Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

R. Stewart
Netflix, Inc.
M. Tuexen
Muenster Univ. of Appl. Sciences
S. Loreto
Ericsson
R. Seggelmann
Metafinanz Informationssysteme GmbH
March 21, 2016

Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol
draft-ietf-tsvwg-sctp-ndata-05.txt

Abstract

The Stream Control Transmission Protocol (SCTP) is a message oriented transport protocol supporting arbitrary large user messages. However, the sender can not interleave different user messages which causes head of line blocking at the sender side. To overcome this limitation, this document adds a new data chunk to SCTP.

Whenever an SCTP sender is allowed to send a user data, it can possibly choose from multiple outgoing SCTP streams. Multiple ways for this selection, called stream schedulers, are defined. Some of them don't require the support of user message interleaving, some do.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 22, 2016.

Table of Contents

1. Introduction

1.1. Overview

When SCTP [RFC4960] was initially designed it was mainly envisioned for the transport of small signaling messages. Late in the design stage it was decided to add support for fragmentation and reassembly of larger messages with the thought that someday Session Initiation Protocol (SIP) [RFC3261] style signaling messages may also need to use SCTP and a single MTU sized message would be too small. Unfortunately this design decision, though valid at the time, did not account for other applications which might send very large messages over SCTP. When such large messages are now sent over SCTP a form of sender side head of line blocking becomes created within the protocol. This head of line blocking is caused by the use of the Transmission Sequence Number (TSN) for three different purposes:

1. As an identifier for DATA chunks to provide a reliable transfer.

2. As an identifier for the sequence of fragments to allow reassembly.

3. As a sequence number allowing to have up to 2**16 - 1 SSNs outstanding.

The protocol requires all fragments of a user message to have consecutive TSNs. Therefore it is impossible for the sender to interleave different user messages.

This document also defines several stream schedulers for general SCTP associations. If support for user message interleaving has been negotiated, several more schedulers are available.

The following Figure 1 illustrates the behaviour of a round robin stream scheduler using DATA chunks. Please note that the use of such an scheduler implies late TSN assignment but it can be used with an [RFC4960] compliant implementation not supporting user message interleaving.

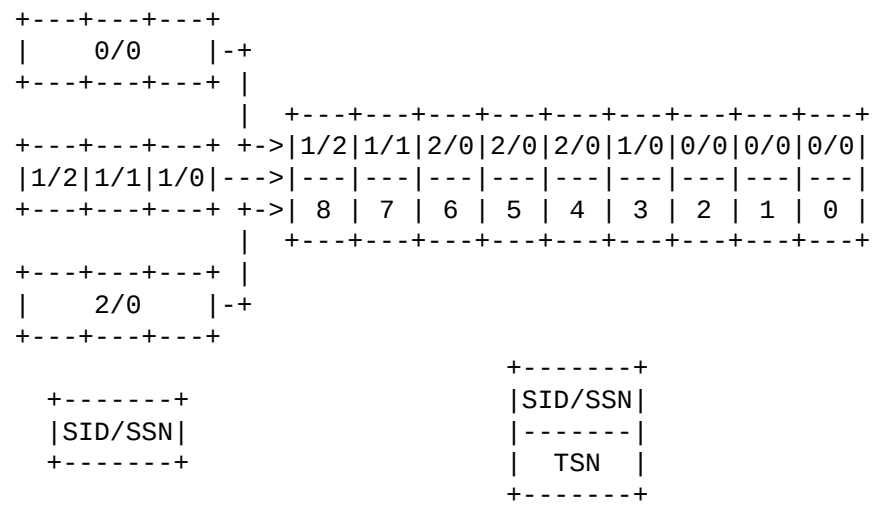

Figure 1: Round Robin Scheduler without User Message Interleaving

This document describes a new Data chunk called I-DATA. This chunk incorporates all the flags and fields except the Stream Sequence Number (SSN) and properties of the current SCTP Data chunk but also adds two new fields in its chunk header, the Fragment Sequence Number (FSN) and the Message Identifier (MID). Then the FSN is only used for reassembling all fragments having the same MID and ordering property. The TSN is only for the reliable transfer in combination with SACK chunks.

