SCTP States

This section describes the states that an instance of the SCTP protocol enters while an association is established, and when it is taken down again. There are a couple of concepts that will need to be explained here. The initialization of an association is completed on both sides after the exchange of four messages. The passive side (let's call it *server*) does not allocate resources until the third of these messages has arrived and been validated. That is to ensure that the association setup request really does originate from the right peer (without the possibility of blind spoofing).

Normal Association Establishment

The Server Side

The server receives an association setup request (an INIT chunk) usually in the CLOSED state, and analyzes the data contained in that chunk. From that it generates all the values needed at its side to enter an established association, and generates a secure hash of these values and a secret key (e.g. with the MD5 or SHA-1 algorithms). The values are then put into the so-called COOKIE, along with the derived message authentication code (MAC). This COOKIE is returned to the sender of the INIT chunk in an INIT-ACK chunk. The server remains in the CLOSED state, and forgets all about the received INIT chunk.

Upon reception of a COOKIE-ECHO chunk (which contains a COOKIE data structure as parameter), the server unpacks the data contained in this COOKIE, and uses again the MAC contained therein to verify whether it was the originator of this COOKIE. If the MAC computes okay, it is a valid COOKIE that this server had created before, and the data values contained in the COOKIE are used to initialize the SCTP instance. The server will send a COOKIE-ACK to the client (optionally bundling a data chunk with this COOKIE-ACK chunk) and enter the ESTABLISHED state. It is then ready to accept data or send data chunks itself.

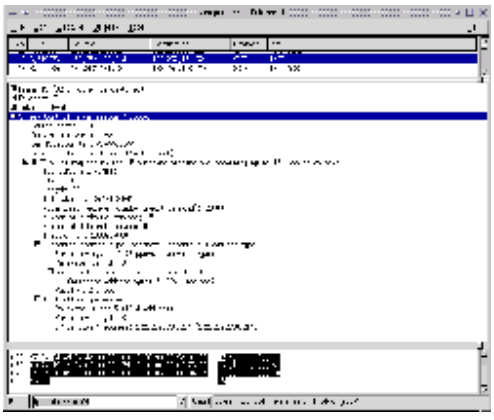
The Client Side

When an Upper Layer (ULP) wants to start an association, it calls the ASSOCIATE primitive (see SCTP API) and all necessary data structures are initialized in order to assemble an INIT chunk. This INIT chunk is sent to one transport address (i.e. combination of IP-address and port) of a server. An init timer is started that triggers repetitive sending of the INIT chunk when it expires before an INIT-ACK chunk was received from the server. If after a configurable number of send events no INIT-ACK was received, an error is reported to the ULP and peer endpoint is reported unreachable. After the client has sent the first INIT chunk, it enters the COOKIE-WAIT state.

When the client receives an INIT-ACK chunk from the server in the COOKIE-WAIT state, it stops the init timer, assembles an COOKIE-ECHO chunk, puts the server's COOKIE from the received INIT-ACK chunk into the COOKIE-ECHO chunk, and returns it to the server. It then starts a cookie timer, that triggers repetitive sending of this COOKIE-ECHO, until a COOKIE-ACK is received from the server. After sending the first COOKIE-ECHO, the protocol instance enters the COOKIE-ECHOED state. If no COOKIE-ACK is received after a configurable number of COOKIE-ECHO send events, the server endpoint is reported unreachable.

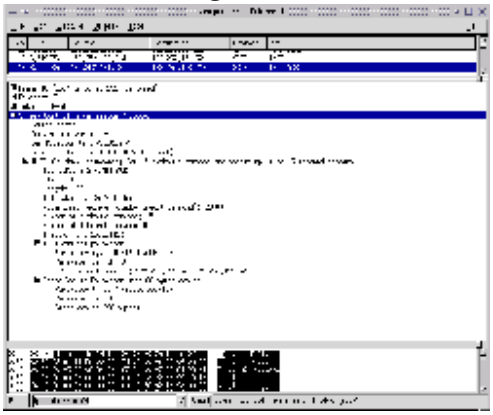
After reception of a COOKIE-ACK chunk from the server, the client enters the ESTABLISHED state. Note that the COOKIE-ECHO may already be accompanied by a bundled data chunk. It is up tp the server whether to accept that data chunk or to drop it.

Real Life Examples - Initialization



The picture to the right displays an INIT chunk that is sent by a client to a server. The image has been taken from the program Ethereal (also see software list on the SCTP Links page). For details, please select the image (e.g. by clicking on it). It displays an INIT chunk that has been sent by a host with the IP address 132.252.150.214 to one with the address 132.252.151.52. The INIT chunk carries the initiation tag 0x191c240f and requests 15 outbound and 15 inbound streams. Along with it, it carries the "Supported Address Types"-TLV

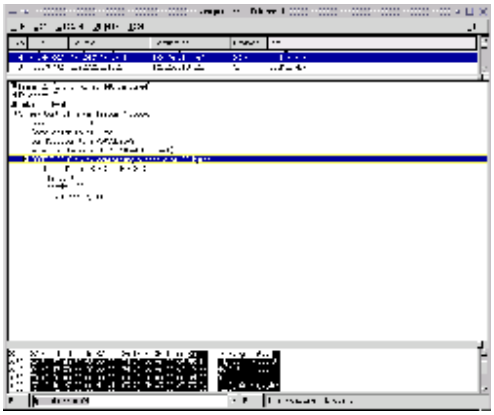
parameter as well as an address parameter with the IPv4 address of the client sending this chunk.

