Internet Engineering Task Force (IETF)
Request for Comments: 8260
Category: Standards Track
ISSN: 2070-1721

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

November 2017

Stream Schedulers and User Message Interleaving for the Stream Control Transmission Protocol

Abstract

The Stream Control Transmission Protocol (SCTP) is a message-oriented transport protocol supporting arbitrarily large user messages. This document adds a new chunk to SCTP for carrying payload data. This allows a sender to interleave different user messages that would otherwise result in head-of-line blocking at the sender. The interleaving of user messages is required for WebRTC data channels.

Whenever an SCTP sender is allowed to send user data, it may choose from multiple outgoing SCTP streams. Multiple ways for performing this selection, called stream schedulers, are defined in this document. A stream scheduler can choose to either implement, or not implement, user message interleaving.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8260.

Copyright Notice

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 signaling messages in the style of Session Initiation Protocol (SIP) [RFC3261] may also need to use SCTP, and a message that is a single Maximum Transmission Unit (MTU) would be too small. Unfortunately this design decision, though valid at the time, did not account for other applications that might send large messages over SCTP. The sending of such large messages over SCTP, as specified in [RFC4960], can result in a form of sender-side head-of-line blocking (e.g., when the transmission of a message is blocked from transmission because the sender has started the transmission of another, possibly large, message). 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 up to 2**16 - 1 Stream Sequence Numbers (SSNs) outstanding.

The protocol requires all fragments of a user message to have consecutive TSNs. This document allows an SCTP sender to interleave different user messages.

This document also defines several stream schedulers for general SCTP associations allowing different relative stream treatments. The stream schedulers may behave differently depending on whether or not user message interleaving has been negotiated for the association.

Figure 1 illustrates the behavior of a round-robin stream scheduler using DATA chunks when three streams with the Stream Identifiers (SIDs) 0, 1, and 2 are used. Each queue for SID 0 and SID 2 contains a single user message requiring three chunks. The queue for SID 1 contains three user messages each requiring a single chunk. It is shown how these user messages are encapsulated in chunks using TSN 0 to TSN 8. Please note that the use of such a scheduler implies late

TSN assignment, but it can be used with an implementation that is compliant with [RFC4960] and that does not support user message interleaving. Late TSN assignment means that the sender generates chunks from user messages and assigns the TSN as late as possible in the process of sending the user messages.

```
+---+---+---+
|    0/0    |-+
+---+---+---+ |
              |  +---+---+---+---+---+---+---+---+---+
+---+---+---+ +->|1/2|1/1|2/0|2/0|2/0|1/0|0/0|0/0|0/0|
|1/2|1/1|1/0|--->|---|---|---|---|---|---|---|---|---|
+---+---+---+ +->| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
              |  +---+---+---+---+---+---+---+---+---+
+---+---+---+ |
|    2/0    |-+
+---+---+---+
                               +-------+
  +-------+                    |SID/SSN|
  |SID/SSN|                    |-------|
  +-------+                    |  TSN  |
                               +-------+
```

Figure 1: Round-Robin Scheduler without User Message Interleaving

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

In addition, the MID is also used for ensuring ordered delivery instead of using the stream sequence number (the I-DATA chunk omits an SSN).

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

```
+---+---+---+
|    0/0    |-+
+---+---+---+ |
              |  +-----+-----+-----+-----+-----+-----+-----+-----+-----+
+---+---+---+ +->|2/0/2|1/2/0|0/0/2|2/0/1|1/1/0|0/0/1|2/0/0|1/0/0|0/0/0|
|1/2|1/1|1/0|--->|-----|-----|-----|-----|-----|-----|-----|-----|-----|
+---+---+---+ +->|  8  |  7  |  6  |  5  |  4  |  3  |  2  |  1  |  0  |
              |  +-----+-----+-----+-----+-----+-----+-----+-----+-----+
+---+---+---+ |
|    2/0    |-+
+---+---+---+
                                     +-----------+
  +-------+                          |SID/MID/FSN|
  |SID/MID|                          |-----------|
  +-------+                          |    TSN    |
                                     +-----------+
```

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. DATA chunks are not permitted when I-DATA support has been negotiated. It should be noted that an SCTP implementation supporting I-DATA chunks needs to allow the coexistence of associations using DATA chunks and associations using I-DATA chunks.

