**Assignment 1: RFC 9260: STREAM CONTROL TRANSMISSION PROTOCOL**

Course Code - Course Name: - COMP4039 – Network Foundations

Program: T433 - Cybersecurity

Section: A

Term: - Winter 2024

Group Number: 06

Student Names - ID:

* Prabhjot Singh Sains – 101495218
* Rahul Patel - 101378458
* Jai Deep Rawat - 101503760
* M. Salmaan Mustafa Shah – 10151007

Report by - Prabhjot Singh Sains

Source: <https://www.rfc-editor.org/rfc/rfc9260.html#name-association-initialization>

Images Source: <https://www.rfc-editor.org/rfc/rfc9260.html#name-association-initialization>

**Introduction**

SCTP was created for transport Public Switched Telephone Network (PSTN) which signals messages over IP networks. It operates over connectionless packet networks like IP, which provides acknowledged, error-free, non-duplicated data transfer, and data fragmentation which adapts to Path Maximum Transmission Unit (PMTU) size, and delivers user messages sequentially in multiple streams, allows optional bundling of messages into a single packet and supports network-level fault tolerance via multi-homing. Its design includes congestion avoidance and resistance to flooding and masquerade attacks. SCTP is suited for various applications like WebRTC.

TCP(RFC0793) plays a significant role in reliable data transfer in IP networks but some applications find it limited which is why applications have developed their protocols over UDP (RFC0768) to overcome the limitations. Some of these limitations include the strict order of data transmission and head-of-line blocking causing delays. Its stream-oriented nature is inconvenient, its socket scope complicates data transfer with multi-homed hosts. TCP is vulnerable to denial-of-service attacks, such as SYN attacks.

|  |  |
| --- | --- |
| Architectural View of SCTP | Functional View of SCTP |

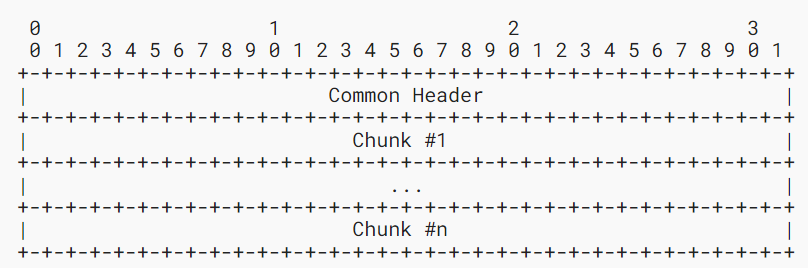
**Architectural View of SCTP**

SCTP is an intermediary layer between the SCTP user application and IP. It ensures that when the user sends a message, it reliably reaches the other side of SCTP peers without getting lost or mixed up.

**Functional View of SCTP**

1. **Association Startup and Takedown:** SCTP uses a four-way handshake. A user initializes the request which starts an association with other SCTP peers. Cookies are used to protect against SYN attacks. It supports both graceful and ungraceful close procedures, i.e. shutdown, and aborts responding to user requests or error conditions within the protocol.
2. **Sequenced Delivery within Streams:** Streams are user messages.SCTP ensures the sequential delivery of streams to the upper-layer protocol. Users can specify the number of streams during association startup. It allows some messages to bypass sequenced delivery.
3. **User Data Fragmentation:** SCTP can fragment user messages when required to fit within the Network’s maximum transmission unit. After fragmented the messages will remain fragmented until fragments are received and reassembled.
4. **Acknowledgement and Congestion Avoidance:** A Transmission Sequence Number (TSN) is assigned to each user data fragment which facilitates reliable delivery. Acknowledgments are sent for every received TSN. If a message needs to be retransmitted the TSN assigned to it is used which helps in congestion avoidance.
5. **Chunk Bundling:** The SCTP packets are made up of a common header followed by chunks of user data or SCTP control information. The user can choose to send multiple data chunks and have some control over delay in sending messages by requesting the SCTP implementation but it can still be delayed by flow or congestion controls.
6. **Packet Validation:** The common header in SCTP has a verification tag that protects against attacks during association initiation and a 32-bit Cyclic Redundancy Check (CRC32c) checksum that provides safety against data corruption within the network.
7. **Path Management:** The set of destination addresses for SCTP packets can be manipulated by the user. Path management monitors the reachability of addresses regularly, which sends notifications to the users of any changes that occur. It defines a communication path for each SCTP endpoint during startup and reports the eligible set of local transport addresses during association initiation, and addresses returned from the peer endpoint to the SCTP user. Based on user instructions the destination endpoints are chosen by the path management.

