# Priorities In Stream Transmission Control Protocol (SCTP) Multistreaming

Gerard J. Heinz II, Paul D. Amer
Protocol Engineering Laboratory, University of Delaware
Newark, DE, 19716, USA

## ABSTRACT

This paper[1] introduces per-stream priorities as a method of decreasing delays of important data during periods of low bandwidth availability. We group streams into priority classes and propose a scheduling algorithm to prioritize data among these classes. Per-stream priorities are useful in applications sending different types of data, such as Instant Messaging (IM) systems. We discuss the details of present (AOL Instant Messenger) and future IM systems (using Session Initiation Protocol), and how these systems benefit from prioritized SCTP. We discuss the practical application of such a scheme in example programs. Through simulation with ns-2, we compare prioritized SCTP to non-prioritized SCTP using an application with two streams, one sending high-priority sporadic data while the other is sending low-priority bulk data. In periods when the bulk data submission rate is greater than the available bandwidth, and the transmission rate of the sporadic data is less than the link transmission rate, we demonstrate that with per-stream priorities the bulk data transfer will not affect the quality of the more important sporadic data transfer.

**Keywords:** SCTP, multistreaming, prioritization, instant messaging, SIP.

## 1. INTRODUCTION

With the steady improvement of computer and network systems, end users are growing more accustomed to on-line multimedia experiences. However, these experiences are often bandwidth-intensive and, to be comfortable to end users, require high throughput connections. While the number of broadband subscribers grows daily, the majority of Internet users still rely on slower dial-up modem connections, which are often insufficient for comfortable viewing of multimedia data. In addition, many pocket devices, such as mobile phones and Personal Digital Assistants (PDAs), now offer web browsing and streaming video over low bandwidth, wireless connections. The maximum throughput achieved by these devices fluctuates depending on signal strength.

In general, current network end users want to request various types of data from server applications. Therefore, the server applications must provide a way to transmit multiple data-types in parallel, and must effectively respond to periods of insufficient bandwidth.

This paper purposes an optimization to the Stream Transmission Control Protocol (SCTP) [1, 2], a new general-purpose transport protocol. This new feature allows end applications to specify the relative importance of data. We introduce data priority based upon logical flows of data from sender to receiver. In Section 2, we discuss the classical approaches to logical data separation, and detail the motivations for moving beyond these approaches. In Section 3, we discuss Instant Messaging and how this important multimedia application will benefit from our prioritized transport service. Section 4 defines per-stream priorities as an optional scheduling algorithm for an SCTP sender. Section 5proposes a possible implementation of per-stream priorities in SCTP. Sections 6 and 7 detail our performance evaluation of priority-enhanced SCTP. Section 8 closes with some remarks and suggestions for future work.

## 2. MOTIVATION

Throughout our discussion, we refer to applications that must exchange multiple types of data between hosts. For instance, a media server needs to transmit sound data and video data to another host. We limit our focus to multimedia applications in which each type of data can be *ranked* in order of relative importance.

Traditionally, transmitting different types of data in parallel between endpoints relied on one of three approaches. In all three situations, Host A would like to send three types of data (labeled Data 1 through Data 3) to Host B. In the first approach, Host A opens three TCP connections to Host B – one connection per data type. While this approach provides logical separation of data based on type, multiple connections defeat TCP-friendly congestion control by allowing an application to gain an unfair portion of available bandwidth at the expense of other data flows in the network.

In the second approach, Host A multiplexes and demultiplexes the three types of data over a single connection. Applications using this approach maintain TCP-friendly congestion control; however, this approach increases complexity for the application programmer, since the application itself must handle the complicated task of efficiently and fairly multiplexing data transmission.

The third approach has a multimedia application on Host A using UDP to transmit to Host B. This approach closely resembles the second approach; however application programmers must also supply their own reliability service due to UDP's unreliable, connectionless service.

