

SCTP MULTISTREAMING: PREFERENTIAL TREATMENT AMONG STREAMS ⁺

Sunil Samtani¹
Janardhan R. Iyengar²
Mariusz A. Fecko¹

¹Telcordia Technologies Inc.
Morristown, NJ – 07960
{ssamtani, mfecko}@research.telcordia.com

²Protocol Engineering Laboratory
CIS Department, University of Delaware
Newark, DE - 19716
iyengar@cis.udel.edu

ABSTRACT

FCS networks are presented with a new transport layer mechanism that, for transmitting multimedia, is markedly superior to transmission over UDP or TCP. SCTP's multistreaming provides an aggregation mechanism for transferring different objects belonging to the same logical application session; however, sharing the congestion state among the streams precludes efficient stream prioritization. We design an SCTP mechanism to provide the application with the service of being able to mark data such that different parts of a transfer (different streams) could be requested to receive preferential treatment from the network. The data flow within an association is divided into separate Sub-association Flows (SF), each SF having its own set of congestion control parameters. We implemented this design using an SCTP stack from Siemens. A number of experiments show that the streams marked with higher priority achieve much better throughput. We plan to investigate a layered congestion avoidance technique that uses state information from individual sub-flows to allow: dynamic addition of sub-flows without slow start; load balancing between paths and sub-flows; and using network state information to provide intelligent feedback to the application.

BACKGROUND

With the introduction of multistreaming in the Stream Control Transmission Protocol (SCTP), Future Combat Systems (FCS) networks are presented with a new transport layer mechanism that, for transmitting multimedia, is markedly superior to transmission over UDP or TCP. Through multistreaming, SCTP provides logical demarcation of data within an application transfer.

⁺ Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

Streams were originally designed to prevent head-of-line blocking at the receiver. Such head-of-line blocking can be observed during an application transfer of multiple, independent objects through a single TCP connection. One could envision using multiple TCP connections for transferring the different application objects, but such a design can have deleterious effects on the network [Balakrishnan et al., 1999].

Conceptually, multistreaming provides an aggregation mechanism for transferring different objects belonging to the same logical application session, such as a multimedia session. Preliminary work has shown the performance benefits of transferring video data over SCTP versus over TCP [Balk et al., 2002]. For a particular transport address of the client, all these different objects follow the same physical path through the network from the server to the client. As a result, the round-trip estimates and the end-to-end available bandwidth estimate (the congestion window, *cwnd*), as probed by the congestion control algorithms, are shared among the different streams. Such sharing of congestion information has been shown to have significant benefits [Balakrishnan et al., 1999].

PREFERENTIAL TREATMENT AMONG STREAMS

As shown in Figure 1, the "X" mark identifies a target in a reconnaissance mission. The application can partition the map into smaller pieces (e.g., region inside the turquoise square), and transmit the partitioned data in multiple streams back to a command and control center. We wish to be able to mark the streams with different ToS (Type of Service) bits or DSCP (Differentiated Services Code Point) markings so that the network can treat these streams differently. The stream that transports the "X" mark should be marked with the highest priority so that the packets could experience minimal latency and loss. Losses experienced by other map pieces should not

affect this piece. In this example, the target information is more relevant to the user when received quickly, and the user need not receive all information to make a critical decision. As described above, we wish to provide the application with the service of being able to send data such that different parts of a transfer (different streams) could be requested to receive preferential treatment from the network.

Currently, adaptive C2 applications transmit multimedia information as different logical streams of data messages. The streams are assigned different priorities on multiple concurrent connections to the command and control center. The application co-ordinates the connection establishment for each different message set. There is no congestion or network state information shared between these connections. SCTP could be used to transmit data for C2 applications on multiple streams within a single association. An example is a universal messaging application, where the user could choose to transmit time-sensitive situational awareness messages on the highest priority stream. The application and the user determine the importance and the priority of the message. At the transport layer, an abstraction that offers dynamic message priority to the user could be used to adapt applications for changing mission-communications requirements. Notwithstanding the benefits offered by SCTP multistreaming, its current design falls short of being able to provide such prioritization among streams. Preliminary results and related work [Akella et al., 2001] have shown that the benefits of preferential treatment among streams are lost due to *false sharing* of congestion information.



