Modeling and Performance Evaluation of SCTP as Transport Protocol for Firewall Control

Sebastian Kiesel and Michael Scharf

Institute of Communication Networks and Computer Engineering
University of Stuttgart, Germany
{kiesel,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 VoIP and multimedia applications requires a dynamic control of these firewalls, which can be implemented using the Simple Middlebox Configuration Protocol (SIMCO). In this paper, we study the performance of SCTP and TCP as transport protocols for the transaction-based signaling protocol SIMCO, which requires small end-to-end delays. We present an analytical model in order to quantify the impact of head-of-line blocking in SCTP. Both, the model and measurements reveal that SCTP can significantly reduce the SIMCO response times by leveraging transmission over multiple parallel streams. While a few SCTP streams can almost completely avoid head-of-line blocking, our measurements show that TCP may suffer from rather large end-to-end delays.

1 Introduction

For quite a long time, firewalls have been in use to protect private networks from unwanted access from the Internet. The simplest type are packet filters.

So-called "Next Generation Networks" (NGN) are an emerging technology intended to replace the ISDN based telephone networks by Session Initiation Protocol (SIP) based "Voice over IP" (VoIP) technology in the future. They differ from traditional IP networks by deploying stateful application layer entities such as SIP proxies in the core network. Because of higher security requirements, firewalls are not only used as a customer premises equipment, but also for screening at the interconnection of different operator's networks. Due to the dynamic nature of SIP, these firewalls have to take part in the session signaling. As traffic of many simultaneous calls has to be inspected, performance is an important issue, in particular the call setup delay to which the dynamic configuration of the firewalls is a contributing factor.

The Stream Control Transmission Protocol (SCTP) has been designed as a transport layer protocol especially for signaling applications. Compared to TCP, SCTP adds a multiplexing layer – the so-called streams – on top of its associations. As in-oder delivery of messages is only guaranteed within the same stream, the delaying effect of so-called "head-of-line blocking" can be reduced if the application software can make use of several streams. In this paper, we show how this feature can be used in time-critical firewall control signaling applications.

While some studies on SCTP performance exist, to the best of our knowledge the end-to-end delay of signaling messages over multiple streams has not been addressed so far. We present an analytical model and measurement results to quantify the impact of head-of-line blocking in SCTP. A related work [1] presents simulation results for the transmission delay of SIP messages transported over UDP, TCP, or SCTP, respectively. Unlike our work, this study only uses one SCTP stream with reliable unordered service and leaves message reordering up to SIP. A recent work [2] analyzes the usage of SCTP 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.

The remainder of the paper is organized as follows. In Section 2, we discuss the interaction of SIP-based VoIP signaling and firewalls. We introduce the IETF MIDCOM architecture and the SIMCO protocol that can be used for firewall control. Section 3 discusses which transport protocols can be used for SIMCO. We give a brief overview of SCTP and present the basic idea how SIMCO can leverage SCTP's multiple streams feature. Section 4 proposes an analytical model of the head-of-line blocking in SCTP for signaling traffic. In Section 5, we present a prototype implementation of "SIMCO over SCTP" and performance measurement results. Finally, Section 6 concludes this paper.

2 Securing IP Telephony Networks by Firewalls

2.1 Firewall Policies and Out-of-band Signaling

A "firewall" is one or a group of network elements enforcing an access control policy on the traffic at the border between network domains with different security levels and requirements. The access control and forwarding functions can be implemented on different layers of the IP protocol stack, e. g., in the application layer by so-called *proxies*. In contrast, *packet filters* are routers that decide whether to forward a packet by comparing header fields of IP and transport