With the introduction of SCTP and its concept of streams, applications are presented with a new transport layer solution to transmitting multiple types of data. This new approach combines advantages of multiple end-to-end connections and application

[1] Prepared, in part, through collaborative participation in the Communications and Networks Consortium sponsored by the US Army Research Lab's CTA Program, Cooperative Agreement DAAD19-01-2-0011. The US Gov't is authorized to reproduce and distribute reprints for Gov't purposes notwithstanding any copyright notation thereon.

multiplexing/demultiplexing. An SCTP stream is a logical, unidirectional communication channel that exists within an SCTP end-to-end association. An SCTP endpoint may request multiple outbound streams during association setup. Each stream is given an independent send and receive buffer (at the sender and receiver, respectively). These streams and buffers exist for the duration of the SCTP association.

Figure 1 shows an example case where Hosts A and B have established a multistreamed association. In this example, Host A would like to transmit three different types of data. Therefore, during association setup, Host A requests three streams to Host B (numbered streams 0 to 2). Host B has only one type of data to send to Host A, and therefore requests and maintains one stream to Host A (numbered stream 0).

Since Host A has multiple outbound streams, the SCTP implementation on Host A must provide a scheduling algorithm for data transmission across the streams. The SCTP Implementers Guide [3] states that an SCTP implementation must choose a scheduling algorithm that avoids stream starvation; delivery of data submitted to a certain stream may not be indefinitely postponed. The guide offers two algorithms: round-robin or first-come first-serve.

In round-robin, Host A would select a data chunk from stream 0's send queue. If Host A could still send data (as allowed by Host B's advertised receiver window and Host A's congestion window), Host A would send a data chunk from stream 1's send queue. A data chunk from A's stream 2's send queue would be sent next. Then, Host A would send a data chunk from stream 0. This round-robin process would continue for the life of the association.

Using first-come first-serve for scheduling, Host A would transmit each data chunk in the order the chunk was received from the application layer regardless of the chunk's send queue.

While streams were originally introduced to prevent head-of-line blocking in SS7-style signaling over IP networks, streams also provide applications with logical data separation and a central point of connection management. Streams are managed within an association's current receiver and congestion windows. With the introduction of the PR-SCTP (Partial-Reliability) Internet Draft [4] to the IETF, SCTP streams can effectively transport loss-tolerant multimedia data in addition to loss-intolerant data. Thus, SCTP streams provide powerful features to applications without increasing application complexity.

Even with these strengths, networks whose nodes are highly mobile and volatile can prove problematic for an SCTP association using multiple streams. Namely, fluctuating available bandwidth (for example, induced by congestion or signal degradation) introduces delays in communication between endpoints across all streams due to SCTP's data chunk scheduling algorithm. For example, multimedia network applications running on PDAs and mobile phones could experience a decrease in the quality of service and data rate as they roamed further from a signal tower.

## 3. SAMPLE APPLICATION

Many applications may benefit from per-stream priorities. In this paper, we focus on one widely used, and increasingly popular, general-purpose application – instant messaging (IM).

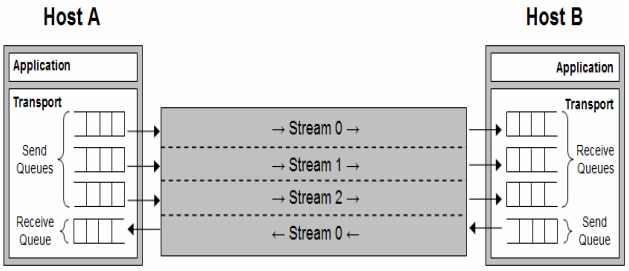



**Figure 1 - Multi-Streamed SCTP Association between**

Current IM architectures (e.g., AOL Instant Messenger [5]) are limited in that they require multiple TCP connections to directly connect users. As explained in Section 2, multiple TCP connections have various disadvantages. The next generation IM protocol, SIP with SIMPLE [2] [6, 7] allows such direct connections while maintaining transport layer independence. Due to SIP's wide support by software industry giants and the 3rd Generation Partnership Project (3GPP), SIP is slated to become a future session protocol of choice for IM software.