In Section 2, this document specifies the user message interleaving by defining the I-DATA chunk, the procedures to use it, and its interactions with other SCTP extensions. Section 3 defines multiple stream schedulers, and Section 4 describes an extension to the socket API for using the mechanism specified in this document.

1.2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. User Message Interleaving

The protocol mechanisms described in this document allow the interleaving of user messages sent on different streams. They do not support the interleaving of multiple messages (ordered or unordered) sent on the same stream.

The interleaving of user messages is required for WebRTC data channels, as specified in [DATA-CHAN].

An SCTP implementation supporting user message interleaving is REQUIRED to support the coexistence of associations using DATA chunks and associations using I-DATA chunks. If an SCTP implementation supports user message interleaving and the Partial Reliability extension described in [RFC3758] or the Stream Reconfiguration Extension described in [RFC6525], it is REQUIRED to implement the corresponding changes specified in Section 2.3.

2.1. The I-DATA Chunk Supporting User Message Interleaving

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 64   |  Res  |I|U|B|E|       Length = Variable       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              TSN                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Stream Identifier      |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Message Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Payload Protocol Identifier / Fragment Sequence Number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                           User Data                           /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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] are the addition of the new Message Identifier (MID) and the new Fragment Sequence Number (FSN) and the removal of the Stream Sequence Number (SSN). The Payload Protocol Identifier (PPID), which is already defined for DATA chunks in [RFC4960], and the new FSN are stored at the same location of the packet using the B bit to determine which value is stored at the location. 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] and [RFC7053].

The old fields are:

Res: 4 bits
These bits are reserved. They MUST be set to 0 by the sender and MUST be ignored by the receiver.

I bit: 1 bit
The (I)mmediate Bit, if set, indicates that the receiver SHOULD NOT delay the sending of the corresponding SACK chunk. Same as the I bit for DATA chunks, as specified in [RFC7053].

U bit: 1 bit
The (U)nordered bit, if set, indicates the user message is unordered. Same as the U bit for DATA chunks, as specified in [RFC4960].

B bit: 1 bit
The (B)eginning fragment bit, if set, indicates the first fragment of a user message. Same as the B bit for DATA chunks, as specified in [RFC4960].

E bit: 1 bit
The (E)nding fragment bit, if set, indicates the last fragment of a user message. Same as the E bit for DATA chunks, as specified in [RFC4960].

Length: 16 bits (unsigned integer)
This field indicates the length in bytes of the DATA chunk from the beginning of the Type field to the end of the User Data field, excluding any padding. Similar to the Length for DATA chunks, as specified in [RFC4960].

TSN: 32 bits (unsigned integer)
This value represents the TSN for this I-DATA chunk. Same as the TSN for DATA chunks, as specified in [RFC4960].

Stream Identifier: 16 bits (unsigned integer)
Identifies the stream to which the user data belongs. Same as the Stream Identifier for DATA chunks, as specified in [RFC4960].

The new fields are:

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 stream: one for ordered messages and one for unordered messages. All of these 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. A message is considered in flight if at least one of its I-DATA chunks is not acknowledged in a way that cannot be reneged (i.e., not acknowledged by the cumulative TSN Ack). 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. Note that in this case, this field is not touched by an SCTP implementation; therefore, its byte order is not necessarily in network byte order. The upper layer is responsible for any byte order conversions to this field, similar to the PPID of DATA chunks. 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, thus allowing arbitrarily 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. A fragment is considered in flight if it is not acknowledged in a way that cannot be reneged. 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.

The handling of the I bit for the I-DATA chunk corresponds to the handling of the I bit for the DATA chunk described in [RFC7053].

2.2.1. Negotiation

An SCTP endpoint indicates user message interleaving support by listing the I-DATA chunk within the Supported Extensions Parameter, as defined in [RFC5061]. User message interleaving has been negotiated for an association if both endpoints have indicated I-DATA support.

If user message interleaving support has been negotiated for an association, I-DATA chunks MUST be used for all user messages and DATA chunks MUST NOT be used. If user message interleaving support has not been negotiated for an association, DATA chunks MUST be used for all user messages and I-DATA chunks MUST NOT be used.