Figure 1: Illustration of Multistreaming: Reconnaissance Mapping

A closer inspection shows that the overall reduction in throughput is due to a variety of reasons:

- 1) The stream receiving a lower level of service from the network may experience more losses. These losses subsequently influence the entire transmission

since the *cwnd* is shared, thus reducing overall throughput.

- 2) Unnecessary Fast Retransmissions: Reordering introduced due to the possibly different forward path delays between the different ToS flows could result in spurious fast retransmissions. Such spurious retransmissions would cause the sender to wrongly infer congestion, thus causing unnecessary reduction of the *cwnd* in the congestion avoidance procedure.
- 3) Errors in roundtrip time estimate: Streams with different ToS marking may experience different delays at the routers, causing the single RTT estimate to have a large variance.

When different streams are marked as different ToS flows, the streams are treated differently by a network that schedules packets based on ToS marking. In such a case, one could think of the different ToS flows as following different virtual paths to the same destination, potentially facing different roundtrip times and congestion states in the network. It is also possible for routers to route packets differently based on the ToS marking, thus forcing packets from the same flow to follow different paths through the network. Consequently, it seems unreasonable to share path parameter estimates at the endpoints, such as the bottleneck bandwidth estimate *cwnd* and RTT estimate across such flows.

CONGESTION CONTROL IN SCTP

In this section, we briefly describe the main features of SCTP's congestion control mechanisms. SCTP's congestion control is based on TCP Additive Increase Multiplicative Decrease (AIMD) congestion control principles [Allman et al., 1999], with the addition of selective acks (SACKs). As in TCP, SCTP includes the slow start, congestion avoidance, and fast retransmit mechanisms. We refer readers to [Stewart et al., 2000] and [Stewart et al., 2001] for the details of these mechanisms as present in SCTP. SCTP congestion control deviates from TCP as follows:

- 1) In SCTP, the increase of the *cwnd* is controlled by the number of acknowledged bytes; while in TCP, it is controlled by the number of new acks received.
- 2) SCTP has no explicit fast recovery algorithm as is used in TCP-NewReno. The recovery period in SCTP is implicit after congestion detection and *cwnd* cutback.
- 3) In SCTP, the Max.Burst parameter is applied to avoid flooding the network with a burst of packets after loss recovery or after an idle period.
- 4) In SCTP, fast retransmit is triggered by the fourth missing report of a chunk. This implies that at least

5*MTU of the receiver and congestion window is required to trigger fast retransmission. In TCP, the minimum receiver and congestion window for fast retransmission is 4*MTU, since three duplicate acks trigger fast retransmit.

DETAILED DESIGN

We now proceed to describe our proposed design. As of today, SCTP is the only transport protocol that provides logical demarcation of data within an association or end-to-end connection. SCTP does so through the provision of multistreaming, but fails to provide ToS marking per stream. No good design or API exists today that an application can use to ask the network for preferential treatment of logically separate parts of an application transfer. We are currently working on a design that extends SCTP to handle such preferential treatment among streams. Although our initial work assumes differential treatment of packets by the network, our long term design goals include being able to adapt to highly dynamic network environments, such as the network bottleneck for all the flows going through the same ToS-unaware router, as may well be the case in FCS networks.

Two sequencing schemes exist in the current specification of SCTP – Stream Sequence Number (SSN) and Transmission Sequence Number (TSN). Chunks within each stream are sequenced by the SSN, which is thus used for in-sequence delivery within each stream to the application. SSNs are used to provide partial ordering within an association. The TSN space is oblivious of streams, and is used for congestion control, loss detection and loss recovery. Multiple streams within an association share the TSN space, which is used for reliable transport. The implicit assumption here is that all the chunks in the association follow the same physical path. Since we have already argued that, with ToS marking per stream, the above assumption may not hold, the resulting dilemma is that a SACK that indicates gaps in TSNs may not be conveying correct information to the data sender. The sender has no way of knowing if the observed reordering among TSNs is a result of ToS marking per stream. To resolve this issue, the sender would have to infer the causative TSN (the TSN that causes the currently received SACK to be sent from the receiver) from the SACKs that have been received so far. This inference may not be accurate and would cause computational overheads at the sender.