The SIMPLE extension [7] allows users to exchange text messages without any relationship (or state) between messages. Text messages traverse a SIP based network in a method similar to that of the general model to be discussed in Section 5.1. However, if two users want to exchange video or voice data, the full functionality of SIP is required.

SIP provides primitives used for the negotiation of multimedia 'sessions' between two hosts, called *calls*. Primitives support the setup and teardown of these calls between two or more users reachable on an IP network. Throughout the network, SIP Registrar Servers track user's reach-ability for a given domain. In Figure 2, Alice belongs to DomainA.com. When she logs onto the SIP network, she exchanges information with the Registrar Server for her domain. The Registrar Server, in turn, makes note that Alice is reachable, or *present*.

If Alice then wants to call Bob, she sends Bob an INVITE message. Since Bob is not in the same domain as Alice, Alice forwards the INVITE to her domain's SIP proxy server[3] (shown in Figure 2 as DomainA.com). Alice's proxy server then performs a DNS lookup to determine the location of DomainB.com's SIP Registrar server. When the lookup is complete, Alice's SIP proxy forwards the request to DomainB.com's Registrar server; which in turn forwards the INVITE to Bob. If Bob accepts the INVITE, then SIP is used to negotiate call options.

Using header information, all control data for the call will flow over the same application-layer path as the first request – that is, control information will flow from Alice to Alice's SIP proxy to Bob's SIP Registrar to Bob. This path is called the *control path* (as denoted by the solid line). The state of the call is maintained

---

[2] SIP – Session Initiation Protocol; SIMPLE – SIP for Instant Messaging and Presence Leveraging Extensions.

[3] In this scenario, server DomainA.com is both a SIP proxy and a SIP registrar (analogously for server DomainB.com).

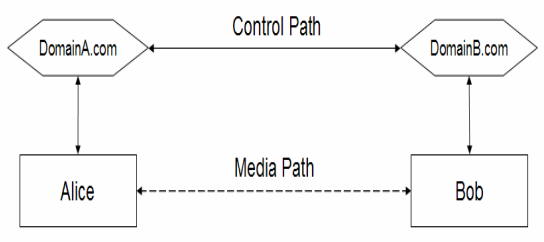



**Figure 2 - a SIP call between Alice and Bob**

though this control path until Alice or Bob decide to terminate the session.

During the call negotiation, a *media path* is also setup whereby Alice can transmit data directly to Bob along a direct connection (as denoted by the dotted line). In this case, media can represent any data type – including text, audio, video, or files.

SIP provides transport layer independence; control and media paths currently operate over either UDP or TCP. Due to SCTP's inherent benefits to signaling, SCTP has been suggested as another possible transport layer to SIP [11]. While one SCTP association could be maintained between Alice and the DomainA.com SIP Server, another could be maintained between Alice and Bob (thereby creating the media path).

Let us return to our IM scenario between Alice and Bob. Alice would like to send Bob a file while maintaining a comfortable text conversation with him. Using SIP over SCTP, a priority scheme addresses this situation. Let the chat data go over stream 0, and the image data go over stream 1. During an SCTP association, chat data would be sent immediately while the bulk data would be delayed. If there remains room in the receiver and the congestion windows after all stream 0 data has been sent, the stream 1's bulk data would be sent. During periods when stream 0 has no data to send, stream 1 would transmit as much data as allowed by the available window space.

## 4. DEFINITION OF PER-STREAM PRIORITY

To grant multimedia applications an ability to address periods of poor network conditions, we investigate the theoretical and practical implications of adding priorities to SCTP streams. A per-stream priority scheme allows an application to specify relative importance of data. Given the network conditions at the moment, transmission of critical data should take precedence, thereby reducing perceived delays for critical data during periods of low quality of service. In this example situation, whenever possible, less important data should be transmitted, depending on available bandwidth.