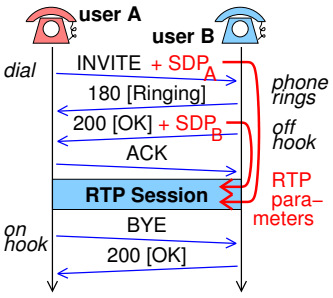

Fig. 1. Negotiation of RTP parameters by means of SIP/SDP

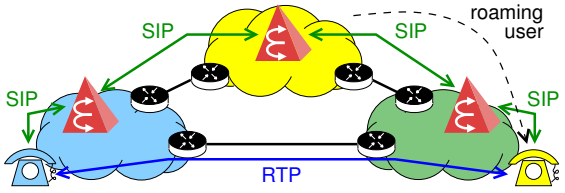

Fig. 2. Signaling messages and media streams may travel on different paths through the network

layers with their access control lists (ACL). Often, a policy with respect to application layer services can be implemented by simply comparing the transport layer destination port numbers with a *static* ACL, because of the *well-known port numbers* concept that most "traditional" Internet services follow.

However, some applications such as many VoIP solutions differ by using different protocols for the signaling and the transport of the actual user data (speech). In the considered scenario, the RTP (Realtime Protocol) media stream parameters such as codec and bit rate as well as IP addresses and UDP port numbers are *dynamically* negotiated using SDP (Session Description Protocol) messages embedded in the SIP signaling (Fig. 1). Therefore, a static packet filter configuration would either block all RTP streams or would have to allow all UDP traffic, rendering the firewall's protection almost useless [3]. As signaling messages and media streams may travel on different paths through the network (e. g., to roaming users, see Fig. 2), firewall traversal becomes even more difficult.

2.2 Architectures for Firewall Control – IETF MIDCOM

There are several approaches to solve the SIP/RTP firewall traversal problem [3]. A solution is to dynamically add rules to the packet filter by means of a signaling protocol. This can be done in two ways. *Path-coupled* firewall signaling such as the IETF NSIS architecture [RFC 4080] sends messages along the future media path, which request to open so-called "pinholes" in all packet filters on the path.

In contrast, when using *path-decoupled* signaling, SIP messages are sent via SIP entities such as back-to-back user agents (B2BUA, i. e., call-stateful SIP proxies), which control the packet filters on the media path. For the signaling between B2BUA and packet filter, the IETF MIDCOM (MIDdlebox COMmunication) architecture [RFC 3303] can be used. The term "middlebox" refers to the generalized concept of network elements that perform "functions other than the normal, standard functions of an IP router" [RFC 3234], such as packet filters and network address translators (NAT).

Fig. 3 shows one possible MIDCOM application scenario with SIP. The firewall protecting domain "A" 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

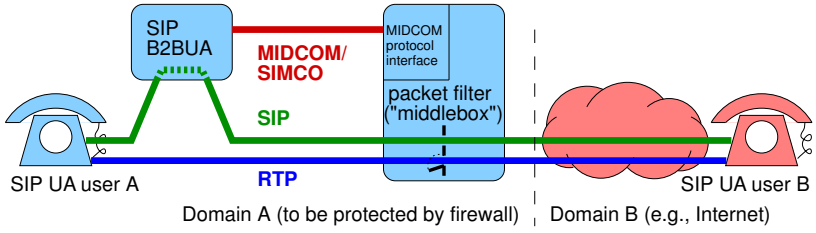

Fig. 3. The MIDCOM architecture

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 call duration. The interworking of SIP and MIDCOM signaling is illustrated by Fig. 4. A more detailed description including considerations of the behavior under error conditions can be found in [3].