An alternative solution would be to define a third sequencing scheme which could be used during per stream ToS marking. This modification would be an extension to the current SCTP specification, allowing (in

the simple case of a single QoS domain) explicit sequencing and acks per ToS flow. Bearing in mind the simplicity of implementing the latter solution, we choose this solution for our work. The design is now described in more detail.

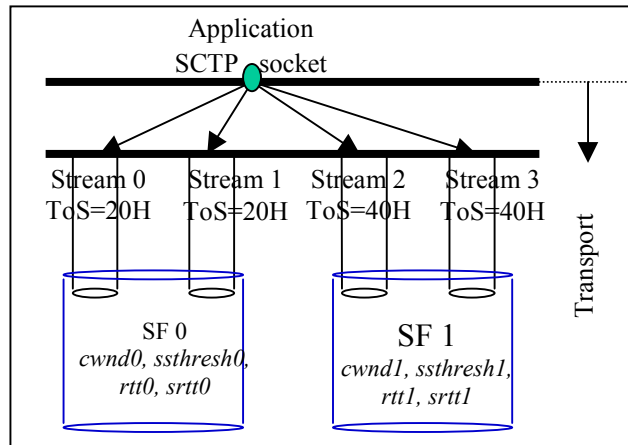


Figure 2: Sub-association Flow Schematic

The design shown in Figure 2 tries to separate the data flow within an association into separate *Sub-association Flows (SF)*, each SF having its own set of congestion control parameters. As shown in the figure, multiple streams can be part of the same logical SF, for example, stream 0 and stream 1 map to SF 0, while stream 2 and 3 map to SF 1. We implemented this design using an SCTP stack from Siemens and the University of Essen. This SCTP stack is a user-space implementation, which runs on the *Linux* operating system. To implement the SF scheme, we made the following changes to the SCTP specification:

- 1) Each SF is identified at the sender by a unique SF Identifier (SFID). Since each SF represents a unique virtual path through the network, the sender should perform *cwnd* and *ssthresh* estimation, RTT estimation, and loss detection and recovery per SF. To facilitate such per-SF treatment, each SF is uniquely represented by an 8-bit *SF identifier* (SFID), which maps to a unique {DestIP, ToS} pair within an association.
- 2) Within each SF, in-sequence chunks are numbered according to a 32-bit SF sequence number (SFSN)}. Thus each TSN uniquely maps to an {SFID, SFSN} pair.
- 3) The receiver of DATA chunks should report cumulative acks, gap acks and duplicates per SF. Thus reliability is now maintained per-SF.

Hence, we modify the DATA chunk to (1) Use SFSN instead of TSN, and (2) Add an 8-bit SFID field to the chunk. This enables the receiver to send SACKs per SF. The initial TSN is still exchanged in the handshake, and

each SF starts with the initial SFSN equal to the initial TSN exchanged. The SFSNs are incremented independently within each SF during the lifetime of the association. The modified DATA chunk is shown in Figure 3. All fields except for the SFID and SFSN fields are as defined in RFC2960 [Stewart et al., 2000].

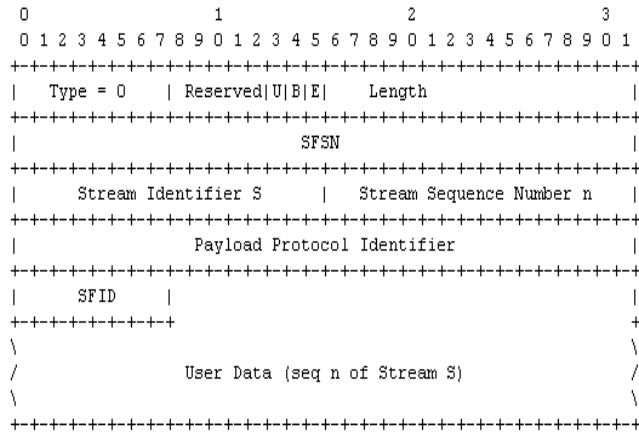


Figure 3: Modified DATA Chunk

Since the receiver sends SACKs per-SF, the SACKs should contain SF information for the sender to perform congestion control and RTT measurements per-SF. We therefore revise the SACK chunk as shown in Figure 4. As in the DATA chunk, we add an 8-bit SFID field. Note that the fields that carry TSNs as defined by RFC2960 have been modified to carry SFSNs.

