a well-known problem of TCP, there are only few studies that quantify the impact of this effect on end-to-end delays. Potential alternatives to TCP for improving delays are multiple parallel TCP connections, multiple SCTP streams, SCTP unordered mode, and UDP-based transport. In the second part of this paper, we study the response time of transaction-based firewall signaling protocols over TCP and SCTP, both by analytical models and by measurements on different operating systems. This part extends work that has been published earlier in [1].

The remainder of this paper is organized as follows: In Section 2, we introduce fundamental firewall concepts and discuss the interaction of SIP signaling and firewalls. In Section 3 different architectures for firewall control are reviewed. Also, SIMCO as one promising signaling protocol for firewall control is presented. Sections 4 and 5 discuss the suitability of TCP, UDP and SCTP for signaling transport. At the example of SIMCO, we compare different approaches to reduce head-of-line blocking. Section 6 presents analytical models to quantify its impact. In Section 7, we present performance measurement results that have been obtained with a prototype implementation of "SIMCO over SCTP", and we compare the measurement results to our analytical models. Finally, Section 8 concludes this paper.

2 Securing IP Telephony Networks by Firewalls

2.1 Firewalls in the Internet

With respect to computer networks, the term "firewall" is used to describe one or a group of network elements that enforce an access control policy on the traffic at the border between network domains with different security levels and requirements. That is, a firewall is basically a gateway that relays traffic from one domain to the other, but only if the traffic is compliant to a specific security policy. In the context of the Internet, firewalls are often used by organizations to protect computers *inside* a private network from unwanted access from the *outside* Internet [2].

To some extent, the concept of firewalls is contradictory to one of the design principles of the Internet, the end-to-end argument. However, firewalls are used because performing access control on incoming data flows at the receiving end system is often not sufficient: First, depending on the size of the local network it might be cumbersome to establish and maintain a consistent access control policy on all systems in the organization. Second, access control mechanisms on the end system might get compromised or disabled by uncooperative users, trojan horses, viruses, etc. Having a second line of defense in front of the end system might be a reasonable choice. While these two arguments are more of practical nature, the third one is also of architectural importance: An end system cannot defend itself against Denial-of-Service (DoS) attacks that try to disrupt the victim's connectivity by flooding its access link with useless data packets. Once the packet flood has traversed the congested access link, it is too late to filter these packets. Therefore, protection against this type of attacks has to be done before the bottleneck link, which is usually the link from the Internet service provider (ISP) to the local network, i. e., the firewall has to be placed at the edge of the core network.

Numerous scientific contributions deal with DoS attack prevention in the Internet. Usually, their focus is limited to protecting hosts that communicate only with a small number of known other hosts, or they propose fundamental changes to the network and protocol architecture, such as adding signaling protocols or having stateful core network elements. For example, [3] gives an overview over several such approaches and proposes a new one, using SIP-based signaling to inform firewalls in the network about new flows. It may be considered questionable whether such extensive changes to the "stateless core"-paradigm will be implemented in the public Internet in the near-term future.

2.2 Firewalls in IP Telephony Platforms

IP telephony platform architectures such as 3GPP's IP Multimedia Subsystem IMS [4], or ETSI TISPAN [5] will use IP and IP-based protocols such as SIP and RTP. Nevertheless, their design philosophy, network architecture, and security assumptions are coined by the classical public switched telephone networks (PSTN), which differ fundamentally from the Internet [6]. These networks are under full operator control and deliver well-defined services, using a highly stateful application-layer control plane (Fig. 1). As these services are usually being charged for, security requirements, especially with respect to authentication, authorization and accounting (AAA), are much higher than in the Internet. Several architectural issues of these IP telephony platforms, in particular with respect to interdomain AAA, are still work in progress at various specification bodies such as 3GPP, ETSI, ITU-T, and IETF, see, e. g., [7] and [8]. Yet, due to enhanced

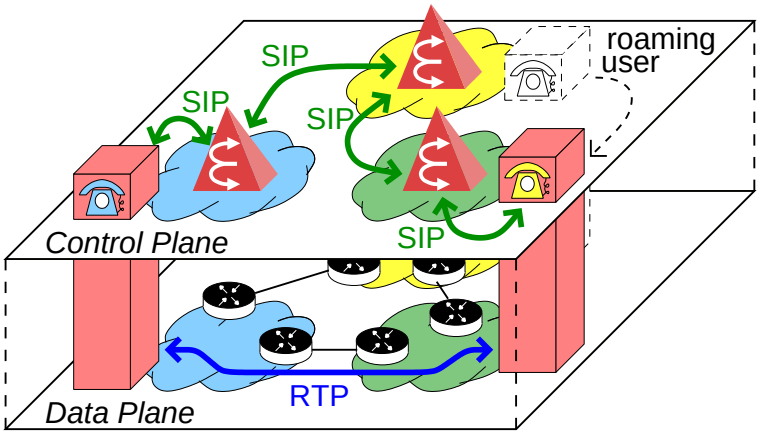

Figure 1: SIP signaling messages and RTP media streams may travel on different paths through the network

security requirements, firewalls will be a crucial building block in order to tighten the network interconnection interfaces, both to the subscribers and to other operator's networks.

One main task of the control plane is to establish and tear down phone calls. SIP back-to-back user agents (B2BUA), softswitches, or similar stateful protocol entities for SIP and other signaling protocols can be used to implement the necessary functions, which are in IMS context referred to as "Call Session Control Function" (CSCF). Call setup includes various policy decisions, considering charging agreements between operators, protection against unwanted calls such as "spam over IP telephony" (SPIT), roaming users, etc. Once the control plane (and the called party, of course) have accepted a specific phone call, the firewalls have to be informed that the corresponding media streams (i. e., speech) may traverse the domain boundaries. This is the task of firewall control signaling, on which we will focus in the remainder of this paper.

2.3 Firewall Policies

The access control decision and the forwarding of traffic can be performed on different layers of the protocol stack. Firewalls for IP networks are implemented most often in the IP layer (so-called "packet filters") or in the application layer ("proxies"). It is also possible to use a combination of both.

A central concept of policy based systems such as firewalls is the *policy rule*, which is "the binding of a set of actions to a set of conditions" [9]. For packet filters, conditions are expressed as 5-tuples (source IP address, destination IP address, transport layer protocol ID, source port number, destination port number) that describe the packet flows to which a specific treatment (action) is applied, e. g., forwarding towards the destination, discarding with or without notifying the sender, or writing an entry to a log file. A policy rule that allows specific packets to pass a packet filter is also called a *pinhole*.

When configuring a firewall system, the system administrator usually starts by defining a high-level policy with respect to services. An important issue is how to convert this high-level policy into an access control list that can be understood by the network elements that build up the firewall and which have to enforce the policy. Current practice for packet filters in the Internet

is to derive a *static* list of pinholes that consider the transport layer port numbers. This is possible due to the in-band signaling and the "well known port numbers"-concept most traditional Internet services such as WWW or e-mail follow.

2.4 Out-of-band Signaling in IP Networks

However, some applications such as many VoIP solutions differ from these concepts described above by using different protocols for signaling and transport of the actual user data (speech). In the considered IP telephony scenarios, the RTP (Real-time Transport Protocol) stream parameters, e. g., codec, bit rate, and their IP addresses and UDP port numbers are *dynamically* negotiated using SDP (Session Description Protocol) messages embedded in the SIP signaling.

The most basic signaling flow is shown in Fig. 2. By means of an SDP attachment in the *INVITE* message, the calling party *A* announces on which address it is going to listen for the RTP stream from *B* to *A*. The called party *B* announces the parameters for the media stream in the opposite direction using the SDP data contained in the *200 OK* message.

Due to this dynamic nature of SIP/SDP and RTP, a static configuration of packet filters on the media path is no real option, as it would either block all RTP streams or would have to allow all UDP traffic, rendering the firewall's protection almost useless in many network scenarios [10]. Instead, firewalls have to interact with the session signaling, which will be considered in detail in the following sections. However, as shown in Fig. 1, this is complicated by the fact that signaling messages and media streams may travel on different paths through the network (e. g., to roaming users).

Fig. 2 also shows that the *200 OK* message with *B*'s SDP is sent as late as when the callee has already accepted the call, i. e., picked up the receiver. Therefore, processing of this information in intermediate systems, e. g., for the dynamic configuration of firewalls, must be performed quickly, in order to keep the answer signal delay [11] as low as possible. Otherwise, the first words of the conversation could get lost, causing inconvenience to the subscribers. There exist extensions to the basic SIP call flow (e. g., [12]) that allow negotiation of all RTP parameters before B's phone starts ringing. Nevertheless, firewall configuration should be fast, as in this case it contributes to the post-selection delay [11], which is unpleasant, too.

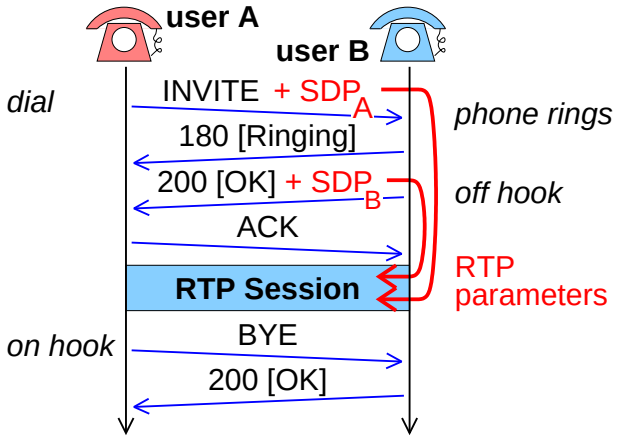

Figure 2: Negotiation of RTP parameters by means of SIP/SDP

It should be mentioned that the calling party must be prepared to receive so-called *early media* as soon as it has sent the *INVITE* message. This may be used for announcements such as the credits left on a prepaid calling card, or in-band ring tones and error messages generated by local exchanges in an analog telephone network interconnected by a gateway. If firewalls are not configured timely, these announcements cannot be delivered to the caller.

2.5 Middleboxes

Firewalls are not the only type of devices that can be found in the middle of an IP network in addition to routers and entities of lower protocol layers. In [13] the term *middlebox* is defined as "any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host." This rather broad definition also covers application layer entities such as HTTP proxies.