The MID is also used for ensuring ordered delivery, therefore replacing the stream sequence number. Therefore, the head of line blocking caused by the original design is avoided.

The following Figure 2 illustrates the behaviour of an interleaving round robin stream scheduler using I-DATA chunks.

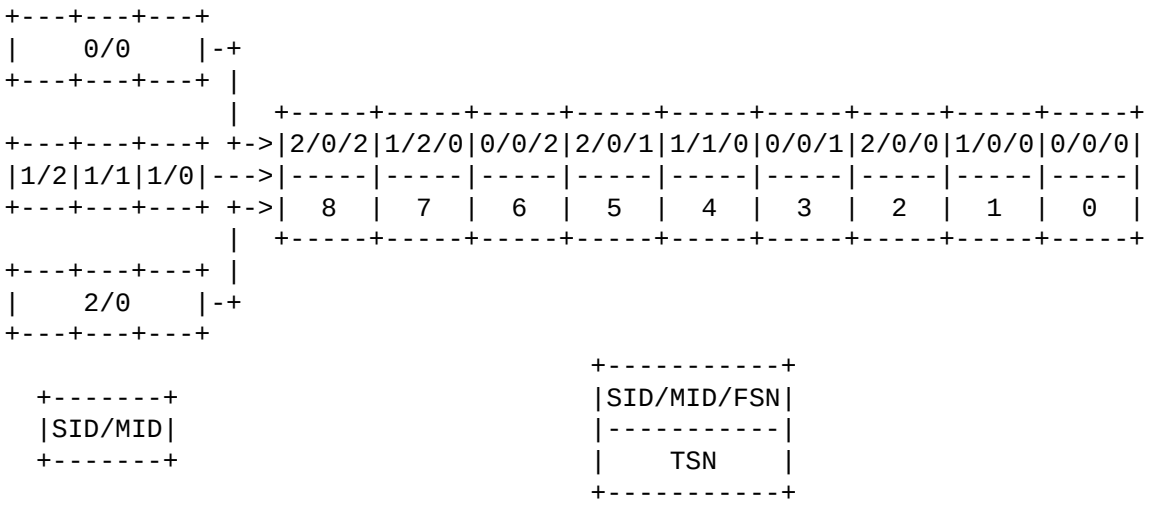

Figure 2: Round Robin Scheduler with User Message Interleaving

The support of the I-DATA chunk is negotiated during the association setup using the Supported Extensions Parameter as defined in [RFC5061]. If I-DATA support has been negotiated for an association I-DATA chunks are used for all user-messages and no DATA chunks. It should be noted, that an SCTP implementation needs to support the coexistence of associations using DATA chunks and associations using I-DATA chunks.

1.2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. User Message Interleaving

The interleaving of user messages is required for WebRTC Datachannels as specified in [I-D.ietf-rtcweb-data-channel].

2.1. The I-DATA Chunk supporting User Message Interleaving

The following Figure 3 shows the new I-DATA chunk allowing user messages interleaving.