An endpoint implementing the socket API specified in [RFC6458] MUST NOT indicate user message interleaving support unless the user has requested its use (e.g., via the socket API; see Section 4.3). This constraint is made since the usage of this chunk requires that the application is capable of handling interleaved messages upon reception within an association. This is not the default choice within the socket API (see the SCTP_FRAGMENT_INTERLEAVE socket option in Section 8.1.20 of [RFC6458]); thus, the user MUST indicate to the SCTP implementation its support for receiving completely interleaved messages.

Note that stacks that do not implement [RFC6458] may use other methods to indicate interleaved message support and thus indicate the support of user message interleaving. The crucial point is that the SCTP stack MUST know that the application can handle interleaved messages before indicating the I-DATA support.

2.2.2. Sender-Side Considerations

The 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 process (move user data into I-DATA chunks and assign a TSN to it) more than one user message in any given stream at any time. At any time, a sender MAY process multiple user messages, each of them on different streams.

The sender MUST assign TSNs to I-DATA chunks in a way that the receiver can make progress. One way to achieve this is to assign a higher TSN to the later fragments of a user message and send out the I-DATA chunks such that the TSNs are in sequence.

2.2.3. Receiver-Side Considerations

Upon reception of an SCTP packet containing an I-DATA chunk whose user message needs to be reassembled, the receiver MUST first use the SID to identify the stream, consider the U bit to determine if it is part of an ordered or unordered message, find the user message identified by the MID, and use the FSN for reassembly of the message and not the TSN. The receiver MUST NOT make any assumption about the TSN assignments of the sender. Note that a non-fragmented message is indicated by the fact that both the E and B bits are set. A message (either ordered or unordered) whose E and B bits are not both set may be identified as being fragmented.

If I-DATA support has been negotiated for an association, the reception of a DATA chunk is a violation of the above rules and therefore the receiver of the DATA chunk MUST abort the association by sending an ABORT chunk. The ABORT chunk MAY include the 'Protocol Violation' error cause. The same applies if I-DATA support has not been negotiated for an association and an I-DATA chunk is received.

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 in combination with the user message interleaving extension, the new I-FORWARD-TSN chunk MUST be used instead of the FORWARD-TSN chunk. The difference between the FORWARD-TSN and the I-FORWARD-TSN chunk is that the 16-bit Stream Sequence Number (SSN) has been replaced by the 32-bit Message Identifier (MID), and the largest skipped MID can also be provided for unordered messages. Therefore, the principle applied to ordered messages when using FORWARD-TSN chunks is applied to ordered and unordered messages when using I-FORWARD-TSN chunks.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 194  | Flags = 0x00  |      Length = Variable        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       New Cumulative TSN                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Stream Identifier       |          Reserved           |U|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Message Identifier                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
\                                                               \
/                                                               /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Stream Identifier       |          Reserved           |U|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Message Identifier                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: I-FORWARD-TSN Chunk Format

The old fields are:

Flags: 8 bits (unsigned integer)
These bits are reserved. They MUST be set to 0 by the sender and MUST be ignored by the receiver. Same as the Flags for FORWARD TSN chunks, as specified in [RFC3758].

Length: 16 bits (unsigned integer)
This field holds the length of the chunk. Similar to the Length for FORWARD TSN chunks, as specified in [RFC3758].

New Cumulative TSN: 32 bits (unsigned integer)
This indicates the New Cumulative TSN to the data receiver. Same as the New Cumulative TSN for FORWARD TSN chunks, as specified in [RFC3758].

The new fields are:

Stream Identifier (SID): 16 bits (unsigned integer)
This field holds the stream number this entry refers to.

Reserved: 15 bits
This field is reserved. It MUST be set to 0 by the sender and MUST be ignored by the receiver.

U bit: 1 bit
The U bit specifies if the Message Identifier of this entry refers to unordered messages (U bit is set) or ordered messages (U bit is not set).

Message Identifier (MID): 32 bits (unsigned integer)
This field holds the largest Message Identifier for ordered or unordered messages indicated by the U bit that was skipped for the stream specified by the Stream Identifier. For ordered messages, this is similar to the FORWARD-TSN chunk, just replacing the 16-bit SSN by the 32-bit MID.