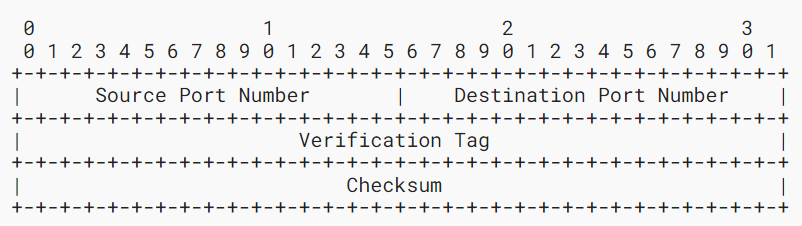
**SCTP Packet Format**



WS

A SCTP packet format consists of a common header and a chunk or multiple chunks. A user data message can be fragmented into multiple chunks if it does not fit into one SCTP packet.

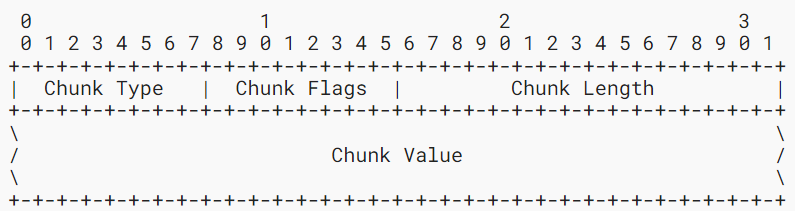
**SCTP Common Header Field Descriptions**



The SCTP packet header has four parts:

1. Source Port Number (16 bits): It is the sender port number that is used by the destination peer to establish the association. The source port number of 0 is not allowed.
2. Destination Port Number (16 bits): It is the port number for the destination host which is used by the receiver to de-multiplex the packet to the correct endpoint. The destination port number of 0 is not allowed.
3. Verification Tag (32 bits): It is used by the receiver to validate the sender. During transmission, its value is set to the Initiate Tag received from the peer endpoint during association initialization, with specific exceptions for INIT, SHUTDOWN COMPLETE, and ABORT chunks.
4. Checksum (32 bits): A 32-bit Cyclic Redundancy Check (CRC32c) checksum which ensures the integrity of the packet during transmission.

**Chunk Field Descriptions**



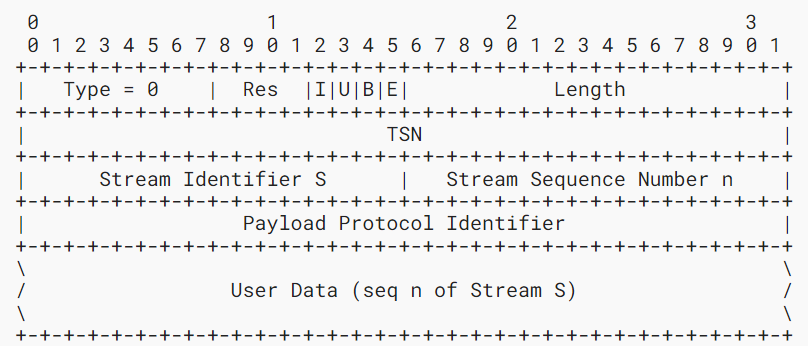
The SCTP packet structure includes the following chunk fields:

