CHAPTER 8

Extra Materials for Chapter 8

In this document, we discuss one topic that we mentioned briefly in Chapter 8: SCTP packets.

8.1 SCTP PACKETS

As we discussed in Chapter 8 of the textbook, an SCTP packet is made of a general header and several chunks.

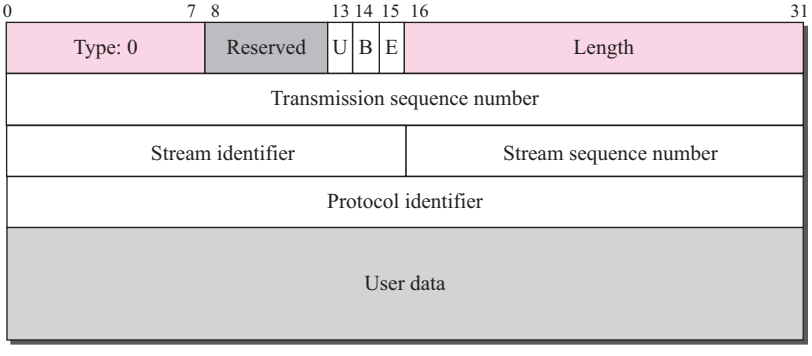
SCTP Chunks

SCTP defines fourteen types of chunks that we describe in this document.

DATA

The **DATA chunk** carries the user data. A packet may contain zero or more data chunks. Figure 8.1 shows the format of a DATA chunk.

Figure 8.1 *DATA chunk*



The descriptions of the common fields are the same. The type field has a value of 0. The flag field has 5 reserved bits and 3 defined bits: U, B, and E. The U (unordered) field, when set to 1, signals unordered data (explained later). In this case, the value of the stream sequence number is ignored. The B (beginning) and E (end) bits together

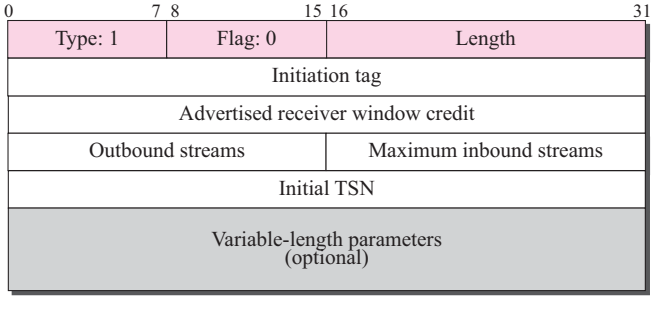
define the position of a chunk in a message that is fragmented. When B = 1 and E = 1, there is no fragmentation (first and last); the whole message is carried in one chunk. When B = 1 and E = 0, it is the first fragment. When B = 0 and E = 1, it is the last fragment. When B = 0 and E = 0, it is a middle fragment (neither the first nor the last). Note that the value of the length field does not include padding. This value cannot be less than 17 because a DATA chunk must always carry at least one byte of data.

- **Transmission sequence number (TSN).** This 32-bit field defines the transmission sequence number. It is a sequence number that is initialized in an INIT chunk for one direction and in the INIT ACK chunk for the opposite direction.
- **Stream identifier (SI).** This 16-bit field defines each stream in an association. All chunks belonging to the same stream in one direction carry the same stream identifier.
- **Stream sequence number (SSN).** This 16-bit field defines a chunk in a particular stream in one direction.
- **Protocol identifier.** This 32-bit field can be used by the application program to define the type of data. It is ignored by the SCTP layer.
- **User data.** This field carries the actual user data. SCTP has some specific rules about the user data field. First, no chunk can carry data belonging to more than one message, but a message can be spread over several data chunks. Second, this field cannot be empty; it must have at least one byte of user data. Third, if the data cannot end at a 32-bit boundary, padding must be added. These padding bytes are not included in the value of the length field.