Support for the I-FORWARD-TSN chunk is negotiated during the SCTP association setup via the Supported Extensions Parameter, as defined in [RFC5061]. The partial reliability extension is negotiated and can be used in combination with user message interleaving only if both endpoints indicated their support of user message interleaving and the I-FORWARD-TSN chunk.

The FORWARD-TSN chunk MUST be used in combination with the DATA chunk and MUST NOT be used in combination with the I-DATA chunk. The I-FORWARD-TSN chunk MUST be used in combination with the I-DATA chunk and MUST NOT be used in combination with the DATA chunk.

If I-FORWARD-TSN support has been negotiated for an association, the reception of a FORWARD-TSN chunk is a violation of the above rules and therefore the receiver of the FORWARD-TSN chunk MUST abort the association by sending an ABORT chunk. The ABORT chunk MAY include the ’Protocol Violation’ error cause. The same applies if I-FORWARD-TSN support has not been negotiated for an association and a FORWARD-TSN chunk is received.

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 MIDs MUST be reset to 0.

Since most schedulers, especially all schedulers supporting user message interleaving, 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 may behave differently depending on whether or not user message interleaving has been negotiated for the association. An implementation MAY implement any subset of them. If the implementation is used for WebRTC data channels, as specified in [DATA-CHAN], it MUST implement the Weighted Fair Queueing Scheduler defined in Section 3.6.

The selection of the stream scheduler is done at the sender side. There is no mechanism provided for signaling the stream scheduler being used to the receiver side or even for letting the receiver side influence the selection of the stream scheduler used at the sender side.

3.1. First-Come, First-Served Scheduler (SCTP_SS_FCFS)

The simple first-come, first-served 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. The usage of user message interleaving does not affect the sending of the chunks, except that I-DATA chunks are used instead of DATA chunks.

3.2. Round-Robin Scheduler (SCTP_SS_RR)

When not interleaving user messages, this scheduler provides a fair scheduling based on the number of user messages by cycling around non-empty stream queues. When interleaving user messages, this scheduler provides a fair scheduling based on the number of I-DATA chunks by cycling around non-empty stream queues.

3.3. Round-Robin Scheduler per Packet (SCTP_SS_RR_PKT)

This is a round-robin scheduler, which only switches streams when starting to fill a new packet. It bundles only DATA or I-DATA chunks referring to the same stream in a packet. This scheduler minimizes head-of-line blocking when a packet is lost because only a single stream is affected.

3.4. 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 scheduling between them is implementation dependent. When the scheduler interleaves user messages, the sending of large, lower-priority user messages will not delay the sending of higher-priority user messages.

3.5. Fair Capacity Scheduler (SCTP_SS_FC)

A fair capacity distribution between the streams is used. This scheduler considers the lengths of the messages of each stream and schedules them in a specific way to maintain an equal capacity for all streams. The details are implementation dependent. interleaving user messages allows for a better realization of the fair capacity usage.

3.6. 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 specific way to use the capacity according to the given weights. If the weight of stream S1 is n times the weight of stream S2, the scheduler should assign to stream S1 n times the capacity it assigns to stream S2. The details are implementation dependent. Interleaving user messages allows for a better realization of the capacity usage according to the given weights.

This scheduler, in combination with user message interleaving, is used for WebRTC data channels, as specified in [DATA-CHAN].

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. Exposure of the Stream Sequence Number (SSN)

The socket API defined in [RFC6458] defines several structures in which the SSN of a received user message is exposed to the application. The list of these structures includes:

struct sctp_sndrcvinfo
   Specified in Section 5.3.2 of [RFC6458] and marked as deprecated.

struct sctp_extrcvinfo
   Specified in Section 5.3.3 of [RFC6458] and marked as deprecated.

struct sctp_rcvinfo
Specified in Section 5.3.5 of [RFC6458].

If user message interleaving is used, the lower-order 16 bits of the MID are used as the SSN when filling out these structures.

4.2. SCTP_ASSOC_CHANGE Notification

When an SCTP_ASSOC_CHANGE notification (specified in Section 6.1.1 of [RFC6458]) 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 listed in the sac_info field.