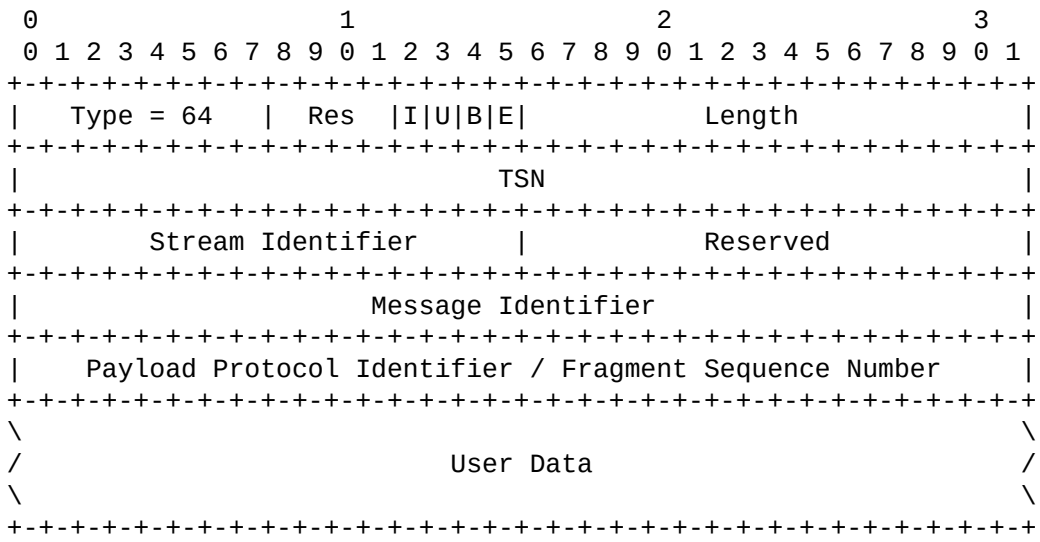

Figure 3: I-DATA chunk format

The only differences between the I-DATA chunk in Figure 3 and the DATA chunk defined in [RFC4960] and [RFC7053] is the addition of the new Message Identifier (MID) and Fragment Sequence Number (FSN) and the removal of the Stream Sequence Number (SSN). However, the lower 16-bit of the MID can be used as the SSN if necessary. The length of the I-DATA chunk header is 20 bytes, which is 4 bytes more than the length of the DATA chunk header defined in [RFC4960].

Reserved: 16 bits (unsigned integer)
This field is reserved. It MUST be set to 0 by the sender and MUST be ignored by the receiver.

Message Identifier (MID): 32 bits (unsigned integer)
The MID is the same for all fragments of a user message, it is used to determine which fragments (enumerated by the FSN) belong to the same user message. For ordered user messages, the MID is also used by the SCTP receiver to deliver the user messages in the correct order to the upper layer (similar to the SSN of the DATA chunk defined in [RFC4960]). The sender uses two counters for each outgoing streams, one for ordered messages, one for unordered messages. All counters are independent and initially 0. They are incremented by 1 for each user message. Please note that the serial number arithmetic defined in [RFC1982] using SERIAL_BITS = 32 applies. Therefore the sender MUST NOT have more than 2**31 - 1 ordered messages for each outgoing stream in flight and MUST NOT have more than 2**31 - 1 unordered messages for each outgoing stream in flight. Please note that the MID is in "network byte order", a.k.a. Big Endian.

Payload Protocol Identifier (PPID) / Fragment Sequence Number (FSN): 32 bits (unsigned integer)

If the B bit is set, this field contains the PPID of the user message. In this case the FSN is implicitly considered to be 0. If the B bit is not set, this field contains the FSN. The FSN is used to enumerate all fragments of a single user message, starting from 0 and incremented by 1. The last fragment of a message MUST have the 'E' bit set. Note that the FSN MAY wrap completely multiple times allowing arbitrary large user messages. For the FSN the serial number arithmetic defined in [RFC1982] applies with SERIAL_BITS = 32. Therefore a sender MUST NOT have more than 2**31 - 1 fragments of a single user message in flight. Please note that the FSN is in "network byte order", a.k.a. Big Endian.

2.2. Procedures

This subsection describes how the support of the I-DATA chunk is negotiated and how the I-DATA chunk is used by the sender and receiver.

2.2.1. Negotiation

A sender MUST NOT send a I-DATA chunk unless both peers have indicated its support of the I-DATA chunk type within the Supported Extensions Parameter as defined in [RFC5061]. If I-DATA support has been negotiated on an association, I-DATA chunks MUST be used for all user messages and DATA-chunk MUST NOT be used. If I-DATA support has not been negotiated on an association, DATA chunks MUST be used for all user messages and I-DATA chunks MUST NOT be used.

A sender MUST NOT use the I-DATA chunk unless the user has requested that use (e.g. via the socket API, see Section 4). This constraint is made since usage of this chunk requires that the application be willing to interleave messages upon reception within an association. This is not the default choice within the socket API (see [RFC6458]) thus the user MUST indicate support to the protocol of the reception of completely interleaved messages. Note that for stacks that do not implement [RFC6458] they may use other methods to indicate interleaved message support and thus enable the usage of the I-DATA chunk, the key is that the the stack MUST know the application has indicated its choice in wanting to use the extension.

