Stream Control Transmission Protocol: Past, Current, and Future Standardization Activities

Thomas Dreibholz and Erwin P. Rathgeb, University of Duisburg-Essen
Irene Rüngeler, Robin Seggelmann and Michael Tüxen, Münster University of Applied Sciences
Randall R. Stewart, Huawei Technologies

ABSTRACT

The Stream Control Transmission Protocol (SCTP) is a general-purpose transport layer protocol providing a service similar to TCP — plus a set of advanced features to utilize the enhanced capabilities of modern IP networks and to support increased application requirements. Nowadays, there are SCTP implementations for all major operating systems. While SCTP was standardized as an RFC several years ago, there is still significant ongoing work within the IETF to discuss and standardize further features in the form of protocol extensions. In this article, we first introduce the SCTP base protocol and already standardized extensions. After that, we focus on the ongoing SCTP standardization progress in the IETF and give an overview of activities and challenges in the areas of security and concurrent multipath transport.

INTRODUCTION

Originally, the public switched telephone network (PSTN), based on Signaling System no. 7 (SS7), and the Internet, based on TCP/IP, were deployed separately. In the past few years, there is a continuous process to converge these distinct networks into a single network, requiring transport of SS7 over IP. Since TCP could not provide the necessary reliability for this task (e.g. strict failover times for emergency calls), the Stream Control Transmission Protocol (SCTP) [1] was standardized by the Internet Engineering Task Force (IETF) Signaling Transport (SIGTRAN) Working Group in 2000. After finishing the base protocol, the IETF Transport Area Working Group (TSVWG) became responsible for maintaining and extending SCTP. Beyond its original use case, the transport of SS7 over IP, SCTP is a general-purpose transport protocol, which provides the same service as TCP plus a set of advanced features regarding security, multihoming, multistreaming, mobility, and partial reliability. This article introduces SCTP and gives an overview of recent and ongoing IETF standardization activities for SCTP and its extensions.

Today, SCTP is not only the mandatory transport protocol for SS7-based PSTN signaling over IP networks, but also mandatory for the IETF Reliable Server Pooling (RSerPool) framework as well as optional for IP Flow Information Export (IPFIX) and Diameter.

BASE PROTOCOL

SCTP was initially specified in RFC 2960 in October 2000. Based on experience with implementations and research results, an updated specification was published in September 2007 as RFC 4960 [1].

GENERAL DESIGN PRINCIPLES

SCTP is a connection-oriented general-purpose transport protocol that preserves message boundaries. An SCTP connection, called an SCTP association, can be used on top of IPv4 and IPv6. One of the main design goals was the efficient transport of small messages in a network fault-tolerant way, important for transporting signaling messages. Another feature is TCP fairness. SCTP should behave in a fair way when it competes with TCP traffic. This is important for deploying SCTP in networks with TCP-based traffic. Therefore, SCTP adopts the congestion and flow control from TCP, which is described in more detail later.

An SCTP packet consists of a common header and a number of chunks. The common header has a fixed length and contains port numbers, a verification tag, and a checksum. SCTP uses the same port number concept as UDP or TCP. The verification tag is used to protect an association against a blind attacker. It is a random number chosen at association setup and must be present in each message of the association. Thus, a blind attacker has to guess this number. The checksum is a 32-bit cyclic redundancy check (CRC32C), which protects the packet against corruption. It

Parts of this work have been funded by the German Research Foundation (Deutsche Forschungsgemeinschaft — DFG).

0163-6804/11/$25.00 © 2011 IEEE

is much stronger than the checksum used for UDP and TCP. Modern Ethernet cards provide hardware support for the computation of the CRC32C in the SCTP packet.

A chunk has a variable length and consists of a type, some flags, a length field, the actual value, and possibly padding, which ensures that the total length in bytes of a chunk is a multiple of four. Since this generic format is used by all chunks, a receiver can parse a received packet even if it does not support some of the received chunk types. Since how to handle unknown chunks is also defined, the packet format is extensible. User messages are put into DATA-chunks, and the other chunks, so-called control chunks, are used for SCTP control information.

Small user messages are put into their own DATA-chunk, and multiple DATA-chunks are put into one packet. This so-called bundling of user messages allows the sending of multiple messages within one SCTP packet.

SCTP also supports the fragmentation and reassembly of user messages. If a user message is too large to fit into a single link layer packet, it is fragmented, and several DATA-chunks are sent in different packets to the receiver, which finally reassembles the user message and delivers it to the user.