INIT

The **INIT chunk** (initiation chunk) is the first chunk sent by an end point to establish an association. The packet that carries this chunk cannot carry any other control or data chunks. The value of the verification tag for this packet is 0, which means no tag has yet been defined. The format is shown in Figure 8.2.

Figure 8.2 *INIT chunk*



The three common fields (type, flag, and length) are as before. The value of the type field is 1. The value of the flag field is zero (no flags); and the value of the length

field is a minimum of 20 (more if there are optional parameters). The other fields are explained below:

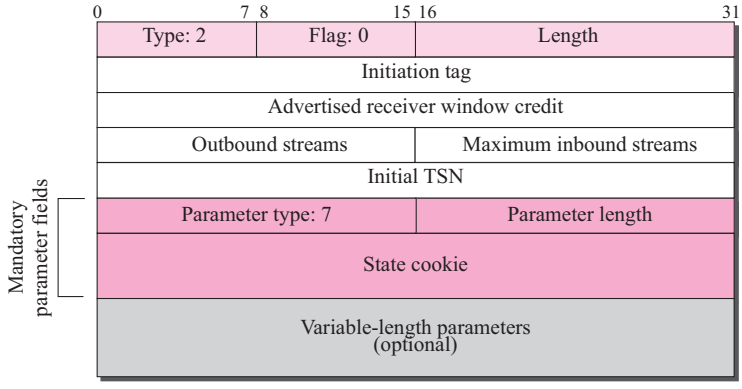
- **Initiation tag.** This 32-bit field defines the value of the verification tag for packets traveling in the opposite direction. As we mentioned before, all packets have a verification tag in the general header; this tag is the same for all packets traveling in one direction in an association. The value of this tag is determined during association establishment. The end point that initiates the association defines the value of this tag in the initiation tag field. This value is used as the verification tag in the rest of the packets sent from the other direction. For example, when end point A starts an association with end point B, A defines an initiation tag value, say *x*, which is used as the verification tag for all packets sent from B to A. The initiation tag is a random number between 0 and $2^{32} - 1$. The value of 0 defines no association and is permitted only by the general header of the INIT chunk.
- **Advertised receiver window credit.** This 32-bit field is used in flow control and defines the initial amount of data in bytes that the sender of the INIT chunk can allow. It is the *rwnd* value that will be used by the receiver to know how much data to send. Note that, in SCTP, sequence numbers are in terms of chunks.
- **Outbound stream.** This 16-bit field defines the number of streams that the initiator of the association suggests for streams in the outbound direction. It may be reduced by the other end point.
- **Maximum inbound stream.** This 16-bit field defines the maximum number of streams that the initiator of the association can support in the inbound direction. Note that this is a maximum number and cannot be increased by the other end point.
- **Initial TSN.** This 32-bit field initializes the transmission sequence number (TSN) in the outbound direction. Note that each data chunk in an association has to have one TSN. The value of this field is also a random number less than 2^{32}.
- **Variable-length parameters.** These optional parameters may be added to the INIT chunk to define the IP address of sending end point, the number of IP addresses the end point can support (multihome), the preservation of the cookie state, the type of addresses, and support of explicit congestion notification (ECN).

INIT ACK

The **INIT ACK chunk (**initiation acknowledgment chunk**)** is the second chunk sent during association establishment. The packet that carries this chunk cannot carry any other control or data chunks. The value of the verification tag for this packet (located in the general header) is the value of the initiation tag defined in the received INIT chunk. The format is shown in Figure 8.3.

Note that the fields in the main part of the chunk are the same as those defined in the INIT chunk. However, a mandatory parameter is required for this chunk. The parameter of type 7 defines the state cookie sent by the sender of this chunk. The chunk can also have optional parameters. Note that the initiation tag field in this chunk initiates the value of the verification tag for future packets traveling from the opposite direction.