The sender maintains following parameters per SF: (1) Congestion Control parameters - cwnd, ssthresh, and (2) RTT estimate parameters - rto, srtt, rttvar. The retransmission timer (T3-timer) is maintained per-SF. The sender performs loss detection and recovery per-SF, which are facilitated by SACKs per-SF and retransmission queues at the sender per-SF.

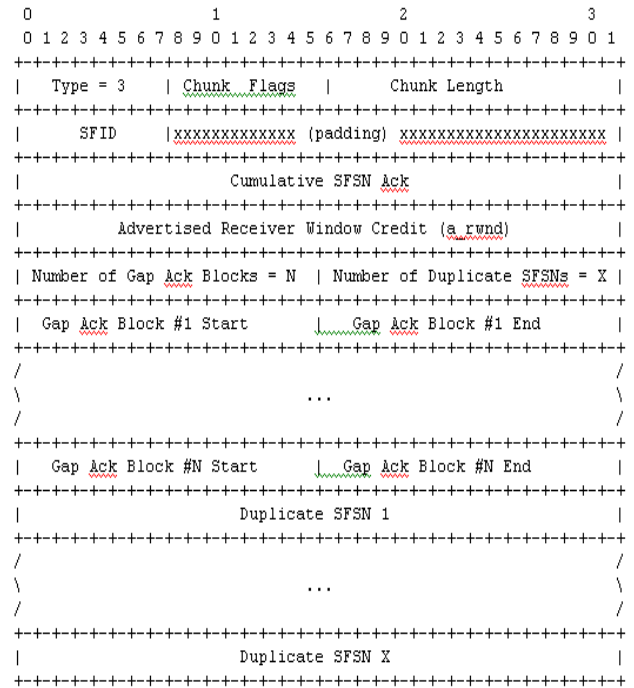


Figure 4: Modified SACK Chunk

These changes to RFC2960 cause changes in on-the-wire bits. Per-SF mechanisms can be implemented with added complexity at the sender alone, with receiver behaviour and on-the-wire bits unchanged from the specification. Conceptually though, the above described design is cleaner. Our motive is to empirically observe and quantify the benefits of performing congestion control, loss detection and recovery per-SF over RFC2960 for preferential marking of streams in ToS responsive networks. We believe that this design serves our purpose.

To preserve backward compatibility, we further negotiate SF capability in an association. At an association's initialization, an SF Capability Negotiation (SFCN) chunk will be transmitted in the Cookie Echo (Init Ack) leg. If the receiver echoes the chunk back in the Cookie Ack (Cookie Echo) leg, then the association will use SF mechanisms. This chunk is a new chunk type. Otherwise, the handshake is the same as in RFC2960.

INITIAL RESULTS & CONCLUSIONS

As shown in Figures 5 and 6, preliminary results from experiments are encouraging. Figure 5 shows transfer latency with unmodified SCTP and Figure 6 with our design modifications to SCTP. One of the ToS flows in both experiments was subjected to 5% loss while the other was subjected to no loss. In Figure 5, sharing the congestion state among the streams causes the low-priority stream to impair the transmission of the high-

priority one. On the other hand, in Figure 6, the high-priority stream (in green) finishes much faster. The links in our experiment are emulated using NISTNET.