2.3 SIMCO

MIDCOM is not a specific protocol but a framework architecture, including an abstract protocol semantics [RFC 3989] that can be implemented in several ways, e. g., by means of a suitably crafted SNMPv3 MIB. An alternative approach is the SIMCO (SImple Middlebox COnfiguration) protocol [4], a transaction based protocol using simple binary TLV 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 so-called "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. When a SIMCO agent asks the middlebox to establish a new policy rule, e. g., by means of a PER (Policy Enable Rule) request, the middlebox creates the

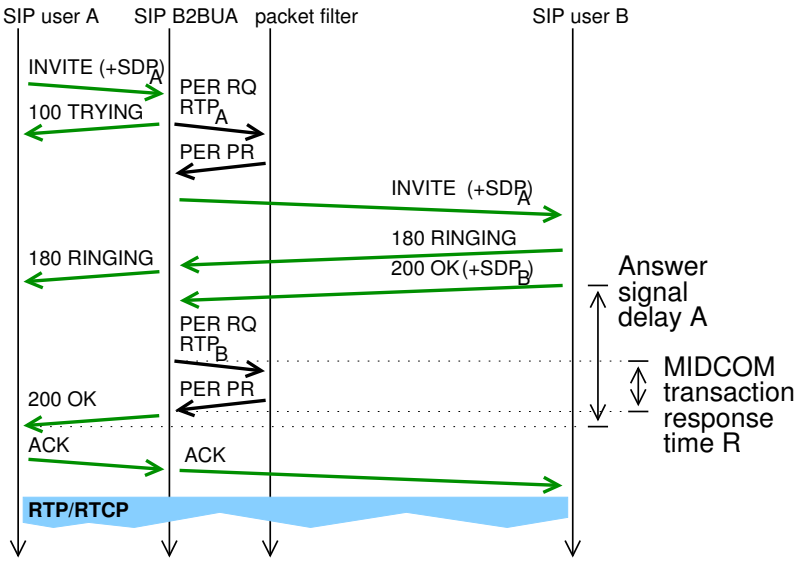

Fig. 4. Interworking of SIP and MIDCOM/SIMCO signaling

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 in later transactions, such as PLC (Policy Lifetime Change) for refreshing the softstate or deleting a rule (new lifetime = 0), to refer to this specific policy rule (Fig. 5).

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 [4] assumes that all transactions between an agent and a middlebox are transported in one SIMCO association over a single persistent TCP connection. Both are established in advance in order to avoid transaction delays caused by the TCP and SIMCO handshake.

As shown in Fig. 4, the response time R of the second PER transaction contributes to the answer signal delay A, which is inconvenient for the users and should therefore be minimized [5]. R consists of the local processing time in the middlebox plus the message transmission delay. In the remainder of the paper, we will focus on the latter effect and investigate in detail how to minimize the transmission delay by using SCTP as transport protocol for SIMCO.

3 Transport Protocols for Firewall Control

Traditionally, there have been two transport layer protocols in the Internet protocol suite. The Transmission Control Protocol (TCP) provides connection oriented, reliable transmission. It is the default transport protocol for SIMCO. The User Datagram Protocol (UDP) offers connectionless, unreliable transport and is therefore unsuitable for SIMCO. The Stream Control Transmission Protocol (SCTP) has been developed as a third transport layer protocol for IP, especially for signaling applications. It will be introduced briefly in the next section before its applicability for SIMCO will be investigated.