In the IETF, the MIDCOM and NSIS working groups are specifying architectures for the interaction of applications and hosts with one sub-class of middleboxes that do not have protocol entities for the application layer protocols the users are interested in. Instead, these middleboxes process individual packets or packet flows. Typical applications are access control (i. e., packet filters) and network address translation (NAT), but also, e. g., network intrusion detection. The MIDCOM and NSIS architectures will be introduced in Section 3.

2.6 Solution Approaches for the SIP/RTP Firewall Traversal Problem

There are many approaches to the problems VoIP has with firewalls. Simple ideas, such as statically configuring packet filters to allow any UDP packets, allow only coarse-grained access control and are not secure enough for many usage scenarios.

Some application developers and users try to bypass firewalls by using well known port numbers of popular protocols (e. g. 80/TCP of HTTP, as some versions of Skype did [14]), or tunneling over allowed protocols such as HTTP. This may be seen as violation of the firewall administrator's policies and it may expose the system again to the risks that lead to the deployment of firewalls in the first place.

Another class of solution attempts, which will not be considered in detail here, is to replace SIP/RTP by one protocol that uses in-band signaling and one well-known port number, e. g., IAX2 [15].

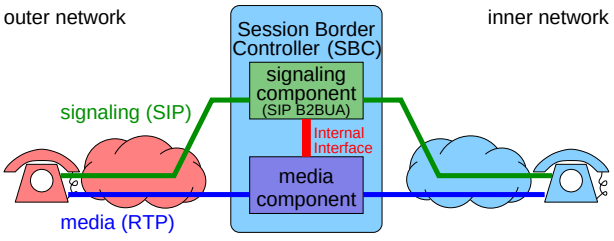

Figure 3: Session Border Controller architecture (acc. to [16])

More elaborate and sound firewall solutions for SIP/RTP have to treat both signaling and media. This can be done within one physical network element, e. g., by adding a SIP parser to the packet filter code, such as the "SIP conntrack helper" introduced in version 2.6.18 of the Linux operating system, in order to determine the addresses used for RTP. The more sophisticated so-called "Session Border Controllers" (SBC) usually consist of a SIP B2BUA and an RTP proxy (see Fig. 3). SBCs may have many additional, partly vendor-specific, and mostly security-related functions such as network topology hiding by stripping address fields from SIP messages, RTP media transcoding or encryption [16]. With respect to access control for RTP, they can be considered as "in a box" version of the MIDCOM architecture (see next section).

However, such "in a box" solutions require or enforce that signaling and media flows have to traverse the same border elements. They require processing of SIP messages and keeping call state not only in central B2BUAs or softswitches, but also at every network border, which significantly increases the total processing effort. Furthermore, this approach inhibits media streams taking an optimized path with potentially smaller end-to-end delays, as shown in Fig. 1.

3 Architectures for Firewall Control

The solutions to the firewall traversal problem considered in the remainder of this paper are based on dynamically adding rules to the packet filter (middlebox) by means of a signaling protocol. This can be done in two ways: *path-coupled* firewall signaling and *path-decoupled* firewall signaling. Both approaches, their advantages and drawbacks, and the current standardization efforts at the IETF will be introduced in the next subsections.

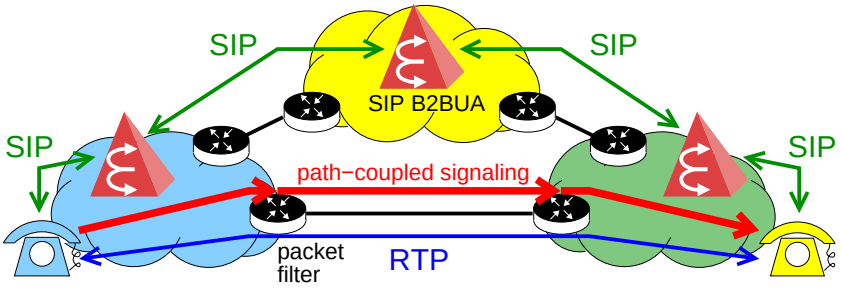

Figure 4: Path-Coupled Firewall Signaling: signaling messages are sent along sections of the (future) media path – not necessarily end-to-end

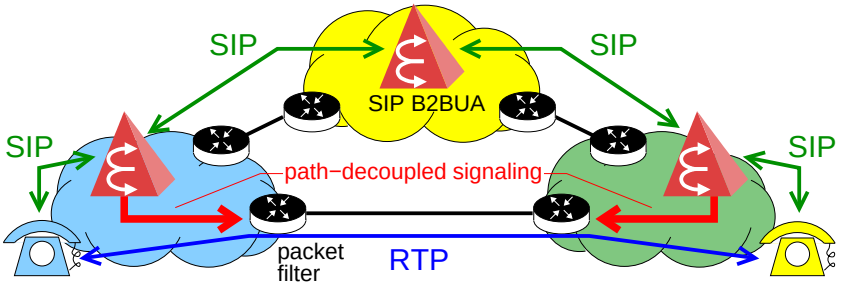

Figure 5: Path-Decoupled Firewall Signaling: middleboxes are controlled from centralized, application-layer stateful instances such as SIP back-to-back user agents (B2BUA)

3.1 Path-Coupled Signaling – IETF NSIS

The idea behind path-coupled signaling is to send signaling messages along the media path, which announce a new flow and create and maintain the necessary state information at intermediate systems on the path (see Fig. 4). This is not limited to middlebox traversal such as NAT and firewall signaling. Indeed, this concept has been proposed and implemented also for the signaling of QoS reservations, e. g., by the Resource Reservation Protocol (RSVP). There were also proposals to use RSVP for path-coupled firewall control [17].

The IETF Next Steps In Signaling (NSIS) working group is currently working on the specification of an architecture [18] and a set of protocols that follow the path-coupled approach. Although the main focus is QoS signaling, NSIS has a much more flexible design than RSVP, therefore allowing other signaling applications as well [19]. It is possible to use NSIS signaling along only parts of the data path, i. e., it does not have to be deployed end-to-end as RSVP has to. Security issues and support for mobile devices were considered from the beginning of protocol design, whereas support for IP multicast has been omitted, in order to reduce complexity.

The basis for the NSIS architecture is a common messaging layer, the General Internet Signaling Transport (GIST) [20] on top of the transport layer (TCP, UDP, or SCTP), which can be protected by IPsec or TLS. GIST's services are used by several application specific NSIS Signaling Layer Protocols (NSLP), such as the QoS NSLP or the NAT/Firewall NSLP [21].

One key challenge in firewall control is authorization. Firewalls would be completely useless if anyone, including attackers, could request to open pinholes for new, possibly malicious flows. Therefore, path-coupled firewall signaling requests have to be authenticated and authorized thoroughly. Furthermore, when considering IP telephony platforms, the firewall signaling has to be linked with the SIP signaling, i. e., pinholes for media streams should be only allowed if a corresponding call state has been established in the B2BUAs.

One approach for this integration is described in [22] at the example of RSVP-based QoS reservations. A cryptographic capability token is generated by a policy server and forwarded via the B2BUA and the already authenticated and authorized SIP dialog to the VoIP client, which

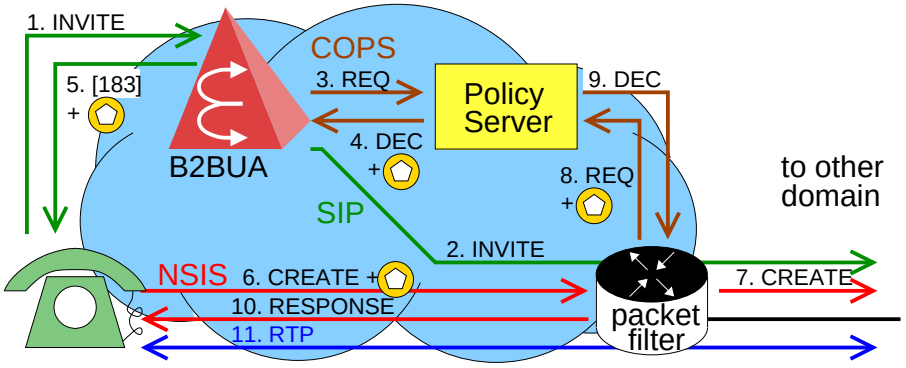

Figure 6: Linking path-coupled (NSIS/NATFW) firewall signaling to SIP session signaling by means of capability token.

For simplicity, large parts of the SIP dialog and other acknowledgements were omitted in this diagram. It is assumed that the SIP session is already authorized by some means, which may require further handshakes. Additional tokens may be required for signaling to firewalls in other domains.

can embed the token into the RSVP messages. Middleboxes can ask the policy server to verify the token, thus authorizing the path-coupled signaling request. A specification how to use this with NSIS is currently being worked on [23]; a prototype implementation [24] demonstrates the feasibility (see Fig. 6).

However, this solution implies a rather large number of handshakes involving various protocols as well as cryptographic operations, which can cause rather high call setup latencies. Furthermore, it relies on client support for NSIS and for copying the authorization token from the SIP to the NSIS message, which is so far not a widely adopted procedure.

3.2 Path-Decoupled Signaling – IETF MIDCOM

In contrast, when using *path-decoupled* signaling, firewalls on the media path are directly controlled from call stateful SIP entities such as B2BUAs or softswitches in the middle of an operator's network (see Fig. 5). As they have to process the SIP signaling anyway, the information about the pinholes for the media streams can be derived there. The IETF MIDCOM (MIDdlebox COMmunication) working group has specified a framework architecture [25] for signaling policy rule requests to the middleboxes.

Fig. 7 shows one possible MIDCOM application scenario with SIP. The firewall consists of the packet filter and the B2BUA. A static rule in the packet filter allows SIP signaling to be sent via the B2BUA. Once the B2BUA has decided to allow the establishment of a specific call, it extracts the RTP parameters from the SIP/SDP messages. It sends Policy Enable Rule (PER) requests to the the packet filter in order to open two corresponding pinholes (one per direction) for the duration of the call (Fig. 8).