We define an SCTP stream priority scheme as:

*Data on stream j always have greater or equal priority in relation to data on stream k, with j < k*

Stream priorities extend SCTP's existing *sender-side* API and scheduling algorithm implementation only. Priorities do not change the on-the-wire SCTP protocol, and thereby do not change SCTP's current packet format (i.e., no addition of a new control chunk). By avoiding such modifications, stream priorities do not require SCTP's *receiver-side* to be aware that prioritization is occurring at the sender. This transparency maintains backward compatibility with non-priority enhanced endpoints, thereby allowing any SCTP receiver to operate with both priority and non-priority enhanced SCTP senders. Transparency facilitates Internet deployment.

Originally, we considered a strict priority scheme for SCTP. In such a scheme, items on stream j always have priority over items on stream k, with j < k. However, a strict priority scheme has an obvious weakness: in cases where an SCTP sender has bandwidth sufficient only to transmit data for streams 0-3, data on stream 4 will be indefinitely postponed. In some applications (such as SS7 signaling and stereo audio streaming applications), multiple data streams are considered of equal importance. By assigning the same priority to two or more streams, our priority scheme will treat those streams' data equally.

## 5. SPECIFICATION

To implement the priority scheme discussed in Section 4, we must first add a new field to the SCTP stream data-structure. This integer field (`priority`) will store a positive value corresponding to the relative priority of the stream. Upon association setup, `priority` must be initialized to `0` for all streams. In this manner, unless the values are changed, priority-enhanced SCTP behaves as basic SCTP.

Secondly, the priority scheme implementation requires the addition of the following interfaces to the SCTP Sockets API:

`sctp_enablepriority` sets a default strict priority scheme among an SCTP association's streams. To adjust priorities and to rank streams of equal importance, use:

```
sctp_enablepriority ()
{
  for (j = 0; j < num_streams; j++)
    streams[j].priority = j;
}
```

`sctp_setequalpriority` sets all streams between (and including) streams startStream and endStream to equal priority. Then, the interface resets the priorities on all streams greater than the endStream.

```
sctp_setequalpriority (int startStream,
                       int endStream)
{
  int p = streams[startStream].priority;
  for (j=startStream; j<=endStream; j++)
    streams[j].priority = p;

  for (j=endStream+1; j<num_streams; j++)
  {
  p++;
  streams[j].priority = p;
  }
}
```

To disable SCTP priorities completely, use:

```
sctp_disablepriority ()
{
  for (j=0; j < num_streams; j++)
    streams[j].priority = 0;
}
```

```
1.  int stream_num;                  // current stream number
2.  int current_priority             // current priority
3.  boolean priority_class_has_data  // does the current priority
                                     //  have any streams that
                                     //  have data to be sent?
4.  int priority_class_base_stream   // the first stream with the
                                     //  current priority

5.  stream_num = 0;
6.  current_priority = 0;
7.  priority_class_has_data = false;
8.  priority_class_base_stream = 0;

9.  while (space exists in SCTP packet)
10. {
11.     if (streams[stream_num] has data chunks to transmit)
12.     {
13.           assign TSN to a chunk from streams[stream_num]
14.           bundle chunk into SCTP packet

15.           priority_class_has_data = true;
16.     }

17.     stream_num++;

18.     if ((priority_class_has_data) &&
            (streams[stream_num].priorities > current_priority))
19.     {
20.           stream_num = priority_class_base_stream;
21.           priority_class_has_data = false;
22.     }
23.     else if(streams[stream_num].priorities > current_priority)
24.     {
25.           current_priority++;
26.           priority_class_base_stream = stream_num;
27.           priority_class_has_data = false;
28.     }

29. }
```

**Figure 3 – proposed priority algorithm for SCTP**

Finally, when priorities are enabled, the algorithm in Figure 3 should be employed to assign TSN numbers to data among the stream queues.