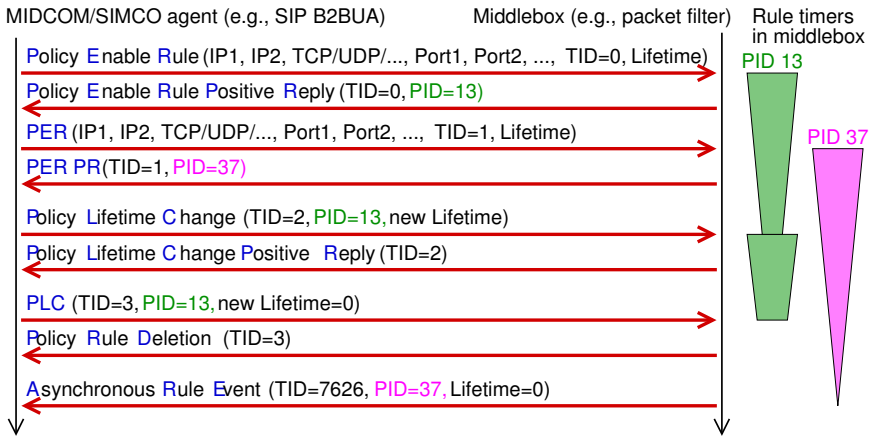

Fig. 5. SIMCO transactions create, modify and delete policy rules

3.1 Stream Control Transmission Protocol (SCTP)

SCTP [RFC 2960] has originally been designed as a part of the SIGTRAN architecture [RFC 2719] for the transport of SS7 [6] PSTN/ISDN telephony signaling over IP. While this rather special purpose is achieved by adaptation layers (e. g., M3UA [RFC 3332]) on top of it, SCTP itself has been designed as a generic transport protocol for IP networks, optimized for signaling applications. It has mechanisms for deployment in environments with high reliability and security requirements, such as "multihoming", i. e., support for having endpoints with several physical network interfaces as well as mechanisms to protect from denial-of-service and blind spoofing attacks [7].

With respect to user data transmission, SCTP provides a reliable datagram service. Similar to UDP, SCTP preserves the boundaries of upper layer protocol (ULP) messages. Therefore and different to TCP, no byte counters or frame delimiters are needed in the ULP. 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.

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. SCTP ensures in-order delivery within the same stream. 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 of other streams can still be delivered. This is illustrated in Fig. 6: Using SCTP, message #4 may be delivered to the ULP before message #3 has been retransmitted, as it is in another stream. Message ordering can even be disabled completely using the "unordered" flag.

Using one association split up into several streams – instead of using multiple associations bearing only one stream each – improves the efficiency of the TCP-like fast retransmit algorithm as it is applied on the aggregate message flow.

3.2 SIMCO over SCTP

When designing "SIMCO over SCTP", a very important problem is how to leverage SCTP's multiple streams feature, in order to reduce the impact of head-

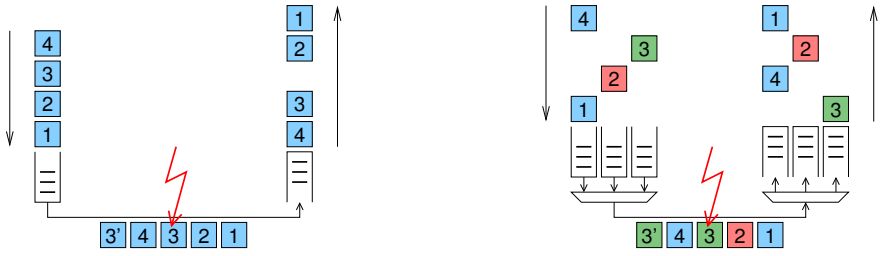

Fig. 6. Illustration of head-of-line blocking: one TCP connection (left) vs. one SCTP association with 3 streams (right)

of-line blocking. This can be further divided into two questions: 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. It is important that the transport layer protocol preserves the order of transactions that refer to the same policy rule. For example, it would be undesirable if a SIMCO agent requested a policy rule and immediately afterwards canceled it, but the delete message was delivered to the middlebox before the enable message. However, there is no requirement that prohibits reordering of SIMCO messages that refer to different policy rules.

Our basic idea is therefore to have several bidirectional stream pairs within the SCTP association. Two requirements shall be fulfilled: (1) All SIMCO messages that belong to one transaction shall be sent over the same pair. (2) All transactions that refer to the same policy rule shall be sent over the same pair.

As transactions always consist of a request sent by the agent and a reply sent by the middlebox, requirement (1) can be implemented in a stateless fashion: The middlebox sends replies on the same stream number as the corresponding request was received on. Requirement (2) can be fulfilled in the following way: Initially, the agent may use any strategy (e. g. round robin) for choosing the stream number on which a message requesting a new policy rule is to be sent. As described in Section 2.3, the middlebox assigns a unique PID to the new policy rule and returns it to the agent. The mapping from PID to stream number will be stored by the agent and used for sending all subsequent transactions that modify or delete this policy rule. The detailed specification of our approach, including special cases, can be found in [8].

