# A Novel Method for SCTP Load Sharing

Andreas Jungmaier and Erwin P. Rathgeb

Computer Networking Technology Group,
IEM, University of Duisburg-Essen
Ellernstr. 29, 45326 Essen, Germany
{ajung,rathgeb}@exp-math.uni-essen.de

**Abstract.** SCTP is a general purpose transport protocol featuring multi-homing support and flexible, message oriented data delivery services. With respect to multi-homing, the SCTP standard uses this feature for network level redundancy only and, thus, does not provide load sharing capabilities among the redundant links. In order to satisfy the strict performance requirements for signaling transport, efficient load sharing among all active links is desirable in certain scenarios. Therefore, some extensions providing load sharing functionality have been proposed. In this paper, we suggest an improved load sharing algorithm for SCTP with path based selective acknowledgements and present results of a simulation study to demonstrate the benefits of our algorithm.

**Keywords**

SCTP, transport protocol, multi-homing, load sharing

## 1 An Overview of the Stream Control Transmission Protocol

Designed by the Signaling Transport Working Group of the IETF in particular for the transport of signaling data, the Stream Control Transmission Protocol (SCTP) [1] is a general purpose, message-oriented, reliable transport protocol providing a more flexible data delivery service than TCP by using a specific SCTP stream layer, and an increased fault tolerance by allowing network level redundancy through multi-homing. Multi-homed endpoints can be reached by a set of transport addresses rather than only one transport address, therefore multiple distinct paths may exist from an endpoint to its peer endpoint.

SCTP packets consist of a common header, followed by a variable number of information units, which are named *chunks*. There are two types of chunks: control and data chunks. Data chunks contain the actual user messages, while several types of control chunks are used to support the peer-to-peer protocol. The reliable transmission of user data involves data chunks with a 32 bit sequence number (TSN), selective acknowledgement (SACK) control chunks, timers, and multiple selective repeat mechanisms. The amount of data that may be outstanding (i.e. sent but not yet received) is limited by a receiver window regularly announced in the SACK. Flow control parameters (congestion window, *cwnd*, and

the slow start threshold, *ssthresh*) are computed for each network path to the peer. Similarly to TCP, SCTP uses an AIMD algorithm for each path to avoid congestion [1].

The main traffic load is carried by one path only which is the *primary path*. Other paths are only used for data retransmissions and heartbeat control chunks (to monitor the availability of a path). The application may, however, explicitly request to use a path other than the *primary* path for data transmission.

## 2 A Novel Method for SCTP Load Sharing

SCTP load sharing may be thought of as a periodic change of the primary path. When such changeovers are periodically triggered, significant reordering can be observed by the receiver [2], which is reported to the sender in SACKs containing gap reports [1]. Standard SCTP reacts with unnecessary fast retransmissions, reducing the *cwnd* which unnecessarily limits the overall throughput. Furthermore, standard SCTP only increases the *cwnd* for a path when an incoming SACK from this path advances the highest cumulative TSN acknowledged so far (CTSNA). Therefore, the growth of the *cwnd* of standard SCTP is not adapted to load sharing, happens too slowly and typically mostly for the slowest path. When chunks are delivered to the receiver out of sequence over multiple paths, standard SCTP will send back one SACK for every new incoming data chunk even if no packet loss occurs. Therefore, the rate of returned SACK chunks should be reduced when load sharing is applied. In [2], Iyengar et al. propose the concurrent multipath transfer (CMT) which aims at resolving these problems. This is achieved by (i) avoiding unnecessary fast retransmissions by taking into account the paths to which data chunks were sent and the history of acknowledged chunks when making a retransmission decision; (ii) delaying selective acknowledgements appropriately; (iii) allowing fairer updates of the congestion window, since the *cwnd* for a path should not only be increased, when the CTSNA value for an association is increased by an incoming SACK. Therefore, a *path CTSNA* variable is introduced which stores the highest TSN that was acknowledged for this path without discontinuity. Now the *cwnd* is advanced for paths on which new data chunks were acknowledged and for which the *path CTSNA* has advanced.

Although the CMT algorithm adapts the behaviour of SCTP fairly well to the specifics of a load sharing scenario there is still room for improvement. We therefore propose an algorithm for sending *path based selective acknowledgements* to improve the load sharing performance without significantly increasing the complexity of the algorithms. The basic idea of our proposal is that in addition to the CMT algorithm, the load sharing receiver maintains a SACK counter *path_sack_count* for each path – and not only one per association as with CMT or standard SCTP – and increases this counter by one whenever a packet with data chunks is received on the corresponding path. Whenever *path_sack_count*=2, a SACK is immediately sent over the corresponding path and the counter is reset zero. As a result, two successive SCTP packets with data chunks arriving on different paths can trigger two SACKs on these paths. It should be emphasized,

that nevertheless the proposed algorithm still sends one SACK chunk for every two data packets on average in accordance to the requirement in [1]. The strict allocation of SACK chunks to their paths is useful for the SACK-clocking, where each incoming SACK can trigger an update of the outstanding bytes counter, and the receiver and congestion windows.