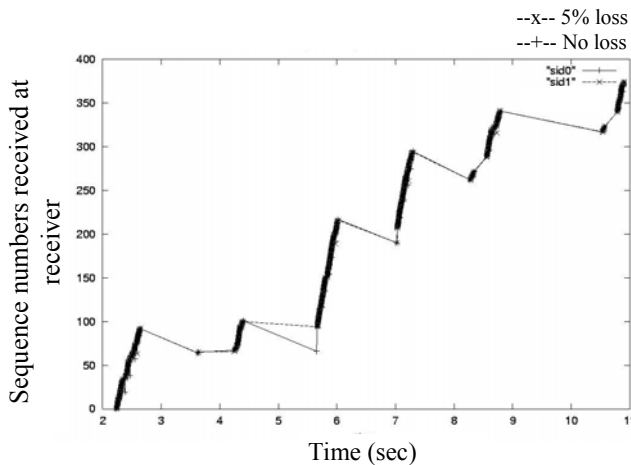


Figure 5: Transfer latency with unmodified SCTP, and per-stream ToS marking

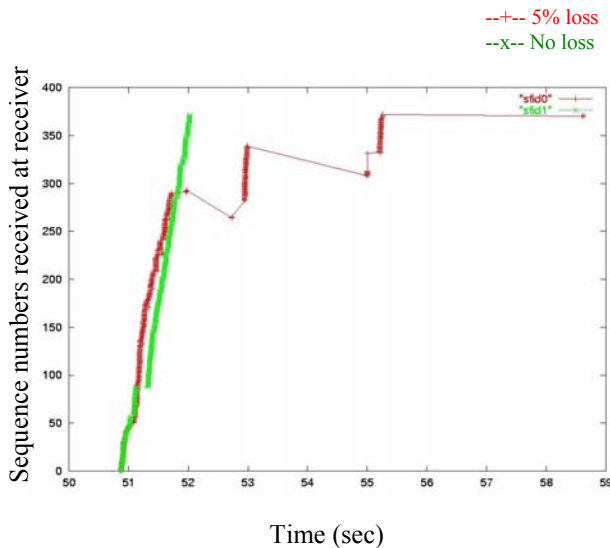


Figure 6: Transfer latency with modified SCTP, and per-stream ToS marking

We implemented prioritized treatment among SCTP streams from the network. The design equipped each Sub-association Flow (SF) with the variables and data structures that are maintained per association in the original SCTP definition. It is possible to establish multiple associations with different priorities as an alternative implementation of the motivating problem. Our approach provides the application developer with an abstraction in SCTP using the sockets API, which creates multiple priority streams within an association with minimal modifications to the SCTP association setup and teardown procedures. Further, we plan to investigate a layered congestion avoidance technique that uses state

information from individual sub-flows to allow for: (a) dynamic addition of sub-flows without slow start, (b) load balancing between paths and sub-flows, and (c) using network state information to provide intelligent feedback to the application.

Working within a single association may have additional benefits. It should be easier for a path-management system to manage a single association for the purpose of scheduling transmissions or load balancing among streams. When embedded inside the transport protocol, the path-management system can gain direct access to all relevant transport-state variables, without a special-purpose API to the transport layer. It can also refine the definitions of basic parameters of the protocol (e.g., primary and secondary paths) based on their interactions with the path-management scheme.

REFERENCES

- [Balakrishnan et al., 1999] H. Balakrishnan, H. Rahul, S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In ACM SIGCOMM, Cambridge, MA, September 1999.
- [Balk et al., 2002] A. Balk, M. Sigler, M. Gerla, M. Y. Sanadidi. Investigation of MPEG-4 Video Streaming over SCTP. Proc. SCI2002, Orlando, July 2002.
- [Akella et al., 2001] A. Akella et al. The Impact of False Sharing on Shared Congestion Management. CMU SCS Technical Report CMU-CS-01-135.
- [Stewart et al., 2001] R. Stewart and Q. Xie. *Stream Control Transmission Protocol (SCTP): A Reference Guide*. Addison Wesley, New York, NY, 2001.
- [Stewart et al., 2000] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson. Stream Control Transmission Protocol. RFC 2960, IETF, October 2000.
- [Allman et al., 1999] M. Allman, V. Paxson, W. Stevens. TCP Congestion Control. RFC 2581, IETF, April 1999.

DISCLAIMER

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government.