4 A Model for Head-of-line Blocking in SCTP

In the following, we model the SIMCO response time when SCTP is used as transport protocol. We assume a scenario where the packet filter is located between two large domains, i. e., the busy hour call arrival rate is rather large.

4.1 Workload Model

As shown in Fig. 4, two pinholes are required to establish a call. The number of SIMCO transactions required to open, maintain, and close a pinhole depends on the call duration. According to Fig. 5, 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:

$$\lambda_{\mathrm{SIMCO}} = \lambda \cdot \int_0^\infty n(T) \cdot f(T)\ \mathrm{d}T\ . \tag{1}$$

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

4.2 Resequencing Delay over Multiple SCTP Streams

In this section, we model the effect of head-of-line blocking when several SCTP streams are used in parallel for data transmission, i. e., the SIMCO traffic is equally distributed over $N \geq 1$ streams. 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 for instance 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 acknowledgement for a DATA chunk arrives at the sender with probability $p_{\mathrm{S}} = (1-p)^2$. 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. Similar to the "fast retransmit" mechanism in TCP, an SCTP endpoint retransmits data when three subsequent SACKs include a missing report [9]. The reliable data delivery is also ensured by a timeout mechanism. However, this mechanism is usually only required if multiple packets get lost in sequence.

The SCTP error recovery by a fast retransmit is illustrated in Fig. 7. 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 supposed to be sent with constant IAT d. Furthermore, we assume that the sending window does not restrict the amount of DATA chunks sent, which is reasonable if the packet loss probability is small.

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

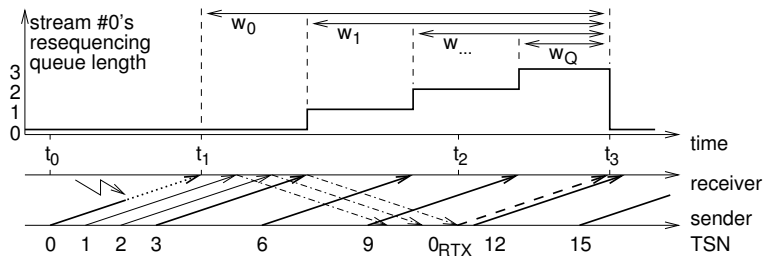

Fig. 7. Illustration of resequencing delays for 3 SCTP streams

$t_2 = t_0 + RTT + 3\,d$ 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.

The waiting times w_n of DATA chunks in the resequencing queue depend on the time $D = w_0$ to detect and recover from the packet loss. As shown by Fig. 7, the minimum value for D is $RTT + 3\,d$. 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 i DATA chunks have been sent, is $P(i) = p_{\mathrm{S}}^3\,(1-p_{\mathrm{S}})^{i-3}\,\binom{i-1}{i-3}$. From this follows

$$D \approx RTT + d\sum_{i=3}^{\infty} P(i)\,i = RTT + \frac{3\,d}{(1-p)^2}\ . \tag{2}$$

This expression is an approximation only since the retransmission may get lost, too. In this case, a retransmission timeout is required which may further enlarge the recovery period. 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.

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

$$W = p\sum_{i=0}^{Q} w_i = p\left((Q+1)\cdot D - \tfrac{Q\,(Q+1)}{2}\,N\,d\right)\ . \tag{3}$$

For bidirectional transactions as in the case of SIMCO, head-of-line blocking may occur in both directions and the mean response time thus follows as

$$R = RTT + 2\,W + \delta\ , \tag{4}$$

where δ represents the processing time in the end systems. As already mentioned, R must be small to minimize the answer signal delay perceived by users.