Association Setup and Teardown

An SCTP association is typically set up using a four-way handshake. The first packet containing an INIT-chunk is responded with a packet containing an INIT-ACK-chunk. It is important that the receiver of the INIT-chunk does not change any state or reserve any resources. Instead it puts all required information in a cookie, which is part of the INIT-ACK-chunk. The receiver of the INIT-ACK-chunk extracts this cookie and sends it back in a packet containing a COOKIE-chunk. Finally a COOKIE-ACK-chunk finishes the four-way handshake. The use of this handshake makes the server resilient against flooding attacks. This handshake is protected against message loss by using a timer-based retransmission scheme. During this handshake several parameters are negotiated: the verification tags requested by each endpoint, the addresses used by each endpoint, the number of streams, the supported protocol extensions, and so on. The format of the INIT- and INIT-ACK-chunks is also extensible.

An SCTP association is normally terminated by a three-message exchange based on the SHUTDOWN-, SHUTDOWN-ACK-, and SHUTDOWN-COMPLETE-chunk. This is the reliable form of an association teardown. It ensures that all user messages are received. In situations where this is not possible, an SCTP packet with an ABORT-chunk is sent, which terminates the association immediately, taking message loss into account.

Multihoming

During association setup, each SCTP endpoint provides a list of its addresses in the INIT- and INIT-ACK-chunk to the peer. For security reasons each endpoint first has to verify that the remote addresses really belong to the peer. This path verification uses so-called HEARTBEAT- and HEARTBEAT-ACK-chunks. These chunks are also used to monitor remote addresses when no user traffic is sent to them to check their reachability. The base protocol uses the multiple remote addresses only for redundancy. This means that it typically sends all user messages to one remote address, the so-called primary address (Fig. 1). If messages have to be retransmitted due to timeouts, other remote addresses are used for the retransmission. After a number of consecutive message losses for a particular remote address, this address is considered unreachable and is not used for user message transport any more until it is reachable again.

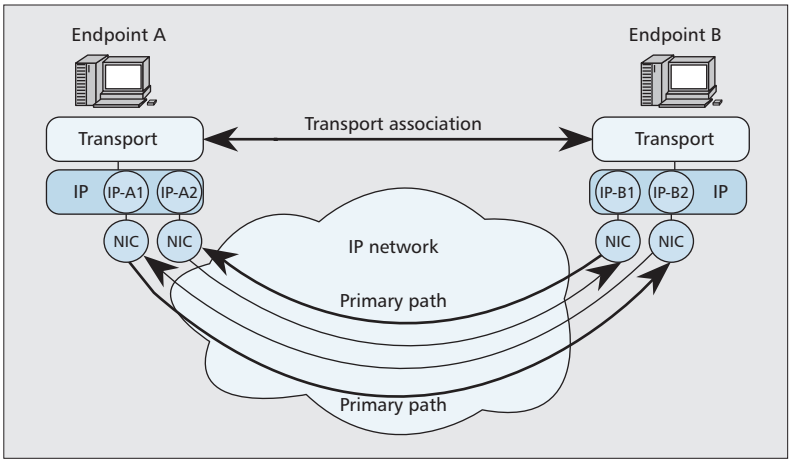

Figure 1. *The principle of multihoming.*

The simultaneous use of multiple paths for data transfer was a feature considered during the initial phase of the SCTP specification. However, there was no known way to realize this in a TCP-friendly way at that time. This has changed recently, and the extension of SCTP to support this is described later.

Multistreaming

An SCTP association has a number of unidirectional channels in each direction. Such a channel is called a stream. The number in both directions does not need to be the same and can vary between 1 and $2^{16} - 1$. It is negotiated during association setup. SCTP provides in-sequence delivery for messages within each stream, but not across different streams. If a message of a particular stream is lost, messages of the other streams do not need to be delayed at the receiver until the lost message has been retransmitted and finally received. Therefore, multiple streams can be used to minimize so-called head-of-line blocking. Multiple DATA-chunks containing messages from the same or different streams can be bundled into a single packet.

An SCTP user can also indicate that a particular message does not require any ordering.

