Stream Control Transmission Protocol

[Reference: RFC 2960; ietf.org]

Features of SCTP

- Acked error-free non duplicated user data transfer
- Data fragmentation conforming to discovered path MTU (max transmission unit)
- Sequenced delivery of user messages within each stream (but have multiple streams)
- Option for out-of-order delivery of a user message
- Bundling of multiple user messages into a single SCTP packet
- Fault-tolerance (at network level) by supporting multi-homing at either/both ends of an association and heart-beat/keep alive messages
- Message boundaries preserved
- SACK
### Protocol Stack

<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
</tr>
<tr>
<td>IP</td>
</tr>
<tr>
<td>DL/MAC/Physical</td>
</tr>
</tbody>
</table>

---

### Need for SCTP (why not use TCP?)

- Head of line blocking in TCP; not good in many situations
- Applications must add their own record-making; TCP is stream (or byte) oriented
- Fault-tolerance: What if NIC with destination IP address fails?
- TCP vulnerable to Denial of Service attacks
- Main motivator:
  - Carrying SS7 signaling on IP Networks
Sample PSTN Network

Circuit Switched Network (Voice)

Packet Switched Network (Signaling)

SSP

A Sample SS7 Network

Packet Switched Network

Traffic Trunks (Carry voice)

Legend

SCP (Signaling Control Point)

STP (Signaling Transfer Point) (~ Router)

SSP (Signaling Switch Point)

S. Venkatesan  Department of Computer Science  2008
(Packet switched) SS7 Networks’s Needs

- Reliability:
  - Multiple parallel links; nodes in mated pairs
- Packet sequencing (between two switches) needed only within each call; not across all calls between two switches
- In VoIP, need to do signaling in IP networks. (Both voice and signaling messages on IP networks)

Sample Fig

What if a NIC fails?

- If using TCP, the connection is torn down and new connection (to alternate IP address) made. Lose all current calls being set up
- Using SCTP?
S. Venkatesan  Department of Computer Science  2008

An Example

Host 1

Process 1

Port 100

IP X1  IP Y1

Network X

Network Y

IP X2  IP Y2

Host 2

Process 2

Port 100

S. Venkatesan  Department of Computer Science  2008

SCTP Components

1. Association startup and teardown
2. Sequenced delivery within streams
3. User data fragmentation
4. Sack, congestion avoidance
5. Chunk bundling
6. Packet validation
7. Path management
**Association Initiation**

Host 1

- INIT chunk
- INIT-Ack (has cookie)
- Cookie-ECHO (data can be bundled)
- Cookie-ACK

Host 2

- Recreate TCB

Connection Established

**Data Transmission**

Host 1

- Data

Host 2

- SACK
SCTP Packet Format

- Common Header
- Chunk 1
- ... 
- Chunk n

Verification tag: Used by receiver to validate sender of SCTP packet
= Initiate Tag value (more on this later)

- Source Port #
- Destination Port #
- Verification Tag
- Alder-32 Checksum

SCTP Chunk

- Chunk Type (8 bits)
- Chunk Flags (8 bits)
- Chunk Length (16 bits)
- Chunk Data

Type:
0 = Payload data
1 = INIT
2 = INIT-ACK
3 = SACK
4 = Heart Beat Request
5 = Heart Beat ACK
6 = Abort
7 = Shut Down
8 = Shut Down Ack ...

Chunk Type is coded carefully.
High order 2 bits say what to do if type in chunk is undefined
- 00 = stop processing packet and discard packet
  (don’t process other chunks)
- 01 = same as above, report unrecognized type
- 10 = skip this chunk and continue processing
- 11 = same as 10, report unrecognized type
SCTP Chunk (Continued)

- Flags: 8 bits. Depends on type of chunk.
  - [When not needed, set all to 0]
- Length: >=4 (in bytes). Padded bytes not counted in length
- Depending on chunk type, there may be chunk-specific parameters [will see examples later]

<table>
<thead>
<tr>
<th>Parameter Type</th>
<th>Parameter Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter Value</td>
<td></td>
</tr>
</tbody>
</table>

Like chunk type, unrecognized parameter types are handled

INIT Chunk

<table>
<thead>
<tr>
<th>Type = 1</th>
<th>Flags = 0</th>
<th>Length (variable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Initiate tag *</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Advertised Receiver Window Credit</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em># of outbound streams</em></td>
<td><em># of inbound streams</em></td>
<td></td>
</tr>
<tr>
<td><em>Initial Transmit Sequence Number (TSN)</em></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Optional/variable length parameters

Initiate Tag: Unsigned 32 bit # [randomly chosen]
Receiver of INIT (responding end) records this.
Must be sent by receiver back to sender in every packet

Value = 0 => receiver aborts association
INIT Chunk: Continued

- Advertised Window Credit: Buffer size in Bytes. (Cannot be decreased)
- # of outbound and inbound streams should both be greater than 0
- Initial TSN: Sequence numbers (like in TCP)

Optional Variable Length Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>8 (4+4 Bytes)</td>
<td>IPv4 address</td>
</tr>
<tr>
<td>6</td>
<td>20 (4+16 Bytes)</td>
<td>IPv6 address</td>
</tr>
</tbody>
</table>

Incoming packets may be addressed to any one of the multiple IP addresses (specified in optional parameters)

This is multi homing

No IP addresses? Use sender’s IP address (from IP Packet)
### INIT Chunk: Continued

