SCTP

Stream Control Transmission Protocol

Ing. Agustín Eijo <agu@frlp.utn.edu.ar>
Universidad Tecnológica Nacional Facultad Regional La Plata

Basado en: SCTP A detailed overview of the protocol and a examination of the socket API
http://www.sctp.org/SCTP_tutorial.ppt

Características de SCTP

- Transferencia de datos confiable con ACK selectivo (SACK)
- Control de flujo
- Delimitadores de Mensajes
- PMTU y fragmentación de mensajes
- Opción de envío de datos fuera de orden
- Soporte Multistream
- Soporte Multihoming
- Prevención contra ataques SYN flood (cookies)
- Escalable

Asociaciones y Endpoints

- Dos conceptos fundamentales de SCTP:
  - Endpoints (extremos de la comunicación)
  - Asociación (relación en la comunicación)
- Un SCTP endpoint puede ser representado como una lista de direcciones con un mismo puerto, ejemplo:
  - endpoint = [10.1.4.2, 10.1.5.3 : 80]

Asociación SCTP

- En TCP, la comunicación es denominada “conexión”
- En SCTP, esto es llamado “Asociación” dado que es un concepto más fuerte que una simple conexión (ej. multihoming)
- Una Asociación SCTP puede ser representada como un par de SCTP endpoints:
  - as = { [10.1.61.11 : 2223], [161.10.8.221, 120.1.1.5 : 80]}

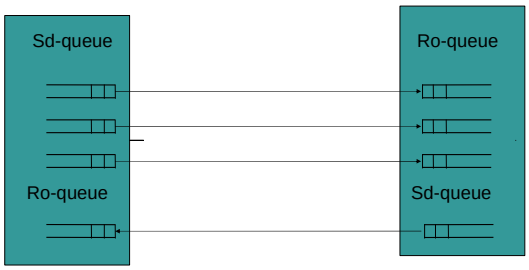
Asociación SCTP

- SCTP como TCP es orientado a la conexión.
- Durante el establecimiento de la asociación ambos extremos intercambian:
  - En TCP 3-way handshake
    (SYN, SYN/ACK, ACK)
  - En SCTP 4-way handshake
    (INIT, INIT-ACK, COOKIE-ECHO, COOKIE-ACK)

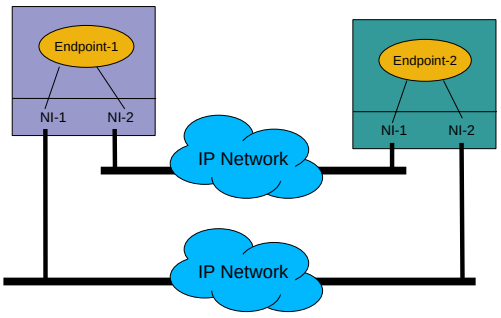
Asociación SCTP

- Una asociación SCTP provee transferencia de datos confiable de mensajes
- El envío de mensajes puede ser ordenado o fuera de orden
  - En el envío ordenado a cada mensaje se le asigna un stream sequence number (SSN)
  - En el envío fuera de orden los mensajes no tienen SSN

Multistream

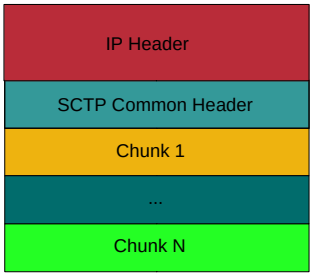


Multihoming

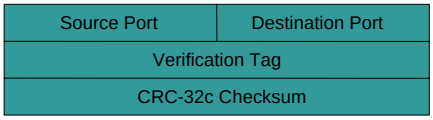


Mensajes SCTP

- Los mensajes SCTP tienen un header común seguido de uno o mas chunks

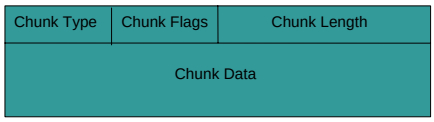


SCTP Common Header



- Source and Destination Port: números de port (16-bit)
- Verification Tag: valor aleatorio de 32-bit elegido por cada endpoint durante el establecimiento de la asociación
  - Permite diferenciar dos asociaciones consecutivas.
  - Mecanismo de detección de ataques.
- CRC32c Checksum: 32-bit Checksum cubre todo el mensaje SCTP (common header y todos los chunks) Mucho más robusto que el checksum de TCP

SCTP Chunks



- Chunk Type: 8-bit indica tipo de chunk.
- Chunk Flags: 8-bit flags, definidos por tipo de chunk.
- Chunk Length: 16-bit longitud en bytes, incluyendo chunk type, chunk flags, y chunk length.