The algorithm in Figure 3 should be added after the implementation checks for space in the congestion and receiver windows. After initialization (Lines 1-8), the algorithm performs a round-robin type operation among the streams while space exists in the SCTP packet (Line 9). First, the current stream is checked for data (Line 11). If the stream has data chunks to transmit, a TSN is assigned to the data chunk at the head of the stream's queue (Lines 13-14). This chunk is now ready for transport. Since the stream had data to send, the `priority_class_has_data` flag is set to `true` (Line 15). This flag allows the algorithm to track when the priority level should be increased.

The algorithm then considers the next stream number (Lines 17-28). If this stream is of the same priority observed before, then we loop to process the new stream's data. If this stream is of a lower priority, we check the `priority_class_has_data` flag. If the flag is set to `true`, then the algorithm resets the stream number to the first stream number with the current priority (Line 26). This ensures that all data of greater priority takes precedence over all lower priority data.

**Theorem:** Figure 3's algorithm in has worst case running-time complexity of order *O (n + d).*

**Proof:** On initial observation, we might conclude that a number of factors influence the running time: namely, size of the SCTP packet, number of streams, amount of data to transmit, and number of priority classes. However, when analyzing the worst-case running-time, we can limit these factors to only the number of streams and the amount of data to transmit.

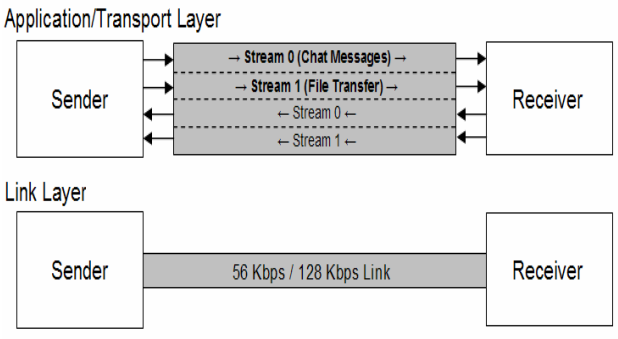



**Figure 5 - Simulation Network**

We do not consider the packet size since, at any time instance, either:

1. The packet size is too small for all of the available stream data – This condition will restrict the running time, since the algorithm terminates once a packet is full, or
2. The packet size is greater than the amount of available data – In this case, the amount of available data will ultimately determine running time.

In theory, a worst-case scenario occurs when the amount of packet space remaining is infinite (of course, in practice, packet space is finite.)

Priority classes do not affect running time. Classes only affect the order in which packets are transmitted during the algorithm's execution, and do not increase the total number of loop iterations.

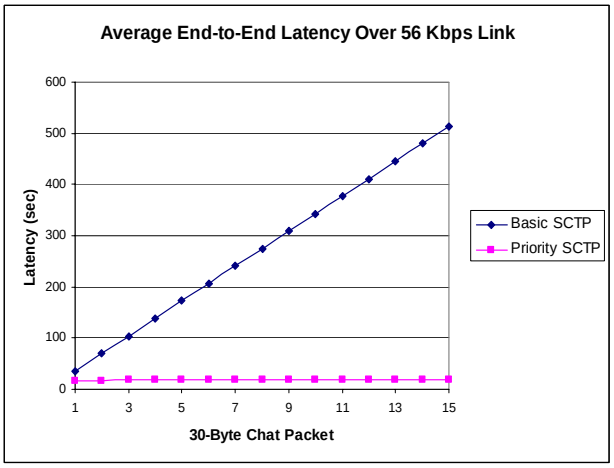
Therefore, we consider the worst-case scenario when at most *d* chunks of data must be transmitted over *n* streams. We set the SCTP packet size to infinity to maximize the running time. Then, the running time (*T*) for our algorithm is: $T = n + d$

The while loop will execute at least once for each stream in the association (to determine if there is data to send on that stream – Line 11) and once for each piece of data. ■

## 6. SIMULATION

To investigate the theoretical and practical implications of adding stream priorities to SCTP, we added the strict priority scheme into the University of Delaware's SCTP module [12] for ns-2. This module is in wide use and provides a strong baseline for SCTP functionality.