Standardized Protocol Extensions

Two protocol features were omitted in the base protocol specification of SCTP to ensure that the RFC was published on time: partial reliability and the reconfiguration of IP addresses being

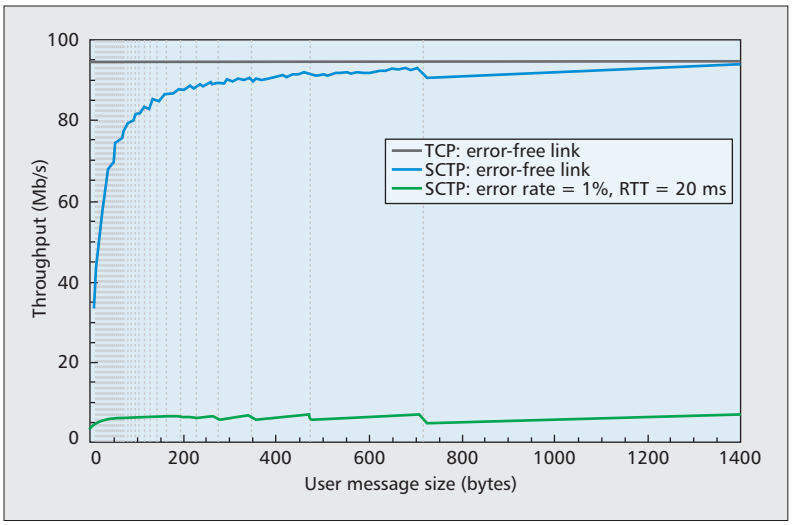

Figure 2. *Maximum throughput for SCTP and TCP.*

used during the lifetime of an association. These two protocol extensions were standardized by the IETF in separate RFCs and one additional RFC was published to define a mechanism providing the necessary security mechanism for dynamic address reconfiguration. In the following subsections these three extensions are described.

Partial Reliability

The Partial Reliability extension (PR-SCTP) is defined in RFC 3758 [2] and enables the SCTP sender of a message to decide that a user message is not transmitted or retransmitted any more. Such a message is called abandoned. A specific method of abandoning messages is called a PR-SCTP policy. An important feature of PR-SCTP is that the receiver is policy agnostic. Only the sender implements a particular policy. This allows an implementer to deploy new policies easily, since only sender side modifications are necessary.

RFC 3758 defines only one PR-SCTP policy, which limits the lifetime of the user message. After a lifetime specified by the application in milliseconds for the message, the user message is abandoned. This PR-SCTP policy can be used if the application layer also runs timers for messages, which is quite common for telephony signaling messages to avoid multiple timers for a single message.

There are two additional PR-SCTP policies currently implemented in FreeBSD. One policy is priority-based. If there is no buffer space available at the sender when sending a message with a given priority, other user messages with lower priorities are abandoned to make enough buffer space available to accept the user message. This policy is used for IPFIX, where different message classes are used. The other PR-SCTP policy limits the number of retransmissions. One extreme example of this PR-SCTP policy is avoiding all retransmissions by limiting the number of retransmissions to zero. This can be used, for example, to transmit multimedia data or to tunnel UDP traffic.

Dynamic Address Reconfiguration

Each SCTP endpoint provides all IP-addresses it is using to its peer during association setup. The base protocol does not allow these addresses to be changed during the lifetime of the association. To overcome this limitation, a protocol extension called Dynamic Address Reconfiguration was specified in RFC 5061 [3]. This extension allows an SCTP endpoint to add or delete a local address and notify its peer to use a specific local address as the primary path.

Thus, it is possible to reconfigure a host by adding or removing network interface cards or changing the IP configuration without disturbing existing SCTP associations. This is important for providing long living SCTP associations and was the original motivation to add this feature.

It should be noted that there are two other important usage scenarios for dynamic address reconfiguration. One is the possibility of supporting mobility at the transport layer. Transport layer mobility allows a transport connection to continue even when some IP-addresses of a host change. The other useful application is setting up an SCTP association as single homed and then adding each additional local address one by one using dynamic address reconfiguration. Hence, it is possible to avoid IP-addresses being part of the SCTP packet, a feature making an SCTP-aware Network Address Translation (NAT) traversal possible.

One of the requirements for this extension was not to introduce any additional security issues to SCTP associations. To ensure this, another extension was necessary, which is introduced in the following subsection.

SCTP Authentication

RFC 4895 [4] defines an extension called SCTP-AUTH. It allows a set of chunks to be protected by using a keyed hash. The key consists of a shared key and a shared secret. The shared key is negotiated during the association setup and transmitted in plaintext. The shared secret must be known by the two SCTP endpoints and is never transmitted. It might be empty. The management of this shared secret is out of the scope of RFC 4895.

The messages for adding and deleting IP addresses using the dynamic address reconfiguration extension are protected by SCTP-AUTH.

SCTP-AUTH also supports negotiation of the hash algorithm and multiple shared secrets. This is important for another use case for this extension: securing SCTP traffic using transport layer security. This is described in detail later.

Performance Impact of Message-Oriented Transport

One of the major requirements when designing SCTP was the restriction that it had to be fair toward TCP. Fairness in this context means the equal sharing of the available bandwidth on all connections. The throughput, defined as the transmitted amount of data per time, is a suitable means of comparison. As TCP is a byte-stream oriented protocol, the throughput for a given network scenario is a constant provided there is always enough data to be sent.