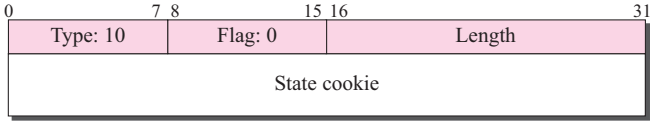
Figure 8.3 *INIT ACK chunk*



COOKIE ECHO

The **COOKIE ECHO chunk** is the third chunk sent during association establishment. It is sent by the end point that receives an INIT ACK chunk (normally the sender of the INIT chunk). The packet that carries this chunk can also carry user data. The format is shown in Figure 8.4.

Figure 8.4 *COOKIE ECHO chunk*

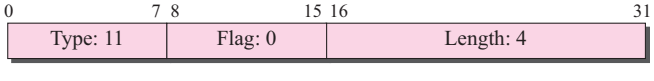


Note that this is a very simple chunk of type 10. In the information section it echoes the state cookie that the end point has previously received in the INIT ACK. The receiver of the INIT ACK cannot open the cookie.

COOKIE ACK

The **COOKIE ACK chunk** is the fourth and last chunk sent during association establishment. It is sent by an end point that receives a COOKIE ECHO chunk. The packet that carries this chunk can also carry user data. The format is shown in Figure 8.5.

Figure 8.5 *COOKIE ACK*

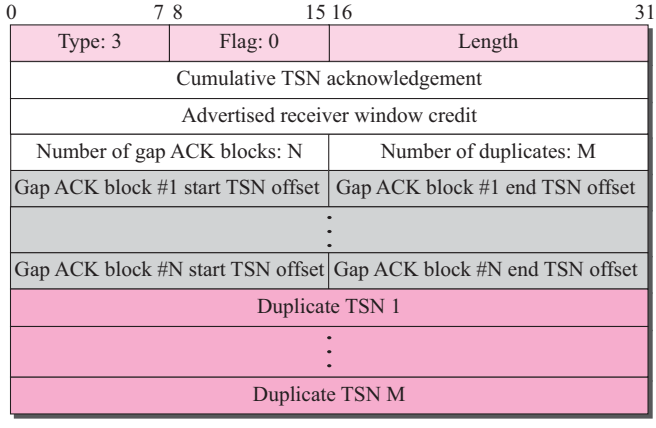


Note that this is a very simple chunk of type 11. The length of the chunk is exactly 4 bytes.

SACK

The **SACK chunk** (selective ACK chunk) acknowledges the receipt of data packets. Figure 8.6 shows the format of the SACK chunk.

Figure 8.6 *SACK chunk*



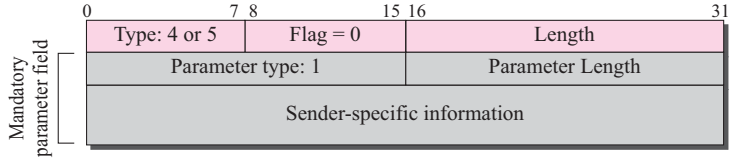
The common fields are the same as discussed previously. The type field has a value of 3. The flag bits are all set to 0s.

- **Cumulative TSN acknowledgment.** This 32-bit field defines the TSN of the last data chunk received in sequence.
- **Advertised receiver window credit.** This 32-bit field is the updated value for the receiver window size.
- **Number of gap ACK blocks.** This 16-bit field defines the number of gaps in the data chunk received after the cumulative TSN. Note that the term *gap* is misleading here: the gap defines the sequence of received chunks, not the missing chunks.
- **Number of duplicates.** This 16-bit field defines the number of duplicate chunks following the cumulative TSN.
- **Gap ACK block start offset.** For each gap block, this 16-bit field gives the starting TSN relative to the cumulative TSN.
- **Gap ACK block end offset.** For each gap block, this 16-bit field gives the ending TSN relative to the cumulative TSN.
- **Duplicate TSN.** For each duplicate chunk, this 32-bit field gives the TSN of the duplicate chunk.