## 3 Evaluation of the Novel Algorithm

To evaluate the proposed algorithms, an OPNET [3] simulation model of SCTP was developed which implements the CMT algorithm and our new *path based selective acknowledgements* (PB-SACK) algorithm. Figure 1 shows the simula-

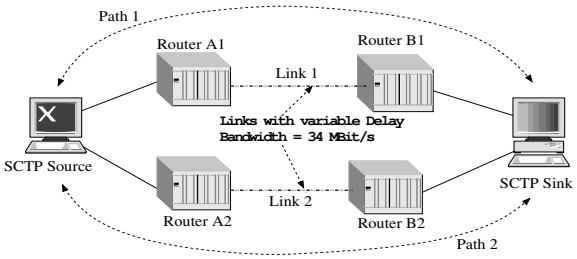


**Fig. 1.** Simulation scenario

tion scenario with two dual-homed IP-based endpoints connected to routers via gigabit ethernet LAN technology. The routers are interconnected by two bottleneck E3 broadband links with a bandwidth of $W = 34,368$ MBit/s. The delay of Link 1, $d_1$, was configured to be 10 ms, and the delay of Link 2, $d_2$, was varied between 10 ms and 200 ms. We assumed a unidirectional data transmission initiated by a saturated traffic source that sends data chunks of constant length (without loss of generality, we assumed 1000 bytes payload per data chunk) towards the receiver.

The results presented in Figure 2 are the averaged throughput and maximum message delays as perceived by the receiver application. To isolate the effects due to the load sharing algorithm from those induced by competing traffic in the network, a scenario without interfering background traffic has been used. Due to the fairly deterministic scenario, the confidence intervals calculated from the repeated simulation runs are insignificantly small and have been omitted. The throughput for both algorithms is limited by the bandwidth of the bottleneck links and fully exploits the link capacity for values of $d_2 \leq 20$ ms. Due to the interdependence between transmissions on both links in the loadsharing scenario, the throughput for higher delays of link 2 decreases, even though $d_1$ remains constant at 10 ms. For 20 ms $< d_2 <$ 70 ms, the throughput of the PB-SACK algorithm is substantially higher than that of the CMT algorithm, as it achieves

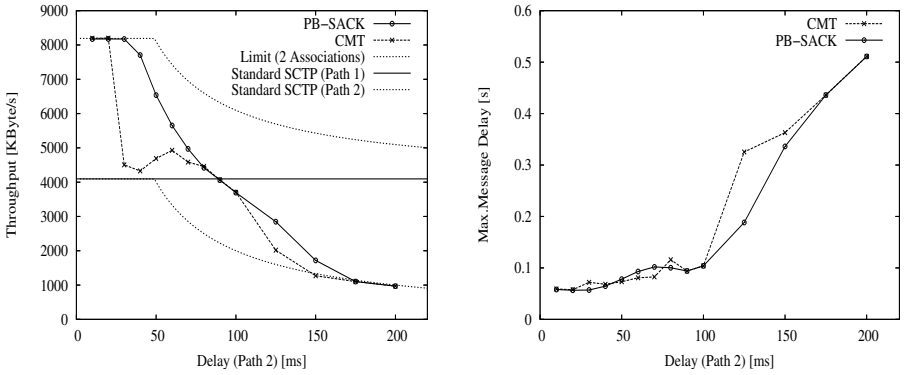


**Fig. 2.** Application layer throughput and maximum message delays of two load sharing algorithms

higher *cwnd* values for path 2. The bandwidth delay product limits the achievable throughput as for all window based transport protocols. Thus, for $d_2 > 100$ ms, the throughput is limited by the receiver window and the link delay. Over the whole parameter range, our PB-SACK algorithm yields a throughput that is higher than or at least as high as that of the CMT algorithm. From the values of the maximum message delays, it is obvious that the increase in throughput of the PB-SACK algorithm has not been achieved at the cost of a higher message delay. For $d_2 > 100$ ms, both algorithms reach a state where effective traffic load distribution cannot be guaranteed any more. At that point, both algorithms allocate traffic almost exclusively to Path 1, and their throughput is limited by a blocked receiver window resulting in an increased message delay.

## 4 Conclusion

The simulations performed so far have confirmed the benefits of load sharing and the superior performance of our PB-SACK algorithm with respect to the achievable throughput for given delay bandwidth combinations. However, additional simulations in more dynamic scenarios are needed. Moreover, a study on how the extension of the scenario to more than two network paths influences the results could provide some additional insight into to possibilities of SCTP-based load sharing.

## References

1. R. Stewart, Q. Xie et al.: RFC 2960 - Stream Control Transmission Protocol, IETF, Network Working Group, October 2000.
2. J.R. Iyengar et al.: Concurrent Multipath Transfer Using SCTP Multihoming, SPECTS 2004, San Jose, July 2004.
3. OPNET Technologies: OPNET Modeler. Commercial simulation tool. Information at http://www.opnet.com/products/modeler/home.html, November 2004