The remaining question is the optimal number of SCTP streams. Using a large number of streams may not be efficient since this may waste resources (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 fullfilled for $N \geq M$ with $M = \lceil \frac{D}{d} \rceil$. According to (2), M is quite insensitive to p. From this follows the optimal number of streams as

$$M \approx \left\lceil RTT \cdot \lambda \cdot \left(\tfrac{3}{2} + \tfrac{h}{L}\right) + 3 \right\rceil\ . \tag{5}$$

5 Performance Evaluation

5.1 Measurement Setup

In order to evaluate our "SIMCO over SCTP" specification [8], a prototype compliant to [4, 8] has been implemented [10]. As shown in Fig. 8, 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) [11, 3]. A load generator emulating user behavior has been implemented for measuring the SIMCO transaction response time (see Fig. 9).

The SIMCO software was implemented in C++ for Linux (kernel 2.6.11) and Solaris 10. The Linux version can use either the "lksctp"-kernel module or standard Linux TCP (using SACKs). For both protocols the "nodelay" socket options have been enabled. For Solaris, so far only TCP performance has been investigated. Measurements were made using two 2.4 GHz Pentium 4 or 500 MHz UltraSPARC IIe computers connected by 100 Mbps Ethernet to a network emulator, which adds a delay of $\Delta = 10\,\text{ms}$ in each direction and randomly drops IP packets with given probability p. The interaction of the middlebox software with the packet filter was disabled to isolate the measured delay from this overhead.

In the following we present the measurement results for one typical scenario with a large-scale softswitch. 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. Measurements with other parametrizations revealed similar results, which are documented in [12].

5.2 Measurement Results

Fig. 10 shows the mean SIMCO response time as a function of the packet loss probability p. All values have been obtained by averaging over the response time of PER requests during a measurement period of 1000 s after the load generator

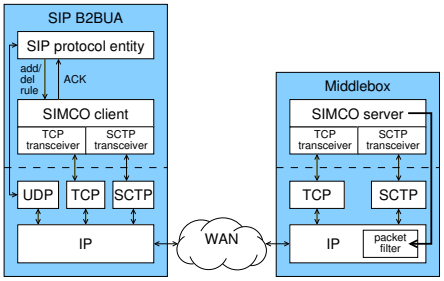

Fig. 8. SIMCO/SCTP prototype with B2BUA for proof of concept in SIP testbed

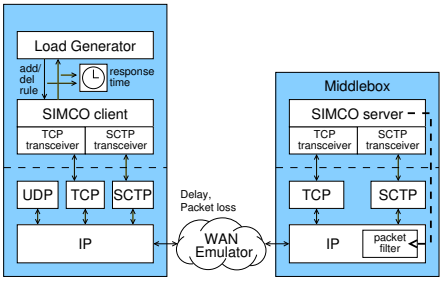

Fig. 9. SIMCO/SCTP testbed with load generator for performance measurements

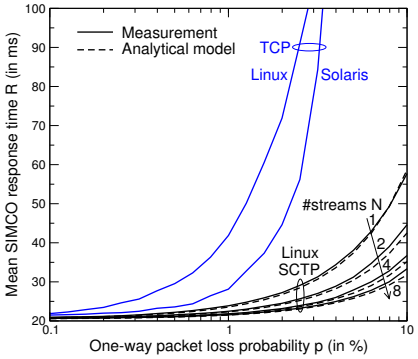

Fig. 10. Comparison of SCTP/TCP

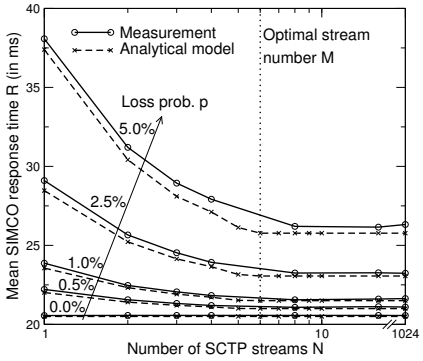

Fig. 11. Impact of SCTP streams no.

has reached the steady state. Fig. 10 reveals 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. 11 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 in eq. (4), 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. 11 also confirms that using a value N larger than the optimum value (here: $M = 6$) does not significantly improve performance.