HEARTBEAT and HEARTBEAT ACK

The **HEARTBEAT chunk** and **HEARTBEAT ACK chunk** are similar except for the type field. The first has a type of 4 and the second a type of 5. Figure 8.7 shows the format of these chunks. These two chunks are used to periodically probe the condition of an association. An end point sends a HEARTBEAT chunk; the peer responds with a HEARTBEAT ACK if it is alive. The format has the common three fields and mandatory parameter fields that provide sender-specific information. This information in the HEARTBEAT chunk includes the local time and the address of the sender. It is copied without change into the HEARTBEAT ACK chunk.

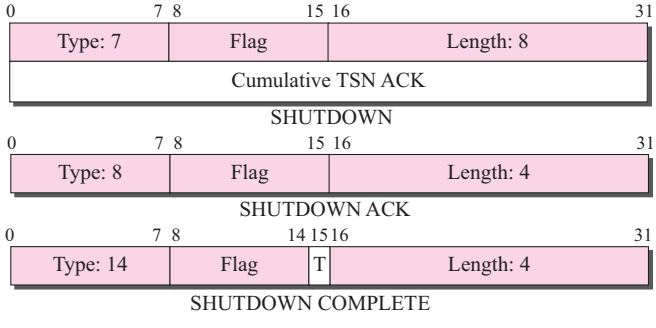
Figure 8.7 *HEARTBEAT and HEARTBEAT ACK chunks*



SHUTDOWN, SHUTDOWN ACK, and SHUTDOWN COMPLETE

These three chunks (used for closing an association) are similar. The **SHUTDOWN chunk,** type 7, is eight bytes in length; the second four bytes define the cumulative TSN. The **SHUTDOWN ACK chunk,** type 8, is four bytes in length. The **SHUTDOWN COMPLETE chunk,** type 14, is also 4 bytes long, and has a one bit flag, the T flag. The T flag shows that the sender does not have a TCB table. Figure 8.8 shows the formats.

Figure 8.8 *SHUTDOWN, SHUTDOWN ACK, and SHUTDOWN COMPLETE chunks*

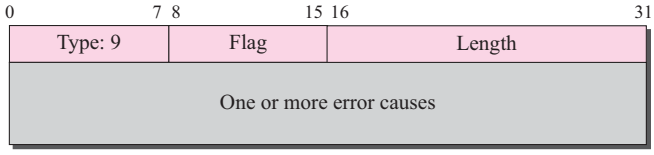


ERROR

The **ERROR chunk** is sent when an end point finds some error in a received packet. Note that the sending of an ERROR chunk does not imply the aborting of the associa-

tion. (This would require an ABORT chunk.) Figure 8.9 shows the format of the ERROR chunk.

Figure 8.9 *ERROR chunk*



The errors are defined in Table 8.1.

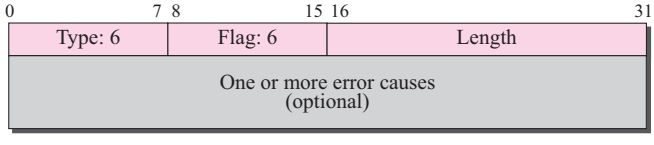
Table 8.1 *Errors*

Code	*Description*
1	Invalid stream identifier
2	Missing mandatory parameter
3	State cookie error
4	Out of resource
5	Unresolvable address
6	Unrecognized chunk type
7	Invalid mandatory parameters
8	Unrecognized parameter
9	No user data
10	Cookie received while shutting down

ABORT

The **ABORT chunk** is sent when an end point finds a fatal error and needs to abort the association. The error types are the same as those for the ERROR chunk (see Table 8.1). Figure 8.10 shows the format of an ABORT chunk.

Figure 8.10 *ABORT chunk*



FORWARD TSN

This is a chunk recently added to the standard (see RFC 3758) to inform the receiver to adjust its cumulative TSN. It provides partial reliable service.