<table>
<thead>
<tr>
<th>Type = 11</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name (instead of IP address)</td>
<td></td>
</tr>
</tbody>
</table>

### INIT-Ack

<table>
<thead>
<tr>
<th>Type = 2</th>
<th>Flags</th>
<th>Chunk Length (variable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Initiate tag *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Advertised Receiver Window Credit *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* # of outbound streams *</td>
<td>* # of inbound streams *</td>
<td></td>
</tr>
<tr>
<td>* Initial Transmit Sequence Number (TSN) *</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| COOKIE parameter; others |

INIT-Ack must contain COOKIE parameter. COOKIE encrypts all state information needed to construct association, COOKIE creation time, and COOKIE life span.
INIT-Ack Continued

- COOKIE expires to protect against replay attacks
- Sender of COOKIE destroys all state
  - It will reconstruct later on receiving COOKIE on COOKIE-Echo

## COOKIE Echo
(S->R)

<table>
<thead>
<tr>
<th>Type</th>
<th>Chunk Flags</th>
<th>Length &gt;=4 (4+COOKIE L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

COOKIE

Data Chunks (Optional); may be bundled with COOKIE Echo

R unpacks COOKIE, reconstructs state, sends COOKIE-ACK to S and can begin accepting data
Flag Bits UBE are used to indicate:

U – Unordered Data
B – Beginning of Fragmented Message
E – End of Fragmented Message

A user message that fits in one chunk would have both the B and E bits set.

---

When one packet is lost, retransmission will occur in one of two ways:

- Repeated SACKs occur reporting the missing packet (via holes) 4 times.
- A time-out on the packet.

Receiver SACKs every packet when a hole exists.
Gaps

- Cumulative TSN is the highest consecutive TSN received (no gaps).
- All gaps/fragments reports describe what has been received.
- All gap/fragments numbers are offsets from the cumulative TSN.
- Retransmissions are made to alternate destinations if possible.

**SACK**

<table>
<thead>
<tr>
<th>Type = 3</th>
<th>Flags = 0</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cumulative TSN</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Advertised Receiver Window Credit</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n# of Gap acks</td>
<td># of Dup TSNS = m</td>
<td></td>
</tr>
<tr>
<td>Gap Ack Block 1 start</td>
<td>Gap Ack Block 1 end</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Gap ack block n start</td>
<td>Gap ack block n end</td>
<td></td>
</tr>
<tr>
<td>Dup TSN 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dup TSN m</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

- Received 200, 201, 202, gap, 210, 211, 212, gap, 218, 219, 220
- Cum TSN = 202
- Number of gaps = 2
- Start = 8; End = 10; \{received 202, 202+8 to 202+10\}
- Start = 16; End = 18

Sending Large Message

- Message size > path MTU
  1. Break message into path-MTU sized chunks
     - Overhead: IP/Data Chunk are common headers
  2. Assign these chunks TSNs in sequential order
  3. BE bits:
     - 10 = first chunk
     - 00 = middle pieces (pieces other than first and last)
     - 01 = last chunk
  4. Unfragmented messages have the BE flags set to 11.
  5. The TSN is useful in guaranteeing correct delivery.
Receiving Large Message

- Save each piece in a re-assembly queue.
- Use TSN to order the pieces.
- When all pieces are received (from 10 to 01) with no missing TSNs, merge all the data together.

Heart Beat and Fault Management

- HB sent on idle destinations at predetermined rate
- Receiver responds with Ack.
- Misses? Timeouts on HB?
  - > Threshold? Report destination is down
  - Report to upper layer
  - If down destination is primary, switch all traffic to an alternate IP address
State Diagram (1st Half)

- **Closed**
  - R: INIT
  - Generate Cookie & S: INIT_ACK
- **Cookie Wait**
  - R: valid COOKIE ECHO
  - Create TCB
  - S: COOKIE_ACK
- **Cookie Echo**
  - R: INIT ACK
  - Create TCB and S: INIT; Start INIT timer
  - S: COOKIE ECHO
  - Stop INIT timer; start COOKIE timer
- **Established**
  - R: INIT ACK
  - S: COOKIE ECHO
  - Stop COOKIE timer

State Diagram (2nd Half)

- **Established**
  - R: SD
  - Check outstanding data chunks
- **SD Pending**
  - No more outstanding Chunks; S: SD; Start timer
- **SD Sent**
  - R: SD Ack
  - Delete TCB
  - Stop timer
  - S: SD Complete
- **SD_ACK Sent**
  - R: SD Complete; Stop timer; Delete TCB
  - Or
  - R: SD ACK; stop timer; Delete TCB
  - S: SD Complete
- **Closed**
  - X :: R: SD
  - S: SD Ack; start timer
Retransmission Timer

- RTO
- Compute and manage RTO of each end point
- Like TCP

Security

- No protection to data contents
  - An attacker can sniff LAN, hijack connection, transmit bad packets, etc.
  - IP Sec can be used when protection against a sniffer is needed
- SCTP uses validation/verification tag to verify packets that belong to an association
  - Blind attacks are not possible
- COOKIE-Protect against SYN attacks
Congestion Control

- Slow start and congestion avoidance per association
- Uses:
  1. Receiver advertised window size
  2. Congestion control window- adjusted by sender based on observed network conditions
  3. Slow start threshold
Need these for each IP address (multi-homed)

Four NACKS (by SACK) => immediate transmission of missing packet (fast retransmit)