Futhermore, Fig. 10 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. Even worse, TCP is not able to transport the offered load of about 100 transactions/s for packet loss probabilities larger than 7 %, which is manifested by socket buffer overflows. In particular the Linux TCP implementation, which is known to be highly optimized, performes quite bad even for packet loss probabilities much smaller than 1 %. Furthermore, there is a non-negligible probability for high call-setup delays. For example, the 99 % quantile of the SIMCO response time is ca. 100 ms for $p = 1\,\%$ if Linux TCP is used. Our analytical model for the head-of-line blocking does not explain this TCP-specific effect.

We have verified the measurements with a pair of simple test programs that use straightforward socket calls. These tests confirm the difference between TCP and SCTP. To the best of our knowledge, this effect has not been reported so far. An analysis of TCP traces shows that sometimes data arriving from the application layer is not immediately sent to the network. These additional delays could be caused by the TCP congestion control that reduces the sending window when facing packet loss. A more detailed insight into this effect can probably be obtained by means of simulation, but this is left for further study.

6 Conclusions

In this paper, we study the performance of the SIMCO protocol using SCTP and TCP as transport protocols. Being a typical signaling protocol, SIMCO can benefit from protocol mechanisms for high-reliability environments, which SCTP provides. Compared to TCP, SCTP can also significantly reduce the SIMCO response time by leveraging transmission over multiple streams, which reduces head-of-line blocking. We propose an analytical model to quantify this effect, 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 for transaction-based signaling causes significant delays even for small packet loss probabilities.

Acknowledgements

The authors would like to thank Christian Blankenhorn, Sebastian Beutel, and Thomas Ruschival for their help with the testbed and the measurements, as well as Martin Stiemerling and Michael Tüxen for valuable discussions and comments, especially on the IETF documents. Michael Scharf is funded by the German Research Foundation (DFG) through the Center of Excellence (SFB) 627.

References

1. G. Camarillo, R. Kantola, and H. Schulzrinne, "Evaluation of Transport Protocols for the Session Initiation Protocol," *IEEE Network*, vol. 17, no. 5, 2003.
2. H. Kamal, B. Penoff, and A. Wagner, "SCTP versus TCP for MPI," in *Proc. Supercomputing 2005*, Seattle, USA, Nov. 2005.
3. A. Müller and S. Kiesel, "Issues with the Interworking of Application Layer Protocols and the MIDCOM Architecture," in *Proc. Eunice Summer School*, 2004.
4. M. Stiemerling, J. Quittek, and C. Cadar, "Simple Middlebox Configuration (SIMCO) Protocol Version 3.0," IETF draft - work in progress, May 2005.
5. ITU-T Study Group 2, "Network grade of service parameters and target values for circuit-switched services in the evolving ISDN," ITU-T, Rec. E.721, May 1999.
6. ITU-T Study Group XI, "INTRODUCTION TO CCITT SIGNALLING SYSTEM No. 7," ITU-T, Recommendation Q.700, Mar. 1993.
7. S. Kiesel, "On the Use of Cryptographic Cookies for Transport Layer Connection Establishment," in *Proc. EUNICE Summer School*, 2002.
8. S. Kiesel, "SIMCO over SCTP," IETF draft - work in progress, Oct. 2005.
9. R. Stewart, "Stream Control Transmission Protocol (SCTP) Specification Errata and Issues," IETF draft - work in progress, Oct. 2005.
10. C. 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.
11. A. Müller, "Extension of a SIP proxy by security functions," Student project (in German), University of Stuttgart, IKR, 2004.
12. S. Kiesel, M. Scharf, S. Beutel, and T. Ruschival, "Performance Measurement Results of SIMCO over TCP and SCTP," University of Stuttgart, IKR, Internal Report 53, 2006.