Compared to path-coupled signaling architectures, fewer protocols and entities are involved in the firewall signaling. The VoIP application software in the user's phones does not have to support a special firewall signaling protocol. There is a clear trust relationship between the rather simple middlebox and its controlling SIP server that performs high-level access control decisions on a per-call basis. As the signaling between these two entities is within the trusted network of one operator, it can be protected easily, e. g., by means of pre-established cryptographic security associations. Together with the fact that less handshakes are needed, this may lead to faster call setup times.

However, path-decoupled signaling has also disadvantages. As only the B2BUA has a signaling association with the firewalls, it must be able to send meaningful error messages to the other SIP entities and gracefully abort the session in case of problems with the firewall signaling [10].

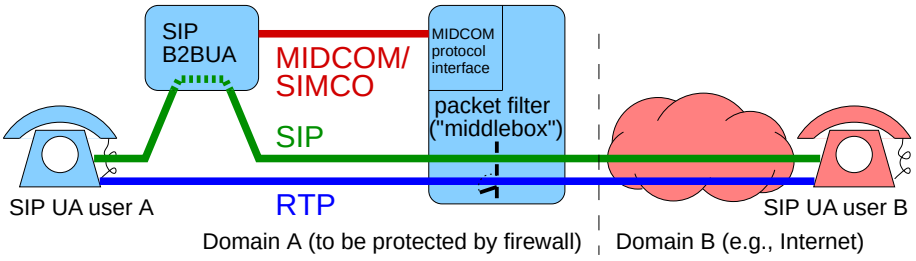

Figure 7: The IETF MIDCOM architecture

Furthermore, the B2BUA needs knowledge about the network topology and routing tables to determine which firewalls need additional pinholes for a new call. Depending on the scenario this might be a nontrivial task. A straightforward solution would be to open pinholes in all firewalls in the domain, possibly causing additional signaling overhead and security problems.

3.3 SIMCO

MIDCOM is not a specific protocol but a framework architecture, including an abstract protocol semantics [26] that can be implemented in several ways [27], e. g., by means of a suitably crafted Management Information Base (MIB) for the Simple Network Management Protocol (SNMPv3).

An alternative approach is the SIMCO (SImple Middlebox COnfiguration) protocol [28], a transaction based protocol using simple binary type-length-value message encoding. A transaction consists of a request from the SIMCO agent (e. g., embedded in the SIP B2BUA) and a positive or negative reply from the middlebox. SIMCO transactions are used to create, modify or delete policy rules at the middlebox, which are the generalized concept of pinholes in packet filters, address bindings in NATs, etc. They are described by various address parameters. Policy rules are soft states, i. e., they are associated with a lifetime attribute and will be removed automatically from the middlebox if the lifetime is not refreshed in time.

All SIMCO messages belonging to one transaction are identified by means of a transaction identifier (TID), which is uniquely assigned by the SIMCO agent (Fig. 9). When an agent asks the middlebox to establish a new policy rule, e. g., by means of a PER (Policy Enable Rule)

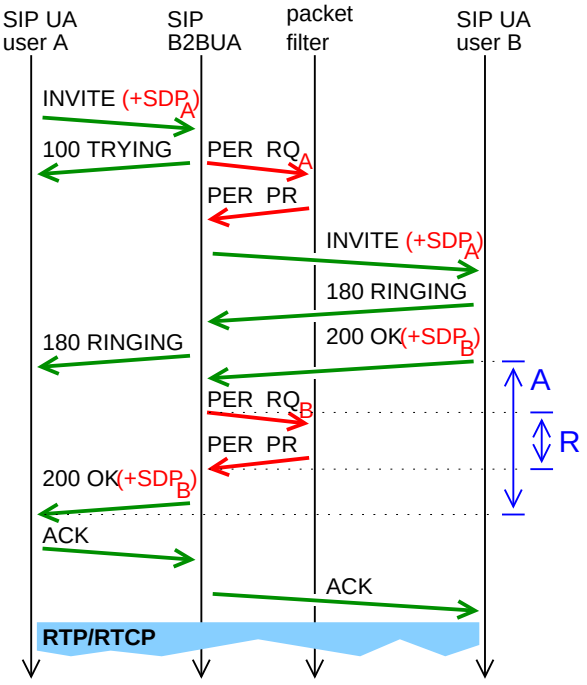

Figure 8: Interaction of SIP and MIDCOM/SIMCO signaling.
The transaction delay R for opening the second pinhole contributes to the unpleasant answer signal delay A.

request, the middlebox creates the rule and assigns a unique policy rule identifier (PID) to it. The PID is returned to the agent, e. g., in the PER positive reply. The agent uses the PID to refer to this specific policy rule in later transactions, such as PLC (Policy Lifetime Change) for refreshing the softstate or deleting the rule (new lifetime = 0).

In addition to the transactions explained above, SIMCO specifies transactions for the management of a SIMCO association and asynchronous notifications to be sent from the middlebox to the agent, e. g., if a policy rule is deleted because of expired lifetime. The SIMCO specification [28] assumes that all transactions between agent and middlebox are transported in one SIMCO association over a single persistent TCP connection. They are established in advance in order to avoid transaction delays caused by the TCP and SIMCO handshake.

Fig. 8 shows the interaction of SIMCO with SIP. Two PER transactions are required to admit the media streams related to a call, one per direction. Additional pinholes may be required in order to support early media.

The first request is sent to open a pinhole that allows media from the called party to the calling party. The second *PER* request is sent by the B2BUA upon reception of the *200 OK* message, when the called party has already picked up the receiver and the conversation is about to start. As also shown in Fig. 8, the response time R of this second SIMCO transaction contributes to the answer signal delay A, which is inconvenient for the users and should therefore be minimized [11]. R consists of the local processing time in the middlebox plus the message transmission delay.

In the remainder of this paper, we will focus on the latter effect and investigate in detail how the SIMCO message transmission delay can be influenced by the choice of the transport layer protocol and its parametrization as it can be configured by the application.

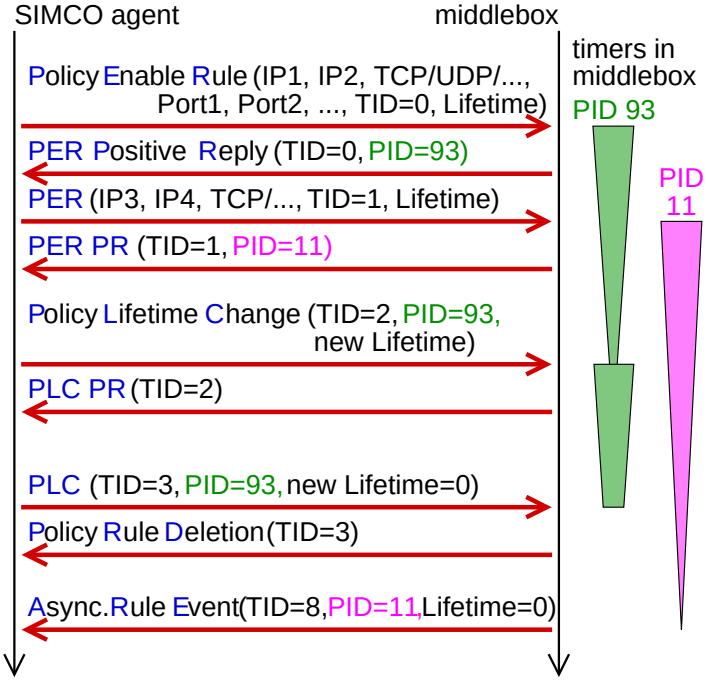

Figure 9: SIMCO transactions (identified by TIDs) create, modify and delete policy rules (identified by PIDs)

4 Candidate Transport Protocols for Firewall Control Signaling

This section briefly reviews the three transport protocols UDP, TCP, and SCTP and discusses to which extend they are suited for (firewall) signaling transport.

4.1 User Datagram Protocol (UDP)

The User Datagram Protocol (UDP) extends the services offered by the IP layer only by adding a multiplexing layer (port numbers). UDP offers connectionless, unreliable transport. No protocol mechanisms are provided for detection of and recovery from errors, and for flow and congestion control.

4.2 Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP), in contrast, is a reliable connection oriented protocol that provides error protection, flow and congestion control. It is the most widely used transport layer protocol in the Internet, both for bulk data transfer and interactive applications.

4.3 Stream Control Transmission Protocol (SCTP)

The Stream Control Transmission Protocol (SCTP) has been developed as a third transport layer protocol for IP. It has originally been designed as part of the IETF SIGTRAN architecture for the transport of "Signaling System No. 7" (SS7) [29] messages over IP. However, this rather special purpose is achieved by adaptation layers on top of SCTP. SCTP itself is a generic transport protocol for IP networks, focusing on environments with high reliability and security requirements. It supports so-called multihoming and re-direction mechanisms in case of link failure. SCTP also uses so-called "verification tags" and a "state cookie" mechanism [30] to protect itself from denial-of-service (DoS) and blind spoofing attacks.

With respect to user data transmission, SCTP combines properties of UDP and TCP, and adds new features as well. SCTP uses the unreliable connectionless packet service offered by IP to provide its upper layer protocol (ULP) with a reliable datagram service. Unlike UDP, SCTP detects packet loss, duplicate packets or bit errors and retransmits or discards the respective packets. SCTP also uses flow control and congestion control algorithms similar to those of TCP. Unlike TCP, SCTP preserves the boundaries of messages: Distinct byte blocks are passed from and to SCTP's ULP instead of a continuous byte stream as with TCP. Therefore, no byte counters or frame delimiters are needed in the application layer for the receiver to dissect the stream into the individual messages.

SCTP allows to split one association (SCTP term for connection) into up to 65536 logical subchannels per direction, so-called "streams". Each user message is transmitted in one of these streams, as selected by the SCTP user. Messages with the unordered flag will be delivered to the upper layer protocol entity as they arrive at the receiver. For all other messages SCTP ensures

in-order delivery, but only within the same stream. Using one association that is split up into several streams – instead of using multiple associations bearing only one stream each – reduces the overhead at connection setup and improves the efficiency of the TCP-like fast retransmit algorithm by using it on the aggregate message flow.

4.4 Head-of-line Blocking in Transport Protocols

Head-of-line blocking can occur when transport protocols offer ordered reliable service, as TCP does: If IP packets get lost, subsequent messages have to wait for the successful retransmission in the receiver queue and are thus delayed. This is illustrated for one TCP connection in Fig 10.