Through simulation we demonstrate that, based on our performance criteria, priority-enhanced SCTP outperforms basic SCTP in the IM application discussed in the Section 5. In our simulation, we rate performance by measuring the end-to-end latency experienced by the text messaging data on stream 0. We demonstrate that the chat performance is not sacrificed in the presence of a file transfer. SCTP with priorities performs as well as basic SCTP in the worst conditions. Otherwise, chat data on



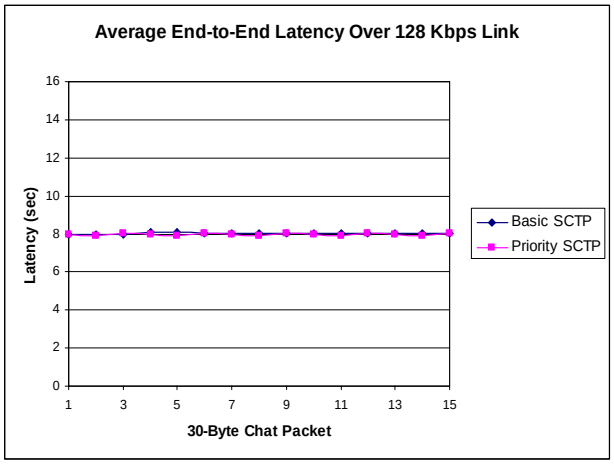




**Figure 6 - Latency for Chat Packets**

stream 0 of priority-enhanced SCTP is delivered to the receiver sooner after submission than the chat data of basic SCTP.

For our simulation, we created a link between two hosts; a sender and a receiver (see Figure 4). We establish an SCTP association with two streams in either direction. The sender's stream 0 carries small packets (30 bytes of application-layer data) representative of text messages used during an instant messaging conversation. To simulate a conversation, this data is generated at 30-second intervals. The sender's stream 1 carries the bulk data for a continuous file transfer, representative of a large image file transfer.

Using this association, we compare "basic SCTP" to "priority SCTP". To illustrate the performance gain by priority SCTP, we test both versions over a simulated 56Kbps (Dial-up modem) link and a 128Kbps (ISDN) link. This link, for both speeds, has a propagation delay of 250ms.

During simulation we expect that, with priorities, the quality of the chat data on stream 0 should not be affected by the bulk data on stream 1 during certain conditions. Specifically, let:

$R_0$ = Rate of data submitted by the application for transmission over stream 0.
$R_1$ = Rate of data submitted by the application for transmission over stream 1.
$R_{Available}$ = Rate at which SCTP can send data to the receiver.

For these values, we assume application-layer data rates. Two conditions exist when priorities will have minimal, if any, effect on data transmission.

Condition (1): $R_0 + R_1 < R_{Available}$. During this time, there will be no transport layer queuing; data will be transmitted immediately after receipt from the application.

Condition (2): $R_0 > R_{Available}$. During this period, queuing occurs on stream 0, introducing delays regardless of the amount of data on stream 1. Under (2), a priority scheme cannot prevent delays on stream 0. However, the delays experienced on stream 0 should still be less than the delays on an unprioritized stream 0.

**Hypothesis**: With per-stream priorities, the quality of sporadic data on stream 0 will not be affected by the bulk data on stream 1 when Condition (3): $R_0 < R_{Available} < R_0 + R_1$. Under these conditions, the data on stream 0 would be transmitted in favor of data on stream 1.

Throughout the simulation, we analyze the 'quality of sporadic data' by measuring the end-to-end latency of data on stream 0. End-to-end latency is measured as the time from when a sending application gives text data to the SCTP sender until the SCTP receiver delivers that text to the receiving application.

## 7. RESULTS

We performed the above simulation 100 times, with different random seeds. For each, we discarded the initial 5 packets to ignore start-up effects, and recorded end-to-end latency the next 15 sequential chat packets. The left graph of Figure 5 shows average end-to-end latency for the 30-byte chat packets on stream 0 across a 56Kbps line. This simulation demonstrates our hypothesis scenario condition (3).