1. Chunk Type (8 bits): It ranges from 0 to 254 and identifies the type of information in the chunk field value. The image below shows the different chunk field values and their type.
2. Chunk Flags (8 bits): It depends on the chunk type specified in the chunk type field. It is set to 0 on transmission and ignored on receiving unless specified.
3. Chunk Length (16 bits): It represents the size of the chunk in bytes, including the chunk type, chunk flags, chunk length, and chunk value fields. The length field is set to 4 if the chunk value field is zero in length. The total chunk length must be a multiple of 4 bytes and padding.
4. Chunk Value (variable length): The actual information to be transferred in the chunks is contained in the chunk value. The usage and format depend on the specific chunk type.

|  |  |
| --- | --- |
|  |  |
| Processing of unknown chunks |
| Chunk Type |  |

If the chunk type is not recognized by the processing endpoint, then the 2-bit highest-order will specify the action to be taken. Below is the processing of unknown chunks.

**Payload Data**



The data chunk in SCTP comprises the following fields:

* Res (4 bits): Set to 0 on transmission and ignored on receipt.
* I bit (1 bit): It is called an immediate bit which is set by the sender whenever the user can benefit from the SACK chunk which is sent back without delay.
* U bit (1 bit): It is called an unordered bit which indicates an unordered DATA chunk with no assigned Stream Sequence Number if set to 1. They are sent to the upper layer without attempting to reorder.
* B bit (1 bit): It is called the Beginning fragment bit which specifies the beginning fragment of the user message if set.
* E bit (1 bit): It is called the Ending fragment bit which which specifies the ending fragment of the user message if set.
* Length (16 bits): It Indicates the length of the data chunk in bytes. The length field is set to 17 for a data chunk having one byte of user data.
* TSN (32 bits): It indicates the Transmission Sequence Number for the DATA chunk. It wraps back to 0 after reaching the maximum range of 232 – 1.
* Stream Identifier S (16 bits): It indicates which stream the user data belongs to.
* Stream Sequence Number n (16 bits): It is the Stream Sequence Number of the user data within the stream and it ranges between 0 to 65535.
* User Data (variable length): The payload of user data, padded to a 4-byte boundary with zero bytes. More than 3 bytes of padding must not be added by the sender.
* Payload Protocol Identifier (32 bits): It is an application-specified protocol identifier. It is passed to SCTP by the upper layer and sent to its sender for identifying the type of information in the DATA chunk. No application identifier is specified when its value is 0.

**SCTP association**

The normal establishment of an SCTP association involves the following steps:

* Initialization (A): SCTP endpoint "A" initiates the process by sending an INIT chunk to endpoint "Z" with its Verification Tag in the Initiate Tag field. “A” enters the COOKIE-WAIT state after starting the T1-init timer.
* Response (B): SCTP endpoint "Z" responds with an INIT ACK chunk, setting the Verification Tag field to Tag\_A and providing its own Verification Tag (Tag\_Z) in the Initiate Tag field. "Z" includes a State Cookie and does not allocate resources until the COOKIE ACK is received.
* Cookie Exchange (C): "A" receives the INIT ACK, stops the T1-init timer, and leaves COOKIE-WAIT. Then "A" sends the State Cookie in a COOKIE ECHO chunk, starts the T1-cookie timer, and enters COOKIE-ECHOED. The COOKIE ECHO may be bundled with outbound DATA chunks but must be the first chunk until COOKIE ACK is received.
* Acknowledgment (D): "Z" responds with a COOKIE ACK chunk after building a TCB and moving to the ESTABLISHED state. COOKIE ACK may be bundled with pending DATA chunks and must be the first chunk in the packet.
* Completion (E): "A" receives the COOKIE ACK, moves to the ESTABLISHED state, and stops the T1-cookie timer. Optionally, "A" notifies its upper-layer protocol (ULP) about the successful association establishment with a COMMUNICATION UP notification.