However, reliability (i. e., protection against message loss) and ordered delivery (i. e., passing the messages to the receiving application in the same sequence as they were sent by the sender) are orthogonal issues [31]. Many signaling applications have high reliability requirements and thus need a reliable transport protocol, but do not require ordered delivery. With respect to ordering, the requirements of applications can be subdivided into three classes [31]: (1) *ordered*, (2) *partial-ordered*, or (3) *unordered* delivery. In the second case, the transport protocol must preserve only the ordering relation of subsets of all messages.

As already mentioned, SCTP allows to offer such a partial-ordered or unordered transport: If one message is lost or corrupted in the network and has to be retransmitted, only the corresponding stream is subject to head-of-line blocking whereas messages for other streams can be delivered, anyway (see Fig. 11). This reduces resequencing delays in the receiver and thus the mean end-to-end delay.

Due to real-time requirements, such resequencing delays are in particular critical for signaling protocols. Of course, packet loss probabilities in well-dimensioned signaling networks are usually small. However, on signaling links with a high amount of traffic, many messages may be affected even by rare packet loss.

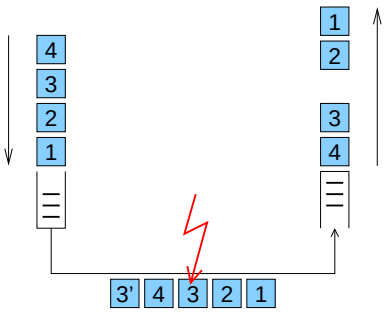

Figure 10: Head-of-line blocking in one TCP connection

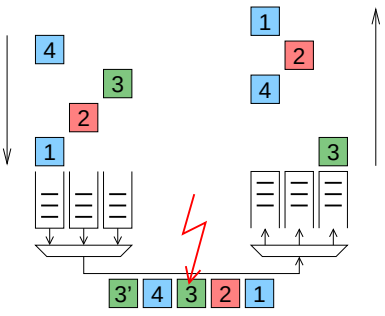

Figure 11: No head-of-line blocking in one SCTP association with 3 streams

5 SIMCO Signaling Transport

In this section, we analyze SIMCO's requirements with respect to ordering and discuss different options for its transport. Then, we compare different solutions that allow to reduce the impact of head-of-line blocking.

5.1 SIMCO's ordering requirements

SIMCO is not designed to work properly if its messages are reordered by the underlying transport – like many other application protocols. For example, consider a B2BUA with SIMCO agent requesting a lifetime extension (PLC with non-zero value) for a policy rule related to an ongoing call. Immediately afterwards it receives a SIP *BYE* terminating the call and thus sends a request to delete the pinhole (PLC with zero value). Under normal conditions it would receive two PLC positive replies. If, however, the lifetime extension request was processed at the middlebox *after* the delete request due to message reordering, it would be answered with a negative reply, as the PID it refers to would no longer exist. This could set off an alarm at the agent.

SIMCO itself has no mechanisms to detect message reordering, such as sequence numbers. Therefore, transactions that refer to the same PID must be delivered in sequence by the transport layer protocol. However, there is no requirement that prohibits reordering of SIMCO messages that refer to different policy rules. As a consequence, SIMCO is one example for an application-layer protocol that can benefit from a partial-ordered transport service.

5.2 One TCP Connection

The specified default transport for SIMCO is to send all messages between agent and middlebox over one persistent TCP connection. As TCP offers reliable ordered transport, SIMCO does not need any mechanisms to recover from message loss. However, the completely ordered transport, which is not needed by SIMCO, is susceptible to head-of-line blocking.

Furthermore, because TCP transports a continuous byte stream instead of datagrams, SIMCO entities must split the incoming stream into individual messages, based on the header's length field. They also must be prepared that a read system call could yield only parts of a message, which have to be buffered until the remaining parts arrive, requiring buffer space and additional states in the protocol state machine.

5.3 Multiple TCP Connections

One solution to reduce head-of-line blocking is to use several parallel TCP connections between the same two end systems. If one connection is subject to head-of-line blocking, other connections can still deliver messages. For partial-ordered delivery, all messages that are in a causal relationship have to be transmitted via the same connection.

This solution does not require a new transport protocol. However, it has several drawbacks compared to SCTP, which will be considered below: First, there is more overhead, as each TCP

connection must be established, maintained, and closed separately. Second, management of buffers, which are required for the reassembly of SIMCO messages that were received in parts, is more complex for parallel TCP connections. And third, each TCP connection has to recover from packet loss independently, whereas SCTP applies error recovery and congestion control mechanisms to the aggregated message flow, which is more efficient [32, 33].

5.4 SCTP Multistreaming

One SCTP association with several streams can provide a reliable, partial-ordered transport for SIMCO while limiting the impact of head-of-line blocking. In addition to this, SIMCO can benefit from other SCTP features such as multihoming for enhanced reliability. Two issues have to be dealt with for efficient use of SCTP multistreaming:

First, how many streams to use for good performance results while not wasting resources. This will be investigated in detail in the later sections of this paper.

The other problem is how to distribute SIMCO messages evenly over several streams while retaining causality for SIMCO. As outlined in Section 5.1, it is important that the transport layer preserves the order of transactions that refer to the same policy rule, but messages referring to different policy rules may be reordered. The basic idea of our specification [34] is therefore that SIMCO messages requesting a new policy rule shall be assigned to an SCTP stream by means of a simple round robin (or similar) scheme. Once this has been decided, all subsequent messages related to the policy rule in question must use the same bidirectional stream pair.

This straightforward idea is complicated in practice by the fact that PIDs are assigned by the middlebox, not by the agent, as shown in Section 3.3 and Fig. 9. Thus, when sending an initial request the agent does not yet know the PID. Therefore, it has to remember the request's TID and wait for the corresponding reply containing the PID, before it can populate a table with the new PID-to-stream mapping. This assignment then can be used for sending all subsequent messages. Special care has been taken to minimize the additional state information. In particular, no additional per-rule state information has to be maintained at the middlebox to support SIMCO over SCTP multistreaming. The detailed specification of our approach, including several special cases, can be found in [34].

5.5 SCTP Unordered Transport

When sending messages in unordered mode, SCTP offers reliable transport, but delivers messages to the upper layer protocol as they arrive. This solution completely avoids head-of-line blocking in the transport layer. However, the upper layer protocol must have own mechanisms to deal with potentially reordered messages, which could introduce delays there. While SIMCO is not designed to handle reordered messages, this mode of operation is used, e. g., for SIP over SCTP [35].

5.6 UDP-based transport

The transport of SIMCO messages over UDP is not considered by the SIMCO specification [28]. Therefore, this mode of transport would require substantial changes to the SIMCO specification. Nevertheless, in order to complement this study, we briefly discuss some of the issues such a solution would have to address.

A "SIMCO over UDP" approach would have first to specify an error detection mechanism such as a timeout and a retransmission strategy in case of message loss. Second, a protection against message duplication would be needed, and mechanisms to order all messages referring to one PID. And third, additional acknowledgement and retransmission mechanisms would be required for asynchronous notifications sent from the middlebox to the agents.

At the end, such a solution would probably be similar to the mechanisms SIP uses when it is transported over UDP. However, it would still lack several important features of the TCP- or SCTP-based solutions, such as flow control and congestion control.[1] Adding these features would require a reimplementation of large parts of the TCP or SCTP protocol stacks at application level. This would be a cumbersome and error-prone violation of the principles of protocol layering and software modularization and have no significant advantages, in particular for highly loaded signaling links that require an efficient protocol operation.

5.7 SCTP Usage and Parametrization for Other Signaling Protocols

Avoiding head-of-line blocking by using SCTP has also been addressed for a couple of other signaling protocols. Several recent documents specify how to transport a given protocol over SCTP. They make different use of SCTP's possibilities with respect to the number of streams, ordered vs. unordered transport, etc. In the following, a short overview of the different approaches is given.

Several different adaptation layers have been specified for the transport of "SS7 over IP" (SIGTRAN). The MTP2 User Peer-to-Peer Adaptation Layer (M2PA) [36] uses two streams to separate protocol-internal control messages from payload data. The MTP3 User Adaptation Layer M3UA [37] uses several streams. Similar to SCTP, the MTP3 offers partial-ordered transport based on Signaling Link Selection (SLS) fields. In order to retain this ordering, messages with the same SLS are mapped to the same SCTP stream. Other adaptation layers such as M2UA, SUA, etc. use similar concepts.

SIP [35] uses only one SCTP stream and sends the messages with the "unordered" flag. Message resequencing at the receiver is done by SIP using the mechanisms that have to be there anyway for the mandatory support of UDP-based transport.

MEGACO/H.248 [38] uses partial-ordered delivery over multiple SCTP streams to avoid head-of-line blocking. There are also proposals to distribute requests over four streams, according to the corresponding phone call's priority. Based on the inbound stream number the receiver could tell a message's degree of urgency, without having to parse it first.

[1] The Datagram Congestion Control Protocol (DCCP, RFC 4340) provides connection oriented, unreliable datagram transport with congestion control. Except for the congestion control, the problems of UDP-based transport would apply for SIMCO over DCCP as well.

DIAMETER [39] distributes its messages evenly over a negotiable number of streams to mitigate head-of-line blocking. It is disallowed to use streams for identification purposes or sequence protection.

The NSIS messaging layer GIST [40] uses several SCTP streams to reduce head-of-line blocking while retaining the sequence of messages that affect the same resource.

The protocols in this section have either been designed explicitly for use with SCTP, or they can use both TCP- and UDP-based transport. The support of UDP implies that the protocol must have means to ensure reliability and in-order delivery where needed. The latter facilitates the transition to SCTP-based transport, e. g., by using one SCTP stream and messages with the "unordered" flag. SIMCO is special as it has been designed originally only for completely ordered, TCP-based transport. Therefore, special attention has to be paid to retaining causality for SIMCO, when mapping messages to streams [34] in order to reduce head-of-line blocking.

6 Modeling Transaction Delays over Different Transport Protocols

In this section, we present analytical models for the response time of SIMCO when different transport protocols are used, in particular focusing on the impact of IP packet losses and potential head-of-line blocking. We compare partial-ordered data transmission over $N \geq 1$ SCTP streams, SCTP unordered mode, and the usage of one (or several) TCP connections. Additionally, a hypothetical UDP-based transport of SIMCO is considered. It should be noted that only few aspects of our model are SIMCO-specific. Thus, the results can be applied to other transaction-based signaling protocols as well.