SCTP Performance under Ideal Conditions

The throughput behavior of SCTP differs from TCP due to its message orientation. If small mes-

sages are bundled, the overhead of the DATA-chunk header can even occupy the better part of a packet. Therefore, the maximum IP packet size fitting into a link layer packet (maximum transmission unit [MTU]), the average user message size (UMS), the number of chunks per packet (CPP), the various header sizes, and even the number of padding bytes have to be taken into account when calculating the throughput.

Figure 2 compares the theoretical throughput for TCP and SCTP on a 100 Mb/s link. The SCTP throughput increases with the message size (i.e., with the decrease of the header to payload ratio). The pronounced steps indicated by vertical lines are caused by the bundling boundaries, meaning that k DATA-chunks with a UMS of N bytes can be assembled into one packet but only $k - 1$ or less of size $N + 1$ bytes. Also noticeable is the zigzagging of the graph, which is caused by the padding bytes.

Performance on Lossy Links

If packets are lost and have to be retransmitted, the packet error rate P_P and round-trip time (RTT) have to be taken into account. Again, in the case of SCTP, its message orientation requires the consideration of the user message size and the bundling of small messages. As shown in [5], this leads to

$$Throughput = \frac{\sqrt{UMS \cdot CPP \cdot MTU} \cdot \sqrt{\frac{3}{2}}}{RTT \cdot \sqrt{P_P}}$$

Figure 2 also shows the throughput on a 100 Mb/s link with an error rate of 1 percent and an RTT of 20 ms. This theoretical result has been validated in [5] by using our SCTP simulation model, which has been included in the INET distribution.

Implementing Congestion and Flow Control for SCTP

As mentioned earlier, SCTP has adopted the mechanisms for congestion and flow control from TCP. Yet again, the message orientation of SCTP has significant impact on the implementation of the algorithms.

In the case of congestion control, the amount of data that may be sent is calculated as the difference between the congestion window and the number of outstanding bytes (i.e., the data that has been sent, but not yet acknowledged). The way these outstanding bytes are counted (i.e., whether the DATA-chunk header is included or not) is not specified in RFC 4960 [1]. In [6], we analyzed the impact of these options on the fairness toward TCP. Simulation results showed that the DATA-chunk headers definitely have to be considered to avoid unfairness toward TCP.

Comparing the flow control behavior of various implementations showed that the sending of small messages might exhaust the receiver window before the advertised receiver window in the SACK-chunk is reduced to zero, which results in spurious retransmissions. This is caused by the storing of additional information for each incoming DATA-chunk, which is not announced in the advertised receiver window. Reference [6] shows that this discrepancy can be overcome by "telling the truth," that is, matching the size of the advertised receiver window with the real receive buffer.

Both recommendations in this subsection can be applied without changing RFC 4960.

Security

Several approaches to provide authentication and encryption for SCTP have been introduced. The first was RFC 3436, describing the use of Transport Layer Security (TLS) over SCTP. Despite TLS originally being developed for TCP, it can also be used with SCTP, although there are some limitations since TLS requires a reliable and in-order transfer. This requires a TLS connection per bidirectional pair of streams with ordered transfer, while unordered transfer as well as PR-SCTP cannot be used.

Internet Protocol Security (IPsec) is also a possible solution to secure SCTP. Standard implementations of IPsec should be able to handle SCTP just as they do other transport protocols. A modification, as described in RFC 3554, is required for SCTP's multihoming features, because the Security Policy Database (SPD) is based on tuples of port numbers and addresses of which an SCTP association would have multiple, one for every path.

There have been other suggestions as well, like secure SCTP and SS-SCTP, but due to difficulties in practical realization, detailed in [7], the standardization of these approaches is rather unlikely.

The latest approach, the SCTP-aware Datagram Transport Layer Security (DTLS), is specified in RFC 6083 [8]. DTLS is an adaptation of TLS for unreliable transport protocols. This includes changes in the calculation of the hash-based message authentication code (HMAC) allowing each message to be verified independently, and thus tolerate message loss and reordering. Therefore, contrary to TLS over SCTP, with DTLS multiple streams as well as unordered transfer and PR-SCTP are usable. However, some adaptations are still necessary, because DTLS cannot protect SCTP's control chunks, which would be a target for attacks. An attacker could tamper with headers to modify the order of messages or use fake PR-SCTP information to intercept and drop packets unnoticed. To avoid this, SCTP-AUTH has to be used to assure the integrity of exposed information, that is, the DATA chunks containing the DTLS messages and the FORWARD-TSN chunks of PR-SCTP [7].