2.2.2. Sender Side Considerations

Sender side usage of the I-DATA chunk is quite simple. Instead of using the TSN for fragmentation purposes, the sender uses the new FSN field to indicate which fragment number is being sent. The first fragment MUST have the 'B' bit set. The last fragment MUST have the

'E' bit set. All other fragments MUST NOT have the 'B' or 'E' bit set. All other properties of the existing SCTP DATA chunk also apply to the I-DATA chunk, i.e. congestion control as well as receiver window conditions MUST be observed as defined in [RFC4960].

Note that the usage of this chunk implies the late assignment of the actual TSN to any chunk being sent. Each I-DATA chunk uses a single TSN. This way messages from other streams may be interleaved with the fragmented message. Please note that this is the only form of interleaving support. For example, it is not possible to interleave multiple ordered or unordered user messages from the same stream.

The sender MUST NOT be fragmenting more than one ordered message in any one stream at any time. The sender MUST NOT be fragmenting more than one un-ordered user message in any one stream at any time. The sender MAY fragment one ordered and one unordered user message within a single stream. At any time a sender MAY fragment an ordered and an unordered user message each off them on different streams.

2.2.3. Receiver Side Considerations

Upon reception of an SCTP packet containing a I-DATA chunk if the message needs to be reassembled, then the receiver MUST use the FSN for reassembly of the message and not the TSN. Note that a non-fragmented messages is indicated by the fact that both the 'E' and 'B' bits are set. An ordered or unordered fragmented message is thus identified with any message not having both bits set.

2.3. Interaction with other SCTP Extensions

The usage of the I-DATA chunk might interfere with other SCTP extensions. Future SCTP extensions MUST describe if and how they interfere with the usage of I-DATA chunks. For the SCTP extensions already defined when this document was published, the details are given in the following subsections.

2.3.1. SCTP Partial Reliability Extension

When the SCTP extension defined in [RFC3758] is used, the the I-FORWARD-TSN chunk MUST be used instead of the FORWARD-TSN chunk. The only difference is that the 16-bit Stream Sequence Number (SSN) has been replaced by the 32-bit Message Identifier (MID).

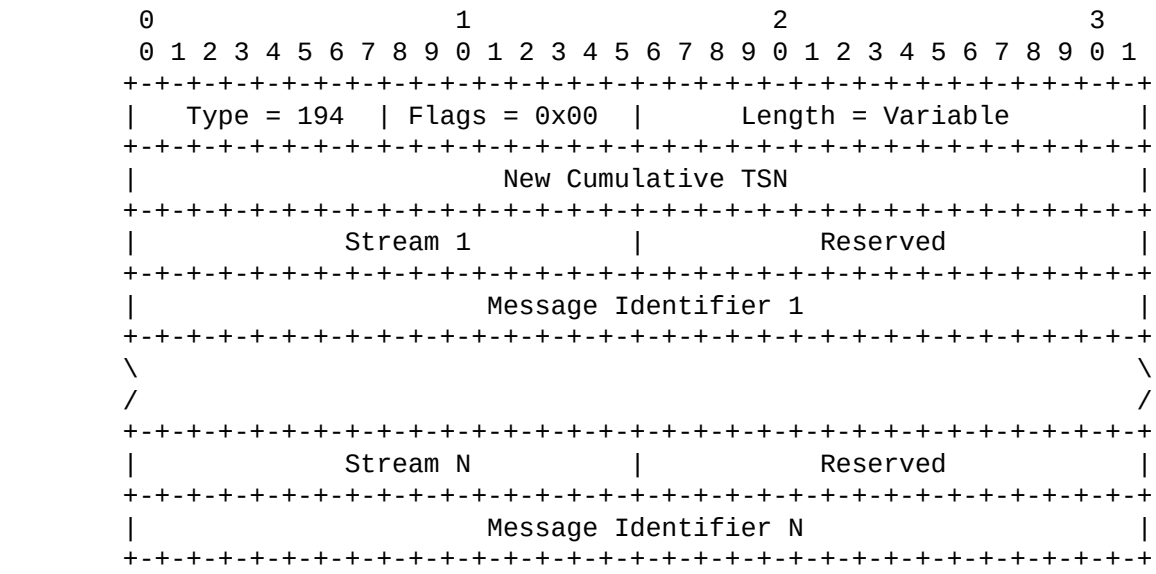

Figure 4: I-FORWARD-TSN chunk format

2.3.2. SCTP Stream Reconfiguration Extension

When an association resets the SSN using the SCTP extension defined in [RFC6525], the two counters (one for the ordered messages, one for the unordered messages) used for the MID MUST be reset to 0 correspondingly.