6.1 Related Work

While many studies on TCP and SCTP performance exist, only few have addressed end-to-end delays of signaling messages and the impact of head-of-line blocking. In general, resequencing delays are a well-known effect and sophisticated theoretical models for automatic repeat request (ARQ) protocols have been developed. An overview can be found for instance in [41]. However, these ARQ models do not consider the specific algorithms used by TCP and SCTP, such as the fast retransmit mechanism.

Several simulation-based studies have compared end-to-end delays of TCP and SCTP, but their results are ambiguous: [42] compares the delay of SIP messages transported over UDP, TCP, or SCTP, respectively. In the latter case, only one stream with reliable unordered service is used. The authors conclude that for packet loss probabilities smaller than 0.3 % head-of-line blocking in TCP does not introduce a significant performance decrease compared to SCTP. A similar work [43] finds no significant differences between SCTP and TCP with selective acknowledgments. However, these simulation results contradict recent measurements based on FreeBSD [44]. None of these studies considers the case of partial-ordered delivery over multiple SCTP streams.

[45] analyzes the usage of multiple SCTP streams for a parallel computing message passing middleware and shows that multiple streams can improve the transmission delays. However, this work is based on measurements only and uses messages that are orders of magnitude larger than typical signaling data. [46] contains a simple analytical SCTP delay analysis for two SCTP streams, but only for the case of asymmetric links. An analytical model for an arbitrary number of SCTP streams has been published first in [1] and generalized in [33]. In the following, we present a further extension of this analysis.

6.2 Model Assumptions

We assume that the path between the two endpoints has a constant unidirectional delay of Δ and thus a minimum round-trip time $RTT = 2\,\Delta$. The path is supposed to suffer from symmetric random packet losses with loss probability p, which may be caused by congestion or transmission errors. Of course, for a well-dimensioned signaling network p is likely to be small. Still, it is important to quantify the performance impact of lossy links in order to derive system dimensioning guidelines. Due to the packet loss in both directions, an acknowledgment for an SCTP data chunk or TCP segment arrives at the sender with probability $p_S = (1-p)^2$.

Furthermore, we assume that the sender window does not restrict the amount of data that can be sent, i. e., we neglect the congestion control. As shown later, this is a reasonable assumption as long as p is small.

For bidirectional transactions as in the case of SIMCO, the main performance metric is the transaction response time R. As already mentioned, the SIMCO response time R must be small to minimize the answer signal delay perceived by users. Error recovery mechanisms and potential head-of-line blocking increase the unidirectional latency $RTT/2$ by an additional delay component W, which depends on the transport protocol. Since transactions require bidirectional message exchange and since head-of-line blocking may occur in both directions, the mean transaction response time thus follows as

$$R = RTT + 2\,W + \delta\,, \tag{1}$$

where δ represents the processing time in the end systems.

6.3 Workload Model

The number of SIMCO transactions required to open, maintain, and close a pinhole depends on the call duration. As shown in Fig. 8, two pinholes are required to establish a call. According to Fig. 9, the pinhole is opened by a PER request, and a PLC with a lifetime extension of 0 is sent when the call is terminated. Due to additional PLCs during the call, the total number of SIMCO transactions per pinhole is $n(T) = 2 + \lfloor \frac{T}{L} \rfloor$, where T is the call duration and L the lifetime extension period. The overall rate of SIMCO transactions depends on the call duration distribution $f(T)$ and the rate of pinhole opening requests λ:

$$\lambda_{\text{SIMCO}} = \lambda \cdot \int_0^\infty n(T) \cdot f(T)\,\mathrm{d}T\,. \tag{2}$$

If we assume that the call duration is exponentially distributed with mean h and PDF $f(T) = \frac{1}{h}\exp(-\frac{T}{h})$, the mean inter-arrival time (IAT) d of SIMCO messages in one direction can be approximated as

$$d = \frac{1}{\lambda_{\text{SIMCO}}} \approx \frac{1}{\lambda}\left(\frac{3}{2} + \frac{h}{L}\right)^{-1} . \tag{3}$$

6.4 Analysis of SCTP Error Recovery

Similar to TCP, SCTP can recover from packet loss by two mechanisms: The *fast retransmit* and the *retransmission timeout*. In the following, we review their function and quantify their impact on SCTP end-to-end delays when packet loss occurs.

Due to the packet loss in both directions, an acknowledgement for a DATA chunk arrives at the sender with probability $p_{\text{S}} = (1-p)^2$. An endpoint can detect packet loss if transmission sequence numbers (TSNs) are missing in the selective acknowledgements (SACKs). A SACK, which is sent upon the reception of a DATA chunk on one stream, contains missing TSN reports for all streams. An SCTP endpoint retransmits data when three subsequent SACKs include a missing report [47].

This SCTP error recovery by a *fast retransmit* is illustrated in Fig. 12. For this figure, we assume that the SCTP association has $N = 3$ streams and a round robin scheduling strategy is applied, i. e., the DATA chunks with transmission sequence numbers 0, 3, 6, ... are sent via stream #0, while DATA chunks with TSNs 1, 4, 7, ... and 2, 5, 8, ... are sent via streams #1 and #2, respectively. For simplicity, DATA chunks are here supposed to be sent with constant IAT d.

In this example, the DATA chunk with TSN 0 is lost. t_0 denotes the point in time when this packet is sent, t_1 is when it should arrive at the receiver. At $t_2 = t_0 + RTT + 3\,d$ the sender has received 3 SACK chunks with missing reports and performs the retransmission. Note that SCTP's SACK messages contain information about missing DATA chunks for all streams. When the retransmitted packet arrives at the receiver at $t_3 = t_2 + RTT/2$, all DATA chunks in stream #0's resequencing queue can be delivered to the upper layer protocol entity.

As shown by Fig. 12, the minimum value for the time to detect the packet loss D_{FRTX} is $t_2 - t_0 = RTT + 3\,d$. However, D may be larger if SACKs get lost, too. Each DATA chunk triggers a SACK chunk, but both may get lost. The probability that three SACKs arrive at the sender, after

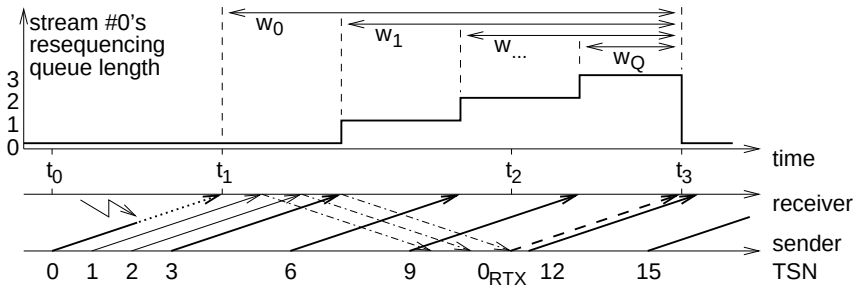

Figure 12: Illustration of resequencing delays for 3 SCTP streams using fast retransmit

i DATA chunks have been sent, is $P(i) = p_S^3 \, (1 - p_S)^{i-3} \, \binom{i-1}{i-3}$. From this follows

$$D_{\text{FRTX}} \approx RTT + d \sum_{i=3}^{\infty} P(i) \, i = RTT + \tfrac{3\,d}{(1-p)^2} \; . \tag{4}$$

The reliable data delivery is also ensured by the *retransmission timeout* mechanism: If the oldest outstanding DATA chunk has not been acknowledged when the retransmission timeout (RTO) expires, missing chunks are retransmitted. As depicted in Fig. 13, the timer is restarted whenever a new acknowledgment arrives. Thus, the error detection time is

$$D_{\text{RTO}} = RTO + \max\left(RTT - d, 0\right) \; . \tag{5}$$

The value of RTO is continuously adapted to the estimated round-trip time and thus not necessarily constant. However, for small network latencies RTO is usually close to its minimum value, i. e., 1 s in case of SCTP.

Considering both recovery mechanisms, the error detection time is

$$D = \min\left(D_{\text{FRTX}}, D_{\text{RTO}}\right) \; . \tag{6}$$

This expression is an approximation only since the retransmission may get lost, too. In this case, a retransmission timeout is definitely required. Several subsequent lost DATA chunks may also trigger overlapping fast recovery periods, which are difficult to describe by a simple model. We neglect both effects in this model since they hardly occur if the packet loss probability p is small.

6.5 Delay over Multiple SCTP Streams

The waiting times w_n of DATA chunks in the resequencing queue depend on $D = w_0$. The number of DATA chunks that have to be queued until the retransmission arrives is $Q = \lfloor \frac{D}{d\,N} \rfloor$. The resequencing delay of the first DATA chunk after the lost one is given by $w_1 = D - N\,d$. The subsequent waiting times are $w_2 = D - 2\,N\,d$, …, $w_Q = D - Q\,N\,d$. The mean waiting time is the sum of all w_i divided by the mean number of DATA chunks between two losses, which is $1/p$. The mean increase of the unidirectional end-to-end delivery delay is thus

$$W = p \sum_{i=0}^{Q} w_i = p \left((Q+1) \cdot D - \tfrac{Q\,(Q+1)}{2} \, N\,d \right) \; . \tag{7}$$

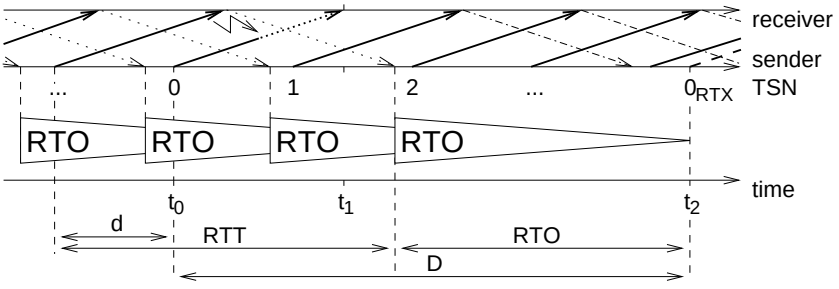

Figure 13: Timeout-based error recovery for one SCTP stream

An important question, which already has been raised in Section 5.4, is the optimal number of SCTP streams. Using a large number of streams may not be efficient since this may waste resources (e. g., memory) in the endpoints. Under the assumption that for small values of p at most one stream is blocked, head-of-line blocking can be avoided completely if no DATA chunks get queued before the retransmission is triggered, i. e., $Q = 0$. This is fulfilled for $N \geq M$ with $M = \left\lceil \frac{D}{d} \right\rceil$. According to (4), M is quite insensitive to p. From this follows the optimal number of streams as

$$M \approx \left\lceil RTT \cdot \lambda \cdot \left(\tfrac{3}{2} + \tfrac{h}{L}\right) + 3 \right\rceil \ . \tag{8}$$

6.6 Delays over SCTP Unordered Mode

Head-of-line blocking can be completely avoided by setting the unordered flag in the messages. Then, the messages are delivered to the upper layer protocol as they arrive. If all messages are sent in this mode, i. e., $w_0 = D$, $w_1 = w_2 = \cdots = 0$, eq. (7) yields

$$W = p \cdot D \ . \tag{9}$$

6.7 Delay over TCP

From a theoretical point of view, the delay of signaling transport over a TCP connection should be identical to using one SCTP stream, since both protocols use very similar error recovery algorithms. Thus, a model for TCP resequencing delays can be easily obtained by setting $N = 1$ in eq. (7).