**Termination of Association**

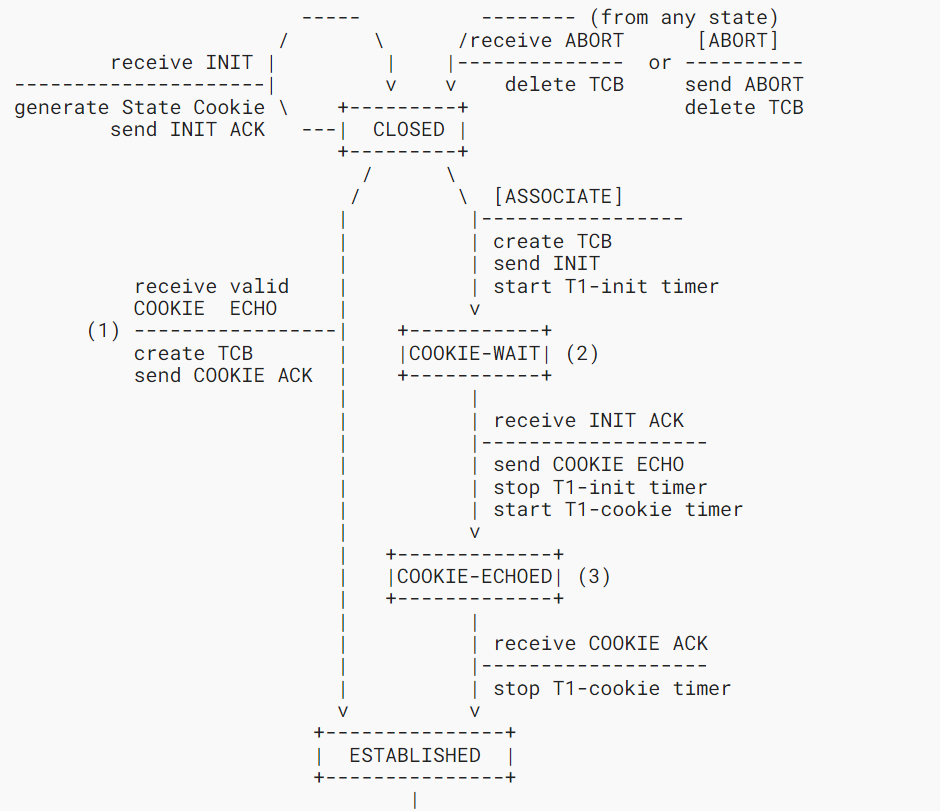
The best practice for an endpoint is to terminate its association when exiting a service. There are two ways to terminate services either abort the process which will discard any pending data or do a shutdown which will terminate the service after checking whether all the queued data reach their endpoint.