In Fig. 3 the DTLS over SCTP performance over a gigabit link is compared to TLS over TCP as a reference, by measuring the throughput with an increasing message size. DTLS over SCTP is slower than TLS, because of the additional HMAC calculation with SCTP-AUTH. Since this HMAC also covers the DTLS messages, there is no need for DTLS to calculate its own. So by specifying a cipher suite that does not include an HMAC, the performance can be optimized, and DTLS over SCTP is almost as fast as TLS over SCTP. The remaining gap is just the result of a slower implementation of the HMAC

One of the major requirements when designing SCTP was the restriction that it had to be fair toward TCP. Fairness in this context means the equal sharing of the available bandwidth on all connections. The throughput, defined as the transmitted amount of data per time, is a suitable means of comparison.

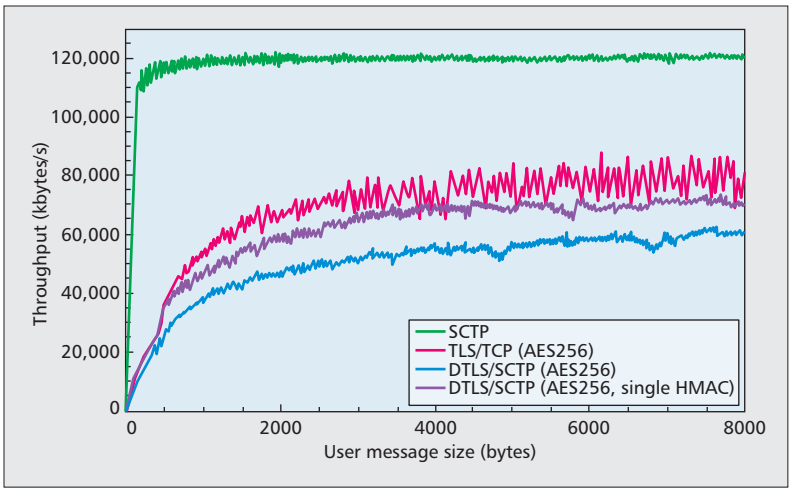

Figure 3. *SCTP-aware DTLS performance comparison.*

algorithm for SCTP-AUTH in the SCTP kernel used for this measurement, compared to the highly optimized implementation of OpenSSL. The gap between the lower curves and the uppermost one shows the performance penalty for using encryption and authentication.

Current Standardization Activities

Currently there are three work items covered by working group documents in the IETF.

Network Address Translation for SCTP

To achieve broad deployment, SCTP's rich features must be usable across all network boundaries. Network Address Translation is the standard method for separating private networks from global networks by translating private IP-addresses to global IP-addresses. NAT has been deployed for TCP and UDP, but SCTP as a fairly new protocol is not supported. The only option to date is the use of a TCP-like NAT for SCTP, which reduces SCTP to the single-homed case, or applying UDP encapsulation, which again has its drawbacks. To take advantage of all SCTP features, a specialized NAT for SCTP has to be deployed. In [9] and [10], we introduce NAT for SCTP that supports multihoming and can be applied in peer-to-peer communication.

Stream Reset

Several groups using SCTP have requested to be able to switch back individual SCTP streams to the state they had directly after association setup or to increase the number of streams. An extension called STREAM-RST has been developed to provide this functionality and is specified in the draft [11].

SCTP Socket API

SCTP as a transport protocol can only be used if the application can communicate with it. This is usually performed via sockets. In [12], the powerful SCTP socket application programming interface (API) is introduced, which provides commands not only to send and receive data but also to change and retrieve the values of important parameters like send or receive buffers, RTO, and HEARTBEAT intervals. Thus, the user can tailor an application according to his/her needs.

Ongoing Development

In addition to the current standardization work described earlier, there is further ongoing research and development. While these activities are still in an early stage of IETF discussion, they will become important in the standardization context in the near future, and are therefore briefly introduced here.

SCTP as defined in [1] only utilizes the primary path in each direction for data transmission. All other paths are only used for retransmissions and constitute backups to handle primary path failures. However, the growing number of multihomed Internet sites leads to a rising demand to utilize all paths *simultaneously*. This simultaneous path usage is commonly referred to as Concurrent Multipath Transfer (CMT). The CMT-SCTP protocol extension [13] adds CMT support to SCTP. While it works well for paths having similar bandwidths, delays and error rates, the existence of dissimilar ones — which are very likely for Internet setups involving different Internet service providers (ISPs) — is challenging.