Another possibility is to use several TCP connections in parallel, combined with a load balancing mechanism. However, different to SCTP, these multiple connections have an independent error recovery, i. e., acknowledgments only refer to data transmitted over the same connection. Since it takes longer to trigger a fast retransmit, multiple TCP connections usually suffer from larger end-to-end delays than one connection. This is why we do not further detail this approach here. An analysis of the usage of multiple parallel TCP connections can be found in [33].

6.8 Delay over UDP

For the sake of completeness, a hypothetical "SIMCO over UDP" transport shall be analyzed, too. As already explained, this would require substantial changes to SIMCO, which have not been standardized. Therefore, for the following analysis, we assume protocol mechanisms and parametrization similar to "SIP over UDP," as specified in [48].

In this case, the client's application layer retransmits a request if the corresponding reply is not received within a timeout period, which is initially set to $TO = 500\,\text{ms}$ and doubled with every failed retransmission attempt. The transaction is completed when, after i unsuccessful sending attempts, an attempt is successful. This happens with probability $q_i = (1 - p_\text{S})^i \cdot p_\text{S}$. The total waiting time before sending this last request is $W_i = TO\,(2^i - 1)$. The mean transaction

response time follows as

$$R = RTT + \delta + \sum_{i=0}^{\infty} q_i \cdot W_i = RTT + \delta + TO \frac{1 - p_S}{2\,p_S - 1} \,. \tag{10}$$

7 Performance Evaluation

7.1 SIMCO Implementation

In order to compare the performance of firewall signaling over different transport protocols, a prototype compliant to [28, 34] has been implemented [49]. As shown in Fig. 14, the middlebox software can control a packet filter (Linux Netfilter). For functional tests, the SIMCO agent has been integrated into the VOVIDA SIP back-to-back user agent (B2BUA) [50, 10]. Furthermore, a load generator emulating user behavior has been implemented for measuring the SIMCO transaction response time (see Fig. 15).

The SIMCO software was implemented in C++ for Linux (kernel 2.6.16) and Solaris 10. The Linux version can use either the "lksctp"-kernel module or standard Linux TCP (using SACKs and the default "BIC" congestion control). For both protocols the "nodelay" socket options have been enabled. It should be noted that the SCTP-specific parts of the source code are much simpler because of SCTP's message oriented API. For Solaris, we only present the TCP results. Our Solaris SCTP measurements are distorted by sporadic longer stalls of the SCTP association, which seem to occur without any obvious reason.

7.2 Measurement Setup

In the following we present selected measurement results for scenarios with a signaling link between a central softswitch and a large firewall (e. g., at the peering point between two large providers).

Measurements were made using two 2.4 GHz Pentium 4 or 500 MHz UltraSPARC IIe computers connected by 100 Mbps Ethernet to a "NIST Net" network emulator, which adds a delay of $\Delta = 10\,\text{ms}$ in each direction and randomly drops IP packets with a given probability p. All

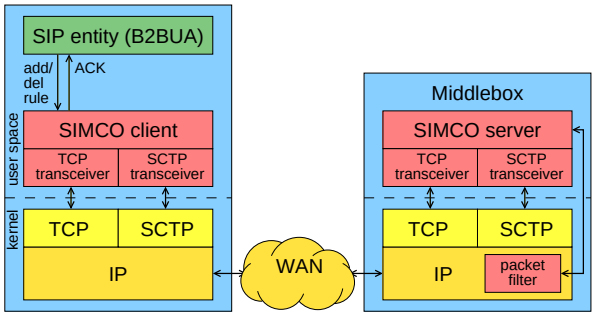

Figure 14: SIMCO/SCTP prototype with B2BUA for proof of concept in SIP testbed

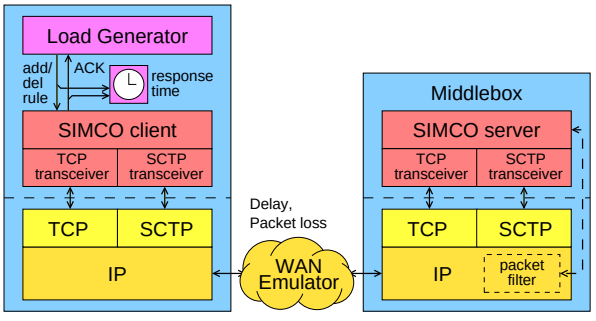

Figure 15: SIMCO/SCTP testbed with load generator for performance measurements

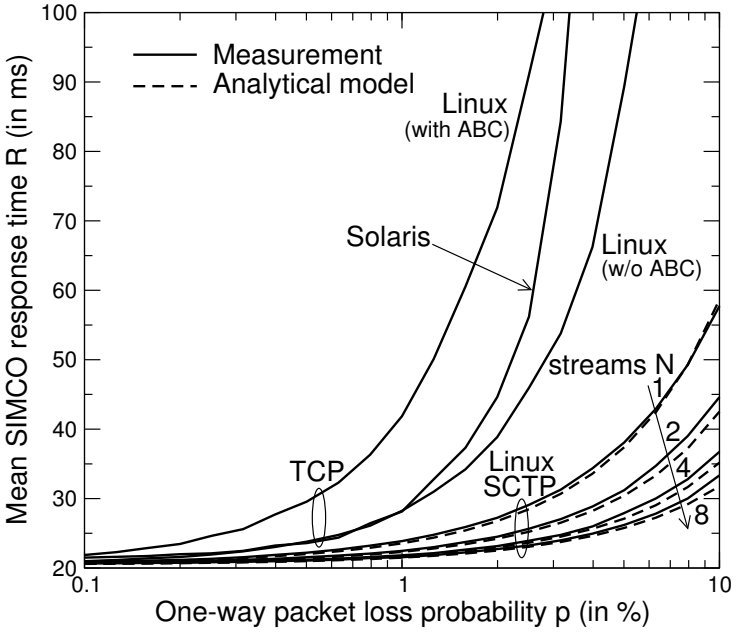

Figure 16: Comparison of SCTP/TCP

measurement values have been obtained by averaging over the SIMCO response time of PER requests during a measurement period of 1000 s after the load generator has reached the steady state. The interaction of the middlebox software with the packet filter was disabled to isolate the measured delay from this overhead.

Unless otherwise described, the load generator requests pinhole openings with exponential IAT $\frac{1}{\lambda} = 30\,\text{ms}$ and exponential lifetime $h = 180\,\text{s}$. With two pinholes per call and a typical busy hour load of $\rho = 0.05\,\text{Erlang}$, this corresponds to a mean number of $m = \frac{1}{2}\,h\,\lambda = 3,000$ simultaneous calls and $S = \frac{h\,\lambda}{2\,\rho} = 60,000$ subscribers. The rule softstates are refreshed every $L = 120\,\text{s}$. From this follows $d \approx 10\,\text{ms}$ as mean IAT of SIMCO messages.

In addition to the results presented here, measurements have also been performed for other scenarios, with very similar results. A detailed documentation of the measurements with our SIMCO prototype can be found in a technical report [51].

7.3 Impact of Packet Loss on SCTP

First, we analyze the usage of multiple SCTP streams for SIMCO. Fig. 16 shows the mean SIMCO response time as a function of the packet loss probability p. The results reveal that using more than one SCTP stream can significantly improve the SIMCO response time R even for moderate loss probabilities such as $p = 2\,\%$. The difference gets larger for higher p, but such situations will hardly occur in well-dimensioned signaling networks.

Fig. 17 presents the SCTP measurement results as a function of the number of streams N. They match very well the response time predicted by the analytical model of eq. (1) and (7), with a processing delay assumed to be $\delta = 0.5\,\text{ms}$. The model slightly underestimates the response time for $p > 1\,\%$. This is probably due to the impact of multiple fast retransmits and timeouts that cannot be neglected for high loss probabilities. Fig. 17 also confirms that using a value N

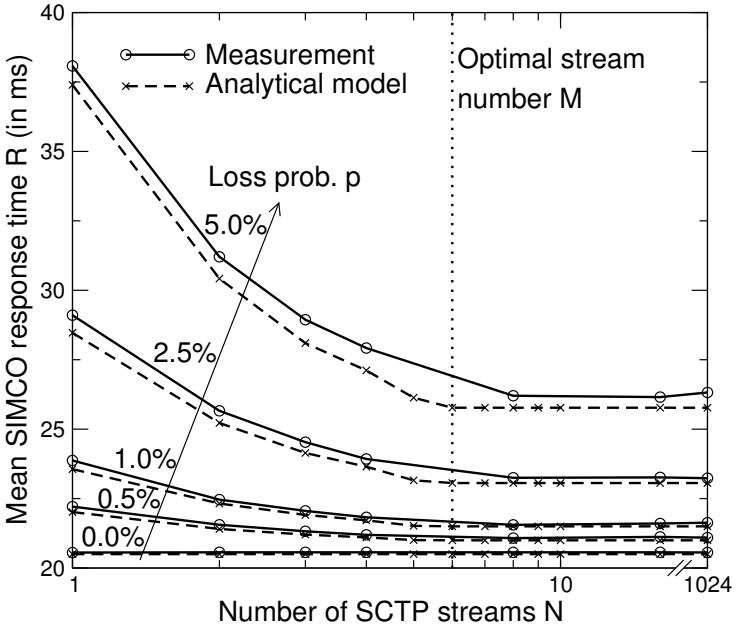

Figure 17: Impact of SCTP streams number

larger than the optimum value M (here: $M = 6$) does not significantly improve performance.