**Abort of an Association**

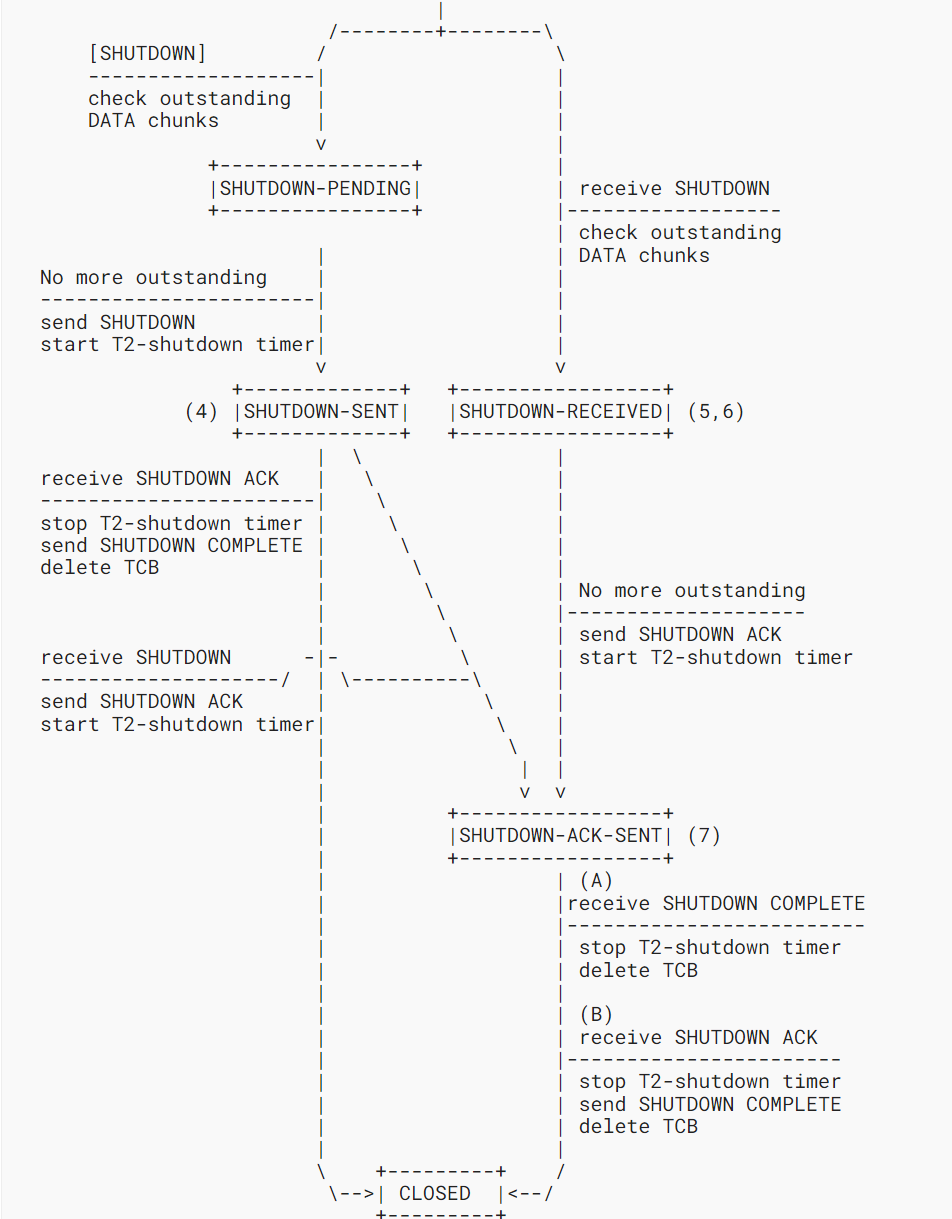
When an endpoint peer wants to terminate its association by abort, it sends an ABORT chunk to its peer which includes the peer’s Verification tag in its outbound packet. The endpoint will apply Verification Tag check rules upon receiving the ABORT chunk packet. After verifying the Verification Tag, the receiving endpoint removes the association and reports the termination to its upper layer.

**Shutdown of an Association**

The SHUTDOWN is initiated by the endpoint entering into the SHUTDOWN-PENDING state and will await the acknowledgment of all the outstanding data by its peer. After receiving acknowledgment of the data, the endpoint sends a SHUTDOWN chunk to its peer with the last TSN, starting the T2-shutdown timer and getting into the SHUTDOWN-SENT state. Upon receiving the SHUTDOWN chunk it enters the SHUTDOWN-RECEIVED state and stops accepting new data. The sender responds to received DATA-containing packets with a SHUTDOWN chunk and may employ a T5-shutdown-guard timer while in the SHUTDOWN-SENT state. If the receiver has no outstanding DATA chunks, it sends a SHUTDOWN ACK chunk, starts a T2-shutdown timer, and enters the SHUTDOWN-ACK-SENT state. The sender stops the T2-shutdown timer upon receiving the SHUTDOWN ACK, sends a SHUTDOWN COMPLETE chunk, and removes association records. Upon receiving the SHUTDOWN COMPLETE chunk, the receiver transitions to the CLOSED state.



SCTP Association



Termination of Association