Since most schedulers require late TSN assignment, it should be noted that the implementation of [RFC6525] needs to handle this.

3. Stream Schedulers

This section defines several stream schedulers. The stream schedulers which can be used even without the user message interleaving support as defined in Section 2 are described in Section 3.1. In Section 3.2 stream schedulers requiring user message interleaving defined in Section 2 are described.

3.1. Stream Scheduler without User Message Interleaving Support

3.1.1. First Come First Serve (SCTP_SS_FCFS)

The simple first-come, first-serve scheduler of user messages is used. It just passes through the messages in the order in which they have been delivered by the application. No modification of the order is done at all.

3.1.2. Round Robin Scheduler (SCTP_SS_RR)

This scheduler provides a fair scheduling based on the number of user messages by cycling around non-empty stream queues.

3.1.3. Round Robin Scheduler per Packet(SCTP_SS_RR_PKT)

This is a round-robin scheduler but only bundles user messages of the same stream in one packet. This minimizes head-of-line blocking when a packet is lost because only a single stream is affected.

3.1.4. Priority Based Scheduler (SCTP_SS_PRIO)

Scheduling of user messages with strict priorities is used. The priority is configurable per outgoing SCTP stream. Streams having a higher priority will be scheduled first and when multiple streams have the same priority, the default scheduling should be used for them.

3.1.5. Fair Bandwidth Scheduler (SCTP_SS_FB)

A fair bandwidth distribution between the streams is used. This scheduler considers the lengths of the messages of each stream and schedules them in a certain way to maintain an equal bandwidth for all streams. The details are implementation specific.

3.1.6. Weighted Fair Queueing Scheduler (SCTP_SS_WFQ)

A weighted fair queueing scheduler between the streams is used. The weight is configurable per outgoing SCTP stream. This scheduler considers the lengths of the messages of each stream and schedules them in a certain way to use the bandwidth according to the given weights. The details are implementation specific.

3.2. Stream Scheduler with User Message Interleaving Support

3.2.1. Round Robin Scheduler (SCTP_SS_RR_INTER)

This scheduler is similar to the one described in Section 3.1.2, but based on I-DATA chunks instead of user messages.

3.2.2. Round Robin Scheduler per Packet (SCTP_SS_RR_PKT_INTER)

This scheduler is similar to the one described in Section 3.1.3, but based on I-DATA chunks instead of user messages.

3.2.3. Priority Based Scheduler (SCTP_SS_PRIO_INTER)

This scheduler is similar to the one described in Section 3.1.4, but based on I-DATA chunks instead of user messages.

3.2.4. Fair Bandwidth Scheduler (SCTP_SS_FB_INTER)

This scheduler is similar to the one described in Section 3.1.5, but based on I-DATA chunks instead of user messages.

3.2.5. Weighted Fair Queueing Scheduler (SCTP_SS_WFQ_INTER)

This scheduler is similar to the one described in Section 3.1.6, but based on I-DATA chunks instead of user messages. This scheduler is used for WebRTC Datachannels as specified in [I-D.ietf-rtcweb-data-channel].

4. Socket API Considerations

This section describes how the socket API defined in [RFC6458] is extended to allow applications to use the extension described in this document.

Please note that this section is informational only.

4.1. SCTP_ASSOC_CHANGE Notification

When an SCTP_ASSOC_CHANGE notification is delivered indicating a sac_state of SCTP_COMM_UP or SCTP_RESTART for an SCTP association where both peers support the I-DATA chunk, SCTP_ASSOC_SUPPORTS_INTERLEAVING should be listen in the sac_info field.