For example, the SCTP unordered payload throughput of a scenario with dissimilar paths is depicted in Fig. 4. The setup consists of two paths, A and B, path A with a bandwidth of ρ_A = 10 Mb/s and path B with a bandwidth of ρ_B varying from 0.1 to 20 Mb/s; 100-kbyte send buffer, 50-kbyte receive buffer, and 1 ms delay on each path are assumed. The results show the average of 24 OMNeT++/INET simulation runs. Clearly, standard SCTP achieves a throughput of around ρ_A = 10 Mb/s for using primary path A (curve 1) or ρ_B Mb/s for using primary path B (curve 2). For plain CMT-SCTP, the expected throughput of around 20 Mb/s is reached at ρ_B = 10 Mb/s (curve 3; i.e., in the case of similar paths), but the performance highly differs from the expected rate in dissimilar cases. To reach the expected case (i.e. scaling linearly with ρ_B), two optimizations are necessary:

- Non-revocable selective acknowledgments (NR-SACK) [14] allow a receiver to non-revokably gap-acknowledge incoming chunks. That is, the sender can remove them from its buffer, although they are not yet covered by a cumulative acknowledgment. This leads to a significant reduction of the required send buffer size, mitigating transmission blocking due to a full buffer.
- Buffer splitting [15] subdivides send and receive buffers into per-path sections. This mechanism solves blocking issues where certain paths can block the removal of chunks due to waiting for a retransmission.

When applying both mechanisms, the achieved payload throughput meets the expectation of $\rho_A + \rho_B$ Mb/s (curve 4).

From the congestion control perspective, CMT-SCTP handles each path like an independent TCP-like flow. While this is useful under

the assumption that all paths are disjoint, it introduces unfairness to competing non-CMT flows when multiple paths share a common bottleneck link. Since the fundamental property of IP-based networks is connection-less routing, such bottlenecks cannot be detected reliably in arbitrary networks (like the Internet). The CMT/RP-SCTP extension combines CMT-SCTP with so-called resource pooling (RP) by handling multiple paths like one big path and applying congestion control accordingly. That is, the congestion window of a path *P* is adapted with the additive increase, multiplicative decrease (AIMD) behavior of TCP-like congestion control, but proportionally to *P*'s share on the total capacity of all paths. A path's capacity is indicated by its current slow start threshold. Multipath TCP applies a similar approach.

A performance improvement on path failures can be achieved by introducing a potentially failed (PF) [16] state to the path management. When a path is considered unreliable, it enters the PF state and is not used for new data transmissions, thus avoiding a performance degradation until the failed path is actually removed by timeout.

NR-SACK [14] and PF [16] as well as CMT-SCTP, CMT/RP-SCTP, and the necessary buffer handling strategies [17] are currently still in the process of IETF standardization.

Implementations and Supported Features

Table 1 provides an overview of the major SCTP implementations for Linux (mainline kernel 2.6.36), FreeBSD (release kernel 8.1[2]), MacOS X,[3] Windows (SctpDrv[4]) and Solaris (OpenSolaris 2009.06) as well as sctplib[5] (a well-known userland SCTP implementation), and the OMNeT++/INET simulation model[6] (providing an external interface to the real world) with their supported features as of October 2010.

Conclusions

In this article, we have given an overview of the recent advances in the IETF standardization process of the SCTP protocol and its extensions. While the core SCTP protocol, including the extensions for partial reliability, chunk authentication, and dynamic address reconfiguration, already completed standardization within the IETF SIGTRAN WG and TSVWG some time ago, there is still a significant amount of ongoing work within the TSVWG and BEHAVE WG to standardize further enhancements like SACK immediately, stream reset, the socket API, and an SCTP-aware NAT. Also, there is clear interest in CMT with SCTP, which is currently under development by multiple research groups and is expected to dominate SCTP standardization activities in the coming years.