The cumulative complementary distribution function (CCDF) for $p = 1\,\%$ is depicted in Fig. 18. As the number of streams N gets larger, the CCDF asymptotically approaches the case of unordered delivery. When N is larger than M, unordered delivery and multiple steams have the same performance. The small steps in the CCDF having a width of about 4 ms can be attributed to the Linux scheduler running at 250 Hz. Furthermore, a larger step at $R = 30$ ms can be observed. Looking at traces reveals that this additional delay of 10 ms is caused by sporadic bundling of data chunks: When two subsequent chunks are assembled to one packet, the first one is delayed by one IAT, i. e., in the mean by $d = 10$ ms.

7.4 Impact of Packet Loss on TCP

Fig. 16 also presents measurement results for TCP, both for Linux and Solaris operating systems. In theory, one would assume that TCP has a similar performance like SCTP with one stream. However, according to our measurements the mean SIMCO response time is significantly larger, in particular for $p > 1\,\%$. In case of Linux, the response time can be reduced by not using the "Appropriate Byte Counting" (ABC) [52] mechanism, which is an experimental extension of the TCP congestion control.[2] Without ABC, Linux and Solaris TCP have a similar performance. Both implementations are not able to transport the offered load of 100 transactions/s for $p > 7\,\%$, which is manifested by socket buffer overflows. This can be explained by the TCP congestion control that limits the throughput when losses occur.

The CCDF in Fig. 18 reveals a non-negligible probability for high delays (for $p = 1\,\%$). The

[2] Appropriate Byte Counting (ABC) was enabled by default in the Linux kernel used in our measurements. However, it has been observed that this ABC implementation unfairly penalizes applications that send small amounts of data. The SIMCO firewall signaling is such an example. As a consequence of these problems, ABC has been turned off by default in Linux kernel 2.6.18.

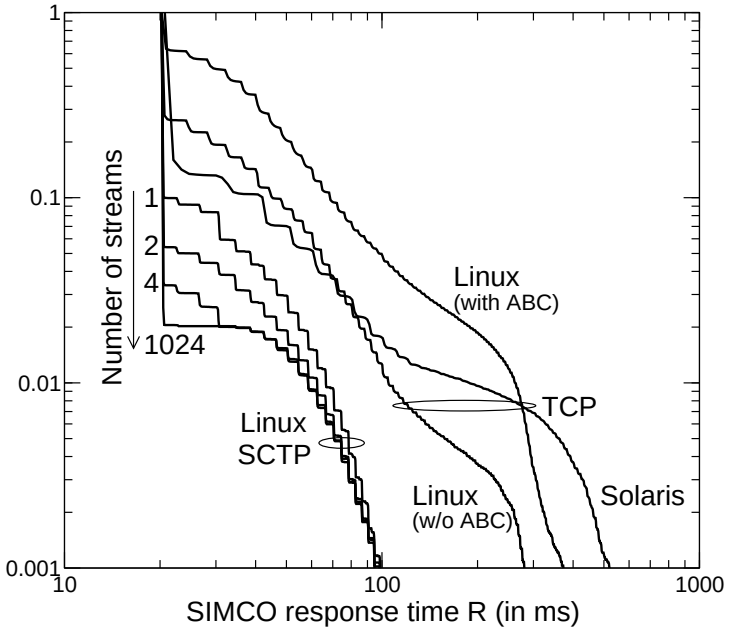

Figure 18: SIMCO response time CCDF ($p = 1\,\%$, $d = 10\,\text{ms}$)

99 % quantile of the response time is in the order of 200 ms. For Solaris TCP, there is a significant probability for delays of about 500 ms. They seem to be caused by retransmission timeouts with a minimum duration of $RTO \approx 500\,\text{ms}$. Linux TCP uses a smaller minimum value of $RTO \approx RTT + 200\,\text{ms}$ [53] and thus does not suffer that much from long delays. However, there is a significant probability of small delays. An analysis of traces shows that Linux sometimes does not send data immediately to the network, but aggregates them to larger packets, even though the "nodelay" option is set and the so-called "Nagle" algorithm thus should be disabled.

In [33], we have verified these measurements with a pair of simple test programs that use straightforward socket calls. These tests confirm the difference between TCP and SCTP, i. e., these effects are not specific to our SIMCO prototype.

7.5 Variable Load

In the following, we vary the offered load λ_{SIMCO} from a rather low data rate of 1 transaction/s to 10.000 transactions/s. Fig. 19 presents the mean SIMCO response time for Linux SCTP, Linux TCP and Solaris TCP. The diagram can be subdivided into three regions:

(1) For small data rates, the response time is increased by the RTO, which expires before a fast retransmit is triggered. Here, SCTP performs worst due to its high minimum RTO of 1 s. As λ_{SIMCO} increases, R increases due to head-of-line blocking, except for a sufficiently large number of SCTP streams.

(2) For $\lambda_{\text{SIMCO}} > \alpha = \frac{3}{RTO}$, the fast retransmit allows a faster error recovery and reduces the end-to-end delay significantly. For Linux SCTP and Solaris TCP, the measured response time is rather close to the value predicted by our analytical model, up to a message rate of about $100\,\frac{1}{\text{s}}$. For Linux TCP, an additional delay of 10 ms to 20 ms can be observed.

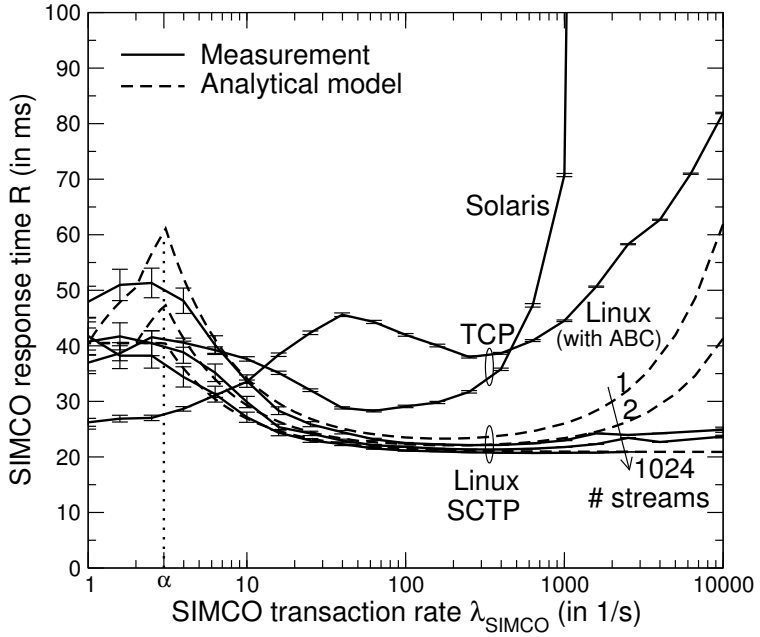

Figure 19: Variable load ($p = 1\,\%$)

(3) The behavior at high data rates differs significantly: The Linux SCTP delay is close to the RTT when a sufficient number of streams or unordered mode is used. The maximum data rate that we could transport with the "experimental" Linux SCTP is about $\lambda_{\text{SIMCO}} = 2,000\,\frac{1}{\text{s}}$. Under Solaris TCP, delays increase for $\lambda_{\text{SIMCO}} > 500\,\frac{1}{\text{s}}$. Here the TCP congestion control seems to limit the sending rate. Our analytical model does not consider this effect and thus underestimates the response time. Interestingly, Linux TCP can transport at least four times this load before delays start to increase. Apparently, the Linux TCP algorithms trade off delay and throughput.

7.6 Hypothetical UDP transport

Finally, we consider UDP as potential alternative to SCTP or TCP. Since this option has not been implemented in reality, we can only compare the analytical models according to equations (1), (7), and (10). Some example results for transaction response times are listed in Table 1.

In the SIP-like error recovery, each lost message has to be recovered by a timeout with minimum duration $TO = 500\,\text{ms}$, which is of the same order as the minimum retransmission timeouts of SCTP and TCP. If the signaling load is rather low, the simple UDP-based error recovery achieves a similar performance like SCTP and TCP, since they have to recover by means of retransmission timeouts, too. For moderate or high signaling load such as 100 transactions/s, SCTP or TCP can benefit from the more efficient fast retransmit mechanism. Then, the mean transaction response time is much smaller than for the UDP-based transport (see Table 1).

It must be mentioned that one could improve the performance of the UDP-based error recovery by dynamically adapting the timeout duration or implementing an equivalent to the fast retransmit at application level. However, such an approach would soon end up in reimplementing most of TCP or SCTP functionality.

Table 1: Mean transaction delay for SCTP, TCP, and UDP according to the analytical models

Mean transaction delay *W*	**Low load (1 trans/s)**	**High load 100 trans/s)**
SCTP (1 stream)	40.5ms	23.5 ms
SCTP (1024 streams)	40.5ms	21.5 ms
TCP	40.5ms	23.5 ms
UDP	30.9ms	30.9 ms

Parameters: $RTT = 20\,\text{ms}$, $p = 1\,\%$, $RTO = 1\,\text{s}$, $\delta = 0.5\,\text{ms}$

7.7 Summary

Summing up, the transport delay of multiple SCTP streams is close to the theoretical minimum given by our analytical model, whereas using TCP can result in significantly larger delays, in particular for transaction rates in the order of $100\,\frac{1}{\text{s}}$. UDP transport cannot be seen as a real alternative since an efficient operation would require lots of additional functionality to be implemented in the application layer.