4.2. Socket Options

option name	data type	get	set
SCTP_INTERLEAVING_SUPPORTED	struct sctp_assoc_value	X	X
SCTP_STREAM_SCHEDULER	struct sctp_assoc_value	X	X
SCTP_STREAM_SCHEDULER_VALUE	struct sctp_stream_value	X	X

4.2.1. Enable or Disable the Support of User Message Interleaving (SCTP_INTERLEAVING_SUPPORTED)

This socket option allows the enabling or disabling of the negotiation of user message interleaving support for future associations. For existing associations it allows to query whether user message interleaving support was negotiated or not on a particular association.

User message interleaving is disabled per default.

This socket option uses IPPROTO_SCTP as its level and SCTP_INTERLEAVING_SUPPORTED as its name. It can be used with getsockopt() and setsockopt(). The socket option value uses the following structure defined in [RFC6458]:

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id: This parameter is ignored for one-to-one style sockets. For one-to-many style sockets, this parameter indicates upon which association the user is performing an action. The special sctp_assoc_t SCTP_FUTURE_ASSOC can also be used, it is an error to use SCTP_{CURRENT|ALL}_ASSOC in assoc_id.

assoc_value: A non-zero value encodes the enabling of user message interleaving whereas a value of 0 encodes the disabling of user message interleaving.

sctp_opt_info() needs to be extended to support SCTP_INTERLEAVING_SUPPORTED.

An application using user message interleaving should also set the fragment interleave level to 2 by using the SCTP_FRAGMENT_INTERLEAVE socket option specified in Section 8.1.20 of [RFC6458]. This allows the reception from multiple streams simultaneously. Failure to set this option can possibly lead to application deadlock. Some implementations might therefore put some restrictions on setting combinations of these values.

4.2.2. Get or Set the Stream Scheduler (SCTP_STREAM_SCHEDULER)

A stream scheduler can be selected with the SCTP_STREAM_SCHEDULER option for setsockopt(). The struct sctp_assoc_value is used to specify the association for which the scheduler should be changed and the value of the desired algorithm.

The definition of struct sctp_assoc_value is the same as in [RFC6458]:

```
struct sctp_assoc_value {
  sctp_assoc_t assoc_id;
  uint32_t assoc_value;
};
```

assoc_id: Holds the identifier for the association of which the scheduler should be changed. The special SCTP_{FUTURE|CURRENT|ALL}_ASSOC can also be used. This parameter is ignored for one-to-one style sockets.

assoc_value: This specifies which scheduler is used. The following constants can be used:

SCTP_SS_DEFAULT: The default scheduler used by the SCTP implementation. Typical values are SCTP_SS_FCFS or SCTP_SS_RR.

SCTP_SS_FCFS: Use the scheduler specified in Section 3.1.1.

SCTP_SS_RR: Use the scheduler specified in Section 3.1.2.

SCTP_SS_RR_PKT: Use the scheduler specified in Section 3.1.3.

SCTP_SS_PRIO: Use the scheduler specified in Section 3.1.4. The priority can be assigned with the sctp_stream_value struct. The higher the assigned value, the lower the priority, that is the default value 0 is the highest priority and therefore the default scheduling will be used if no priorities have been assigned.

SCTP_SS_FB: Use the scheduler specified in Section 3.1.5.

SCTP_SS_WFQ: Use the scheduler specified in Section 3.1.6. The weight can be assigned with the sctp_stream_value struct.

SCTP_SS_RR_INTER: Use the scheduler specified in Section 3.2.1.

SCTP_SS_RR_PKT_INTER: Use the scheduler specified in Section 3.2.2.

SCTP_SS_PRIO_INTER: Use the scheduler specified in Section 3.2.3. The priority can be assigned with the sctp_stream_value struct. The higher the assigned value, the lower the priority, that is the default value 0 is the highest priority and therefore the default scheduling will be used if no priorities have been assigned.

SCTP_SS_FB_INTER: Use the scheduler specified in Section 3.2.4.

SCTP_SS_WFQ_INTER: Use the scheduler specified in Section 3.2.5. The weight can be assigned with the sctp_stream_value struct.