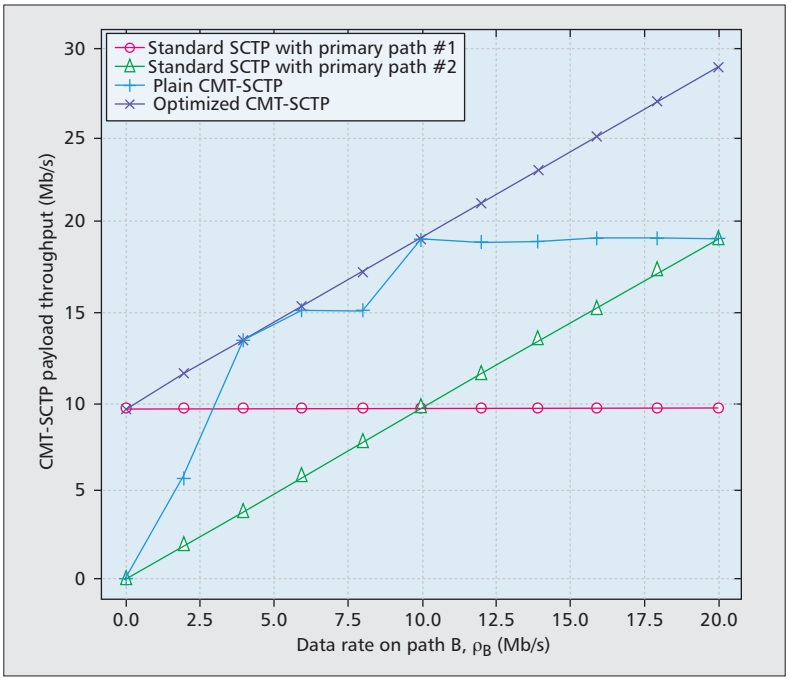

Figure 4. *Throughput of CMT-SCTP transport over dissimilar paths.*

References

[1] R. Stewart, "Stream Control Transmission Protocol," IETF RFC 4960, Sept. 2007.

[2] R. Stewart *et al.*, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension," IETF RFC 3758, May 2004.

[3] R. Stewart *et al.*, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration," IETF RFC 5061, Sept. 2007.

[4] M. Tüxen *et al.*, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)," IETF RFC 4895, Aug. 2007.

[5] I. Rüngeler, *SCTP — Evaluating, Improving and Extending the Protocol for Broader Deployment*, Dissertation, Univ. of Duisburg-Essen, Faculty of Econ., Inst. for Comp. Sci. and Business Info. Sys., Dec. 2009.

[6] I. Rüngeler, M. Tüxen, and E. P. Rathgeb, "Congestion and Flow Control in the Context of the Message-Oriented Protocol SCTP," *Proc. 8th Int'l. IFIP Net. Conf.*, Aachen, Germany, 2009, pp. 468–81.

[7] R. Seggelmann, M. Tüxen, and E. P. Rathgeb, "Design and Implementation of SCTP-aware DTLS," *Proc. Int'l. Conf. Telecommun. and Multimedia*, July 2010.

[8] M. Tüxen, R. Seggelmann, and E. Rescorla, "Datagram Transport Layer Security for Stream Control Transmission Protocol," IETF RFC 6083, Jan. 2011.

[9] M. Tüxen *et al.*, "Network Address Translation for the Stream Control Transmission Protocol," *IEEE Network*, vol. 22, no 5, 2008, pp. 26–32.

[10] R. Stewart, M. Tüxen, and I. Rüngeler, "Stream Control Transmission Protocol (SCTP) Network Address Translation," draft-ietf-behave-sctpnat-04.txt, Dec. 2010, work in progress.

[11] R. Stewart, P. Lei, and M. Tüxen, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration," draft-ietf-tsvwg-sctp-strrst-09.txt, Nov. 2010, work in progress.

[12] R. Stewart *et al.*, "Sockets API Extensions for Stream Control Transmission Protocol (SCTP)," draft-ietf-tsvwg-sctpsocket-27.txt, Jan. 2011, work in progress.

[13] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths," *IEEE/ACM Trans. Net.*, vol. 14, no 5, Oct. 2006, pp. 951–64.

[14] P. Natarajan *et al.*, "Non-Renegable Selective Acknowledgments (NR-SACKs) for SCTP," draft-natarajan-tsvwg-sctp-nrsack-06.txt, Aug. 2010, work in progress.

[15] T. Dreibholz *et al.*, "On the Use of Concurrent Multipath Transfer over Asymmetric Paths," *Proc. IEEE GLOBECOM*, Miami, FL, Dec. 2010.

[16] Y. Nishida and P. Natarajan, "Quick Failover Algorithm in SCTP," draft-nishida-tsvwg-sctp-failover-02.txt, Dec. 2010, work in progress.

[17] M. Becke *et al.*, "Load Sharing for the Stream Control Transmission Protocol (SCTP)," draft-tuexen-tsvwg-sctp-multipath-01.txt, Dec. 2010, work in progress.

[2] Patches available for "in progress" features.

[3] http://sctp.fh-muenster.de/sctp-nke.html, not officially supported by Apple; patches available for "in progress" features.

[4] http://www.bluestop.org/SctpDrv/, not officially supported by Microsoft.

[5] http://www.sctp.de/sctp.html, stable release 1.0.9; "experimental" features only available in experimental versions.

[6] INET SCTP development version, to be released soon.