The first-come first-serve nature of basic SCTP increases the end-to-end latency of the data on stream 0. As hypothesized, this low bandwidth link produces an unacceptable latency for stream 0 data. Since the data on stream 1 will be queued at a constant rate, the sender's transmit queue will be filled with data from stream 1. All of this data must be transmitted before new data on stream 0 is sent. Therefore, the latency on stream 0 increases over time as more data for stream 1 is queued for transport at the sender.

While using priority SCTP over a 56Kbps link, the latency on stream 0 remains fairly constant, despite the lower bandwidth. The chat packets report an average delay of 8 seconds. We know that the data on stream 0 is transmitted upon receipt by the sender's transport layer from the sending application.

The right graph of Figure 5 demonstrates Condition 1 above. At the higher bandwidth (128 Kbps), priority SCTP behaves similar to basic SCTP. The link speed for both the priority and basic transmissions is faster than the rate at which the sending application produces packets. A 30-byte chat message does not experience queuing in either priority SCTP or basic SCTP.

Therefore, the latency observed, derived from the propagation delay, in the graph is equivalent for both protocols. This condition demonstrates the stability of priority-SCTP.

We did not simulate Condition 2 above. While maintaining a realistic chat traffic simulation, Condition 2 is unlikely in any practical scenario. To observe Condition 2 in our simulation, $R_{Available}$ would reduce to less than 1 byte per second.

## 8. CONCLUSIONS AND FUTURE WORK

The strict priority scheme presented in this paper only addresses reliable data. Future work should investigate using priorities with PR-SCTP [4], the partially reliable data transfer option for SCTP. PR-SCTP, as mentioned before, allows streams to have varying levels of reliability. Applications using both PR-SCTP and per-stream priorities could transmit important data while expiring data of lesser importance in times of insufficient bandwidth. Such an investigation would extend per-stream priorities to effectively handle such data as left and right stereo feeds, MPEG video frames, and other time-sensitive material.

## 9. 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 US Government.

## 10. REFERENCES


[1] 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**, 10/00.

[2] R. Stewart, Q. Xie. **Stream Control Transmission Protocol (SCTP): A Reference Guide**. New York: Addison-Wesley, 2002.

[3] R. Stewart, L. Ong, I. Arias-Rodriguez, K. Poon, P. Conrad, A. Caro, M. Tuexen. "SCTP Implementers Guide," **draft-ietf-tsvwg-sctpimpguide-10.txt, Internet-Draft**, 12/03 (Work in Progress).

[4] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, P. Conrad. "SCTP Partial Reliability Extension." **draft-ietf-tsvwg-prsctp-03.txt, Internet Draft**, 1/04 (Work in Progress).

[5] AOL Instant Messenger, www.aim.com.

[6] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler "SIP: Session Initiation Protocol," **RFC 3261**, 6/02.

[7] B. Campbell, J. Rosenberg. "Session Initiation Protocol Extension for Instant Messaging," **RFC 3428**, 12/02.

[8] R. Stewart, Q. Xie, L. Yarroll, J. Wood, K. Poon, K. Fujita, M. Tuexen. "Sockets API Extensions for Stream Control Transmission Protocol." **draft-ietf-tsvwg-sctpsocket-08.txt, Internet-Draft**, 9/03 (Work in Progress).

[9] UnOfficial AIM/OSCAR Protocol Spec, aimdoc.sourceforge.net/OSCARdoc

[10] Java Developer Pages: Terminology, www.aim.aol.com/javadev/terminology.html

[11] J. Rosenberg, H. Schulzrinne, G. Camarillo. "The Stream Control Transmission Protocol as a Transport for the Session Initiation Protocol." **draft-ietf-sip-sctp-04.txt, Internet-Draft,** 6/02 (Work in Progress).

[12] A. Caro, J. Iyengar. ns-2 SCTP module. http://pel.cis.udel.edu.