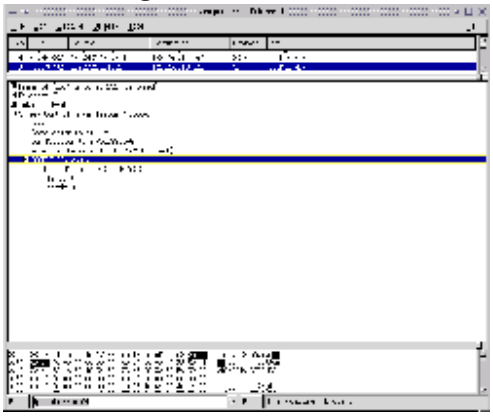


The response to an INIT chunk is a chunk that acknowledges it, and is called INIT-ACK chunk. Along with the INIT-ACK chunk, very similar parameters may be transmitted. In this example the server on IP address 132.252.151.52 returns his own tag 0x29e1115e in the INIT-ACK chunk, along with the number of inbound and outbound streams it is willing to accept (both are also 15 here).

Note that the tag in the SCTP common header is the same as the tag advertised by the client in its INIT chunk. Along with the INIT-ACK chunk a variable length parameter is sent, the so-called COOKIE data structure. The COOKIE must contain all the data that is needed by a server to initialize a new association. When this data structure is created, the server must not actually allocate any resources. Only when the very same COOKIE structure is returned to the server by the client that wants to establish the association, the server may actually allocate resources for the new association. To avoid possible tampering with any data contained in the COOKIE, this structure also contains a secure message authenticatin code (i.e. an MD5 or SHA-1 hash over the data structure **and** a secret key).

The client must then return the COOKIE data structure to the server in a so-called COOKIE-ECHO chunk. The COOKIE-ECHO chunk contains the very same variable length parameter as the INIT-ACK chunk. According to the RFC2960 the client may already transmit a DATA chunk after the COOKIE-ECHO chunk, which is not done here, though. As soon as the COOKIE-ECHO chunk is received by the server, it verifies the COOKIE structure (using the hash function and its secret key), and uses the data contained therein to initialize an appropriate association structure. It then notifies its user process with a COMMUNICATION-UP notification, and returns an acknowledgement to the client.





The server sends back a very simple acknowledgement of the COOKIE reception to the client, a so-called COOKIE-ACK chunk. Along with this chunk, it may already transmit data chunks, which is, however, not done here in the example to the left. After the client has received a COOKIE-ACK chunk, it will notify its user process of the successful association establishment with a COMMUNICATION-UP notification. At this point, both sides know that the association is established and may start normal data transmission and heartbeating (see the SCTP Data Transmission and SCTP Multihoming pages).

Association Termination

Both sides may decide to terminate an SCTP association for a number of reasons, and can do so practically at any time (provided they are in a state that is not CLOSED :-) There is the possibility of a graceful shutdown, ensuring that no data is lost, or hard termination, not taking care of the

peer.

Graceful Termination of an Association

Upon receiving the SHUTDOWN primitive from its upper layer user process, an SCTP instance should stop accepting data from this process, and start sending a SHUTDOWN chunk, as soon as all of its outstanding data has been acknowledged. This process is secured by a timer, that repeats this process, should the SHUTDOWN be lost.
The peer will, at one point, receive the SHUTDOWN, and reply by sending a SHUTDOWN ACK chunk, as soon as all of **its** data has been acknowledged (also secured by a timer !).
When the first peer (that started the shutdown procedure) receives the SHUTDOWN ACK, it will stop the timer, send a SHUTDOWN COMPLETE, and remove all data still belonging to that association, and enter the CLOSED state.
The peer that receives this SHUTDOWN COMPLETE chunk may then also remove all record of this association, and enter the CLOSED state.
Should the last SHUTDOWN COMPLETE message be lost, the peer will repeat sending SHUTDOWN ACK chunks, until an error counter has been exceeded, which reports the other peer unreachable.

Aborting the Association

An endpoint may also decide to abort an existing association, taking into account that data still in flight may not be acknowledged, by sending an an ABORT chunk to its peer endpoint. The sender MUST fill in the peer's Verification Tag in the outbound packet and MUST NOT bundle any DATA chunk with the ABORT.
The receiver of the ABORT does not reply, but validates the chunk, and removes the association, if the ABORT contains the correct tag value. If so, it also reports termination to its uppler layer process.
Should the ABORT be lost, and the endpoint sending it terminate directly after sending it, it will take a rather long time to determine that the peer has gone (i.e. after the Peer Error Counter has been exceeded).

Special Cases

There are a number of special cases that need to be considered. These occur, when one endpoint is interrupted, restarted etc.
Sections 5.2.4 and 9 of RFC2960 describe the handling of these cases:

- Peer restart case, where the peer uses a new tag value.
- Cross initialization, where both peers send an INIT chunk at about the same time.
- Excessive delay of COOKIE chunks, etc.
- One peer trying to re-establish an association, while the other one tries to terminate it.