Feature	Linux	FreeBSD	MacOS X	Windows	Solaris	sctplib	OMNeT++
Standard SCTP (RFC 4960)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Explicit congestion notif. (ECN)	Yes	Yes	Yes	Yes	No	No	No
Partial reliability (PR-SCTP)	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Chunk authentication (SCTP-AUTH)	Yes	Yes	Yes	Yes	Yes	No	Yes
Dynamic address reconfiguration	Yes	Yes	Yes	Yes	Yes	Experimental	Yes
Packet drop rreporting (PKTDROP)	No	Yes	Yes	Yes	No	No	Yes
SACK immediately	No	Yes	Yes	Yes	No	No	Yes
Advanced stream schedulers	No	In progress	In progress	In progress	No	No	Yes
Stream reset	No	Yes	Yes	Yes	No	No	Yes
Non-revokable SACKs (NR-SACK)	No	Yes	Yes	Yes	No	No	Yes
Potentially failed (PF) path state	No	Yes	Yes	Yes	No	No	No
CMT-SCTP	No	Yes	Yes	Yes	No	No	Yes
CMT/RP-SCTP	No	In progress	In progress	In progress	No	No	Yes
Buffer splitting	No	In progress	In progress	In progress	No	No	Yes
Chunk rescheduling	No	In progress	In progress	In progress	No	No	Yes
Secure-SCTP	No	No	No	No	No	Experimental	No
Checksum offloading	Yes	Yes	No	No	Yes	No	(not useful)
Socket API	Yes	Yes	Yes	Yes	Mostly	Yes	(not useful)

Table 1. *SCTP implementations and supported features.*

BIOGRAPHIES

THOMAS DREIBHOLZ (dreibh@iem.uni-due.de) is an assistant professor in the Computer Networking Technology group at the Institute for Experimental Mathematics, University of Duisburg-Essen, Germany. Currently, his main research topics are SCTP and the RSerPool framework. He is the author of various research papers at international conferences and in journals. Furthermore, he contributed multiple Working Group and Individual Submission Drafts to the IETF RSerPool and TSVWG Working Groups' standardization processes. He is also an author of multiple IETF RFC documents.

IRENE RÜNGELER (i.ruengeler@fh-muenster.de) received her diploma in computer science and economics at the University of Hagen in 1992 and 2000, respectively. She joined Münster University of Applied Sciences in 2002 where she works as research staff member. Her research interests include innovative transport protocols, especially SCTP, and their performance analysis, signaling transport over IP-based networks, and fault-tolerant systems. In 2009 she received her Ph.D. (Dr.rer.nat) from the University of Duisburg-Essen, Germany.

ROBIN SEGGELMANN (seggelmann@fh-muenster.de) received his Master's degree in 2008 from Münster University of Applied Sciences, where he has continued working as a research assistant. His main work is the optimization of SCTP's streaming features, their use in common applications, as well as security and mobility with DTLS. Since 2009 he is also involved in authoring standardization documents regarding these topics at the IETF.

MICHAEL TÜXEN (tuexen@fh-muenster.de) studied mathematics at the University of Göttingen and received his Dipl.Math. degree in 1993 and Dr.rer.nat. degree in 1996. In 1997 he joined the Systems Engineering group of ICN WN CS of Siemens AG, Munich. Since 2003 he is a professor in the Department of Electrical Engineering and Computer Science, Münster University of Applied Sciences. In the IETF he is participating in the Working Groups Signaling Transport (SIGTRAN), Reliable Server Pooling (RSerPool), and Transport Area (TSVWG). His research interests include innovative transport protocols, especially SCTP, IP-based networks, and highly available systems.

ERWIN P. RATHGEB [SM] (rathgeb@iem.uni-due.de) holds the Alfried Krupp von Bohlen und Halbach-Chair for Computer Networking Technology at the Institute for Experimental Mathematics, University of Duisburg-Essen, Germany. He is the author of a book on ATM and has published more than 50 papers in journals and at international conferences. He is a member of GI, IFIP, and ITG, where he is chairman of the expert group on network security. His current research interests include network security as well as concepts and protocols for next-generation Internet.

RANDALL STEWART (rstewart@huawei.com) currently works for Huawei Technologies Inc. as a Distinguished Engineer. His current duties include architecting, designing, and prototyping Huawei's next generation routing and switching platform. Previously he was a Distinguished Engineer at Cisco Systems. In other lives he has also worked for Motorola, NYNEX S&T, Nortel, and AT&T Communication. Throughout his career he has focused on operating system development, fault tolerance, and call control signaling protocols. He is also a FreeBSD committer having responsibility for the SCTP reference implementation within FreeBSD.