4.3. 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

4.3.1. 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 for querying whether or not user message interleaving support was negotiated on a particular association.

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 zero 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 interleaving of user messages from different streams. Please note that it does not allow the interleaving of user messages (ordered or unordered) on the same stream. Failure to set this option can possibly lead to application deadlock. Some implementations might therefore put some restrictions on setting combinations of these values. Setting the interleaving level to at least 2 before enabling the negotiation of user message interleaving should work on all platforms. Since the default fragment interleave level is not 2, user message interleaving is disabled per default.

4.3.2. 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 of the association for 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.

SCTP_SS_RR: Use the scheduler specified in Section 3.2.

SCTP_SS_RR_PKT: Use the scheduler specified in Section 3.3.

SCTP_SS_PRIO: Use the scheduler specified in Section 3.4. 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 3.5.

SCTP_SS_WFQ: Use the scheduler specified in Section 3.6. The weight can be assigned with the sctp_stream_value struct.

sctp_opt_info() needs to be extended to support SCTP_STREAM_SCHEDULER.

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

Some schedulers require additional information to be set for individual 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

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 of the association for 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 identifier of 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.

sctp_opt_info() needs to be extended to support SCTP_STREAM_SCHEDULER_VALUE.

4.4. Explicit EOR Marking

Using explicit End of Record (EOR) marking for an SCTP association supporting user message interleaving allows the user to interleave the sending of user messages on different streams.

5. IANA Considerations

Two new chunk types have been assigned by IANA.

5.1. I-DATA Chunk

IANA has assigned the chunk type for this chunk from the pool of chunks with the upper two bits set to ’01’. This appears in the "Chunk Types" registry for SCTP as follows:

ID Value	Chunk Type	Reference
64	Payload Data supporting Interleaving (I-DATA)	RFC 8260

The registration table (as defined in [RFC6096]) for the chunk flags of this chunk type is initially as follows:

Chunk Flag Value	Chunk Flag Name	Reference
0x01	E bit	RFC 8260
0x02	B bit	RFC 8260
0x04	U bit	RFC 8260
0x08	I bit	RFC 8260
0x10	Unassigned	
0x20	Unassigned	
0x40	Unassigned	
0x80	Unassigned	

5.2. I-FORWARD-TSN Chunk

IANA has assigned the chunk type for this chunk from the pool of chunks with the upper two bits set to ’11’. This appears in the "Chunk Types" registry for SCTP as follows:

ID Value	Chunk Type	Reference
194	I-FORWARD-TSN	RFC 8260

The registration table (as defined in [RFC6096]) for the chunk flags of this chunk type is initially empty.

6. Security Considerations

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

It should be noted that the application has to consent that it is willing to do the more complex reassembly support required for user message interleaving. When doing so, an application has to provide a reassembly buffer for each incoming stream. It has to protect itself against these buffers taking too many resources. If user message interleaving is not used, only a single reassembly buffer needs to be provided for each association. But the application has to protect itself for excessive resource usages there too.

7. References

7.1. Normative References

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc3758>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://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, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc7053>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[DATA-CHAN]
Jesup, R., Loreto, S., and M. Tuexen, "WebRTC Data Channels", Work in Progress, draft-ietf-rtcweb-data-channel-13, January 2015.

[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, <https://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, <https://www.rfc-editor.org/info/rfc6458>.

Acknowledgments

The authors wish to thank Benoit Claise, Julian Cordes, Spencer Dawkins, Gorry Fairhurst, Lennart Grahl, Christer Holmberg, Mirja Kuehlewind, Marcelo Ricardo Leitner, Karen E. Egede Nielsen, Maksim Proshin, Eric Rescorla, Irene Ruengeler, Felix Weinrank, Michael Welzl, Magnus Westerlund, and Lixia Zhang for their invaluable comments.

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

Authors' Addresses

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

Email: randall@lakerest.net

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

Email: tuexen@fh-muenster.de

Salvatore Loreto
Ericsson
Torshamnsgatan 21
164 80 Stockholm
Sweden

Email: Salvatore.Loreto@ericsson.com

Robin Seggelmann
Metafinanz Informationssysteme GmbH
Leopoldstrasse 146
80804 Muenchen
Germany

Email: rfc@robin-seggelmann.com