Tipos de Chunk

- Hay 20 tipos de chunk actualmente definidos en SCTP (Incluyendo non-RFC/Internet Draft extensions):
  - (1) DATA (0x00)
  - (2) INITIATION [INIT] (0x01)
  - (3) INITIATION-ACKNOWLEDGMENT [INIT-ACK] (0x02)
  - (4) SELECTIVE-ACKNOWLEDGMENT [SACK] (0x03)
  - (5) HEARTBEAT (0x04)
  - (6) HEARTBEAT-ACKNOWLEDGMENT [HEARTBEAT-ACK] (0x05)
  - (7) ABORT (0x06)
  - (8) SHUTDOWN (0x07)

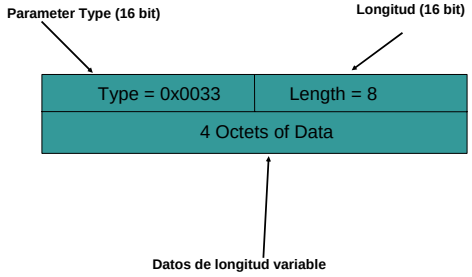
Tipos de Chunk

- (9) SHUTDOWN-ACKNOWLEGMENT [SHUTDOWN-ACK] (0x08)
- (10) OPERATIONAL-ERROR [ERROR] (0x09)
- (11) COOKIE-ECHO (0x0A)
- (12) COOKIE-ACKNOWLEDGMENT [COOKIE-ACK] (0x0B)
- (13) EXPLICIT CONGESTION NOTIFICATION ECHO [ECNE] (0x0C)
- (14) CONGESTION WINDOW REDUCE [CWR] (0x0D)
- (15) SHUTDOWN-COMPLETE (0x0E)

Tipos de Chunk (Extensiones)

- PR-SCTP - RFC 3758
  - (16) FORWARD-TSN (0xC0)
- ADD-IP draft
  - (17) ADDRESS-CONFIGURATION [ASCONF] (0xC1)
  - (18) ADDRESS-CONFIGURATION-ACKNOWLEDGMENT [ASCONF-ACK] (0x80)
- Packet-Drop draft
  - (19) SCTP-PACKET-DROP-REPORT [PKT-DROP] (0x81)
- Authentication draft
  - (20) AUTHENTICATION [AUTH] (0x82) - about to undergo drastic changes and will probably add 2-3 chunks.

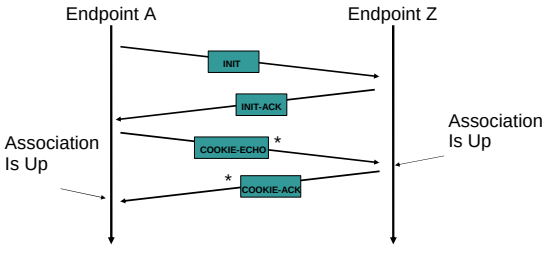
Formato de los parámetros



Establecimiento de una asociación

- SCTP usa 4-way handshake para establecer una asociación.
- El extremo que realiza la apertura activa envía un INIT chunk.
- El envío de INIT incluye varios parámetros ejemplos:
  - Direcciones IPv4 e IPv6
  - Soporte de extensiones como PR-SCTP (Partial Reliability)

Asociación completa

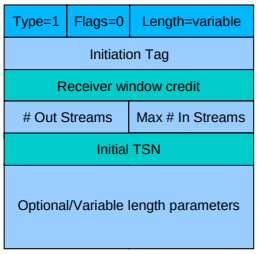


* -- pueden ser incluidos datos de usuario

Enviando INIT

- Dos importantes valores son enviados al enviar un INIT e INIT-ACK:
  - Verification Tag (V-Tag) valor que se utilizará en todos los envíos (ubicado en el campo initiate tag)
  - Initial TSN provee el punto de inicio (transport secuence space TSN)

INIT Chunk

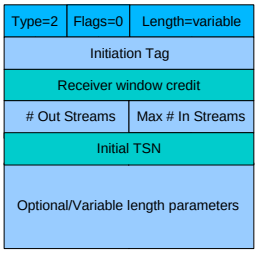


- Initiation Tag: valor aleatorio de 32-bit
- Receiver Window Credit: valor inicial de rwnd usado para el control de flujo
- # of Outbound Streams: número de streams que desea enviar el origen
- Max # of Inbound Streams: numero máximo de stream que el origen puede recibir
- Initial TSN: valor aleatorio de 32-bit TSN número inicial de secuencia

Recibiendo INIT

- El receptor del INIT valida que exista una aplicación escuchando. Y en caso de no existir envía un ABORT hacia el origen.
- Se realiza un chequeo y validación, enviando un INIT ACK sin guardar estado, esto previene ataques tipo TCP SYN.
- En el INIT-ACK, el receptor responde incluyendo varios parámetros como hizo el origen al enviar el INIT incluyendo uno importante state cookie.