4.2.3. Get or Set the Stream Scheduler Parameter (SCTP_STREAM_SCHEDULER_VALUE)

Some schedulers require additional information to be set for single streams as shown in the following table:

name	per stream info
SCTP_SS_DEFAULT	n/a
SCTP_SS_FCFS	no
SCTP_SS_RR	no
SCTP_SS_RR_PKT	no
SCTP_SS_PRIO	yes
SCTP_SS_FB	no
SCTP_SS_WFQ	yes
SCTP_SS_RR_INTER	no
SCTP_SS_RR_PKT_INTER	no
SCTP_SS_PRIO_INTER	yes
SCTP_SS_FB_INTER	no
SCTP_SS_WFQ_INTER	yes

This is achieved with the SCTP_STREAM_SCHEDULER_VALUE option and the corresponding struct sctp_stream_value. The definition of struct sctp_stream_value is as follows:

```
struct sctp_stream_value {
  sctp_assoc_t assoc_id;
  uint16_t stream_id;
  uint16_t stream_value;
};
```

assoc_id: Holds the identifier for the association of which the scheduler should be changed. The special SCTP_{FUTURE|CURRENT|ALL}_ASSOC can also be used. This parameter is ignored for one-to-one style sockets.

stream_id: Holds the stream id for the stream for which additional information has to be provided.

stream_value: The meaning of this field depends on the scheduler specified. It is ignored when the scheduler does not need additional information.

5. IANA Considerations

[NOTE to RFC-Editor:

"RFCXXXX" is to be replaced by the RFC number you assign this document.

]

[NOTE to RFC-Editor:

The suggested values for the chunk type and the chunk flags are tentative and to be confirmed by IANA.

]

This document (RFCXXXX) is the reference for all registrations described in this section.

A new chunk type has to be assigned by IANA. IANA should assign this value from the pool of chunks with the upper two bits set to '01'. This requires an additional line in the "Chunk Types" registry for SCTP:

ID Value	Chunk Type	Reference
64	New DATA chunk (I-DATA)	[RFCXXXX]

The registration table as defined in [RFC6096] for the chunk flags of this chunk type is initially given by the following table:

Chunk Flag Value	Chunk Flag Name	Reference
0x01	E bit	[RFCXXXX]
0x02	B bit	[RFCXXXX]
0x04	U bit	[RFCXXXX]
0x08	I bit	[RFCXXXX]
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

6. Security Considerations

This document does not add any additional security considerations in addition to the ones given in [RFC4960] and [RFC6458].

7. Acknowledgments

The authors wish to thank Christer Holmberg, Karen E. Egede Nielsen, Irene Ruengeler, Felix Weinrank, and Lixia Zhang for her invaluable comments.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644334 (NEAT). The views expressed are solely those of the author(s).

8. References

8.1. Normative References

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <http://www.rfc-editor.org/info/rfc1982>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <http://www.rfc-editor.org/info/rfc3758>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, DOI 10.17487/RFC5061, September 2007, <http://www.rfc-editor.org/info/rfc5061>.

[RFC6096] Tuexen, M. and R. Stewart, "Stream Control Transmission Protocol (SCTP) Chunk Flags Registration", RFC 6096, DOI 10.17487/RFC6096, January 2011, <http://www.rfc-editor.org/info/rfc6096>.

[RFC6525] Stewart, R., Tuexen, M., and P. Lei, "Stream Control Transmission Protocol (SCTP) Stream Reconfiguration", RFC 6525, DOI 10.17487/RFC6525, February 2012, <http://www.rfc-editor.org/info/rfc6525>.

[RFC7053] Tuexen, M., Ruengeler, I., and R. Stewart, "SACK-IMMEDIATELY Extension for the Stream Control Transmission Protocol", RFC 7053, DOI 10.17487/RFC7053, November 2013, <http://www.rfc-editor.org/info/rfc7053>.

8.2. Informative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <http://www.rfc-editor.org/info/rfc3261>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <http://www.rfc-editor.org/info/rfc6458>.

[I-D.ietf-rtcweb-data-channel] Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", draft-ietf-rtcweb-data-channel-13 (work in progress), January 2015.

Authors' Addresses

Randall R. Stewart
Netflix, Inc.
Chapin, SC 29036
United States

Email: randall@lakerest.net

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Salvatore Loreto
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Salvatore.Loreto@ericsson.com

Robin Seggelmann
Metafinanz Informationssysteme GmbH
Leopoldstrasse 146
80804 Muenchen
Germany

Email: rfc@robin-seggelmann.com