8 Conclusions

In this paper we motivate the necessity of access control in multi-domain IP telephony networks, in order to achieve security standards comparable to the PSTN, and we discuss principal solution approaches. Specifically, we detail different architectures and protocols for dynamic firewall control, and present the SIMCO protocol as one solution for path-decoupled signaling. Since dynamic firewall control is subject to stringent timing constraints, different solutions to minimize transport delays on signaling links are compared. We study the performance of using SCTP, TCP and UDP as transport protocols for SIMCO. Compared to TCP, SCTP can significantly reduce the SIMCO response time by leveraging transmission over multiple streams. We propose an analytical model to quantify delays caused by packet loss, and verify it with measurements. We show that a small number of SCTP streams is sufficient to almost completely avoid head-of-line blocking. Furthermore, our measurements, both for Linux and Solaris, reveal that using TCP causes significant delays even for small packet loss probabilities. Finally, a potential UDP transport does not show any significant performance improvements.

References

[1] Sebastian Kiesel and Michael Scharf. Modeling and performance evaluation of SCTP as transport protocol for firewall control. In *Proc. IFIP-TC6 Networking Conference, Springer LNCS 3976*, pages 451–462, Coimbra, Portugal, May 2006.

[2] J.P. Holbrook and J.K. Reynolds. Site Security Handbook. RFC 1244, IETF, July 1991.

[3] Saikat Guha and Paul Francis. Towards a Secure Internet Architecture Through Signaling. Technical Report cul.cis/TR2006-2037, Cornell University, Ithaca, NY, 2006.

[4] Gonzalo Camarillo and Miguel-Angel García-Martín. *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*. Wiley, 2 edition, 2005.

[5] Telecommunications and internet converged services and protocols for advanced networking (TISPAN); NGN functional architecture release 1. ETSI Standard ES 282 001, 2005.

[6] Maresca, M. and Zingirian, N. and Baglietto, P. Internet protocol support for telephony. *Proceedings of the IEEE*, 92(9):1463–1477, 2004.

[7] ITU-T Study Group 13. General principles and general reference model for Next Generation Networks. Rec. Y.2011, ITU-T, October 2004.

[8] R. Penno. SPEERMINT Peering Architecture. IETF draft - work in progress, IETF, draft-ietf-speermint-architecture-00, August 2006.

[9] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser. Terminology for Policy-Based Management. RFC 3198, IETF, November 2001.

[10] Andreas Müller and Sebastian Kiesel. Issues with the interworking of application layer protocols and the MIDCOM architecture. In *Proc. Eunice Summer School*, pages 188–195, Tampere, Finland, 2004.

[11] ITU-T Study Group 2. Network grade of service parameters and target values for circuit-switched services in the evolving ISDN. Rec. E.721, ITU-T, May 1999.

[12] G. Camarillo, Ed., W. Marshall, Ed., and J. Rosenberg. Integration of Resource Management and Session Initiation Protocol (SIP). RFC 3312, IETF, October 2002.

[13] B. Carpenter and S. Brim. Middleboxes: Taxonomy and Issues. RFC 3234, IETF, February 2002.

[14] Salman Baset and Henning Schulzrinne. An analysis of the Skype peer-to-peer internet telephony protocol. In *Proc. IEEE INFOCOM*, Barcelona, Spain, April 2006.

[15] B. Capouch. IAX: Inter-Asterisk eXchange Version 2. IETF draft - work in progress, IETF, draft-guy-iax-01, March 2006.

[16] G. Camarillo. Requirements from SIP (Session Initiation Protocol) Session Border Control Deployments. IETF draft - work in progress, IETF, draft-camarillo-sipping-sbc-funcs-04, June 2006.

[17] Utz Roedig, Manuel Goertz, Martin Karsten, and Ralf Steinmetz. RSVP as firewall signalling protocol. In *Proc. 6th IEEE Symposium on Computers and Communications*, pages 57–62, Hammamet, Tunisia, July 2001.

[18] R. Hancock, G. Karagiannis, J. Loughney, and S. Van den Bosch. Next Steps in Signaling (NSIS): Framework. RFC 4080, IETF, June 2005.

[19] X. Fu, et al. NSIS: a new extensible IP signaling protocol suite. *Communications Magazine, IEEE*, 43(10):133–141, 2005.

[20] H. Schulzrinne and R. Hancock. GIST: General Internet Signaling Transport. IETF draft - work in progress, IETF, draft-ietf-nsis-ntlp-10, July 2006.

[21] M. Stiemerling. NAT/Firewall NSIS Signaling Layer Protocol (NSLP). IETF draft - work in progress, IETF, draft-ietf-nsis-nslp-natfw-12, June 2006.

[22] L-N. Hamer, B. Gage, and H. Shieh. Framework for Session Set-up with Media Authorization. RFC 3521, IETF, April 2003.

[23] J. Manner. Authorization for NSIS Signaling Layer Protocols. IETF draft - work in progress, IETF, draft-manner-nsis-nslp-auth-01, June 2006.

[24] Michael Stier, Emanuel Eick, and Eckhart Koerner. A practical approach to SIP, QoS and AAA integration. In *Proc. IFIP-TC6 Networking Conference, Springer LNCS 3976*, pages 654–665, May 2006.

[25] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan. Middlebox communication architecture and framework. RFC 3303, IETF, August 2002.

[26] M. Stiemerling, J. Quittek, and T. Taylor. Middlebox Communications (MIDCOM) Protocol Semantics. RFC 3989, IETF, February 2005.

[27] M. Barnes (Ed.). Middlebox Communications (MIDCOM) Protocol Evaluation. RFC 4097, IETF, June 2005.

[28] M. Stiemerling, J. Quittek, and C. Cadar. NEC's Simple Middlebox Configuration (SIMCO) Protocol Version 3.0. RFC 4540, IETF, May 2006.

[29] ITU-T Study Group XI. INTRODUCTION TO CCITT SIGNALLING SYSTEM No. 7. Recommendation Q.700, ITU-T, March 1993.

[30] Sebastian Kiesel. On the use of cryptographic cookies for transport layer connection establishment. In *Proc. EUNICE Summer School*, pages 177–184, Trondheim, Norway, 2002.

[31] Paul D. Amer, Christophe Chassot, Thomas J. Connolly, Michel Diaz, and Phillip Conrad. Partial-order transport service for multimedia and other applications. *IEEE/ACM Trans. Netw.*, 2(5):440–456, 1994.

[32] Preethi Natarajan, Janardhan R. Iyengar, Paul D. Amer, and Randall Stewart. SCTP: an innovative transport layer protocol for the web. In *Proc. WWW '06: 15th intl. conf. on World Wide Web*, pages 615–624, Edinburgh, Scotland, 2006.

[33] Michael Scharf and Sebastian Kiesel. Head-of-line blocking in TCP and SCTP: Analysis and measurements. In *Proc. IEEE Globecom*, San Francisco, CA, USA, November 2006.

[34] S. Kiesel. SIMCO over SCTP. IETF draft - work in progress, IETF, draft-kiesel-midcom-simco-sctp-01, April 2006.

[35] J. Rosenberg, H. Schulzrinne, and G. Camarillo. The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP). RFC 4168, IETF, October 2005.

[36] T. George, B. Bidulock, R. Dantu, H. Schwarzbauer, and K. Morneault. Signaling System 7 (SS7) Message Transfer Part 2 (MTP2) – User Peer-to-Peer Adaptation Layer (M2PA). RFC 4165, IETF, September 2005.

[37] G. Sidebottom, Ed., K. Morneault, Ed., J. Pastor-Balbas, and Ed. Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) – User Adaptation Layer (M3UA). RFC 3332, IETF, September 2002.

[38] ITU-T Study Group 16. Gateway control protocol: Transport over Stream Control Transmission Protocol (SCTP) – Corrigendum 1. Rec. H.248.4, ITU-T, March 2004.

[39] B. Aboba and J. Wood. Authentication, Authorization and Accounting (AAA) Transport Profile. RFC 3539, IETF, June 2003.

[40] X. Fu. General Internet Signaling Transport (GIST) over SCTP. IETF draft - work in progress, IETF, draft-ietf-nsis-ntlp-sctp-00, June 2006.

[41] Ye Xia and David Tse. Analysis on packet resequencing for reliable network protocols. *Performance Evaluation*, 61:299–328, 2005.

[42] Gonzalo Camarillo, Raimo Kantola, and Henning Schulzrinne. Evaluation of transport protocols for the session initiation protocol. *IEEE Network*, 17(5):40–46, 2003.

[43] M. Lulling and J. Vaughan. A simulation-based comparative evaluation of tranport protocols for SIP. *Comp. Commun.*, 29(4):525–537, 2006.

[44] Karl-Johan Grinnemo, Torbjörn Andersson, and Anna Brunstrom. Performance benefits of avoiding head-of-line blocking in SCTP. In *Proc. ICAS/ICNS 2005*, Papeete, Tahiti, October 2005.

[45] Humaira Kamal, Brad Penoff, and Alan Wagner. SCTP versus TCP for MPI. In *Proc. Supercomputing 2005*, Seattle, USA, November 2005.

[46] Andreas Jungmaier, Erwin P. Rathgeb, Michael Schopp, and Michael Tüxen. SCTP – a multi-link end-to-end protocol for IP-based networks. *AEÜ Intl. J. of Electron. and Commun.*, 55(1):46–54, 2000.

[47] R. Stewart, I. Arias-Rodriguez, K. Poon, A. Caro, and M. Tuexen. Stream Control Transmission Protocol (SCTP) Specification Errata and Issues. RFC 4460, IETF, April 2006.

[48] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, IETF, June 2002.

[49] Christian Blankenhorn. Evaluation of SCTP as transport protocol for transaction-based applications at the example of a protocol for firewall control. Student project (in German), University of Stuttgart, IKR, 2005.

[50] Andreas Müller. Extension of a SIP proxy by security functions. Student project (in German), University of Stuttgart, IKR, 2004.

[51] Sebastian Kiesel, Michael Scharf, Sebastian Beutel, and Thomas Ruschival. Performance measurement results of SIMCO over TCP and SCTP. Internal Report 53, University of Stuttgart, IKR, 2006.

[52] M. Allman. TCP Congestion Control with Appropriate Byte Counting (ABC). RFC 3465, IETF, February 2003.

[53] P. Sarolahti and A. Kuznetsov. Congestion control in Linux TCP. In *Proc. USENIX Annual Technical Conference*, Monterey, California, USA, June 2002.