INIT-ACK Chunk

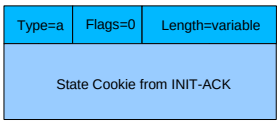


- Initiation Tag: valor aleatorio de 32-bit
- Receiver Window Credit: valor inicial de rwnd usado para el control de flujo
- # of Outbound Streams: número de streams que desea enviar el origen
- Max # of Inbound Streams: numero máximo de stream que el origen puede recibir
- Initial TSN: valor aleatorio de 32-bit TSN número inicial de secuencia

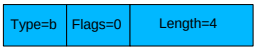
State cookie

- El parámetro state cookie:
  - Es firmado (generalmente con MD5 o SHA-1)
  - Cubre todos los parámetros necesarios para el establecimiento de una asociación (generalmente INIT completo y algunas partes de INIT-ACK)
  - Depende de la implementación, pero debe incluir el timestamp

Cookie Echo y Cookie Ack Chunk



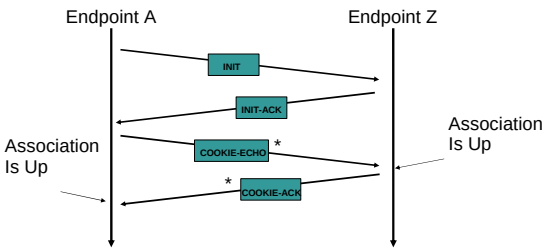
Cookie-Echo



Cookie-ACK

- Cookie-Echo y Cookie-ACK son chunks simples, pero ayudan a prevenir ataques
- Ambos permiten ser enviados junto a otros chunks, como DATA chunk

Asociación completa



* -- pueden ser incluidos datos de usuario

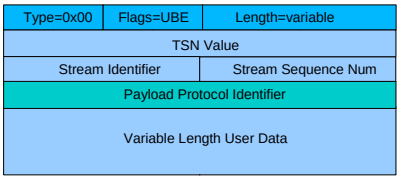
Recibiendo y enviando Mensajes

- El envío de mensajes se realiza con las funciones sndmsg() o sctp_sndmsg()
- Un simple rcvmsg() o sctp_rcvmsg() NUNCA devolverá mas de un mensaje
- Cuando se recibe un mensaje SCTP se deben procesar primero los chunks de control.
- Los chunks de datos deben ser entregados a la aplicación.
- Si un mensaje es fragmentado debe retenerse hasta estar completo.

Retransmisión

- Por cada DATA chunk enviado el receptor debe enviar un SACK chunk.
- En caso de no recibir el SACK Chunk en el tiempo establecido RTO (Retransmission Time Out) el origen puede retransmitir el DATA Chunk.
- SCTP mantiene estadísticamente el RTT (Round Trip Time) para cada asociación.

DATA Chunk

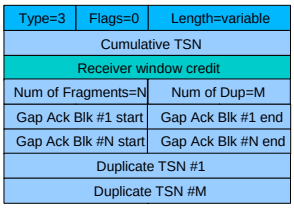


- Flags Bits 'UBE'
  - U – Datos fuera de orden
  - B – Comienzo de mensaje fragmentado
  - E – Fin de mensaje fragmentado
- Un mensaje completo debe tener los flags B y E bits activados

Campos DATA Chunk

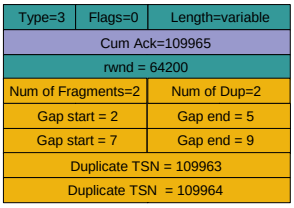
- TSN: número de secuencia utilizado para ordenar y reensamblar y retransmitir
- Stream Identifier: número de stream para los datos
- Stream Sequence Number: identifica el mensaje dentro del stream
- Payload Protocol Identifier: identificador del protocolo de la capa superior
- User Data: mensaje

SACK Chunk



- Cumulative TSN Acknowledgment: el mayor TSN consecutivo recibido
- Receiver Window Credit: valor actual de la ventana del receptor
- # of Fragments: cantidad de bloques faltantes
- # of Duplicates: cantidad de bloques duplicados
- Gap Ack Block Start / End TSN offset: comienzo y fin consecutivo de bloques faltantes relativo al TSN
- Duplicate TSN: TSN de bloque duplicado

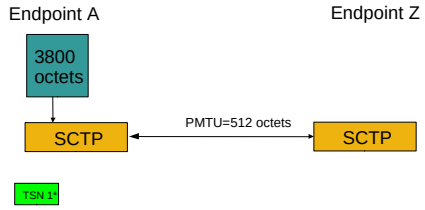
SACK Chunk Example



Que hacer cuando no se puede enviar un mensaje completo?

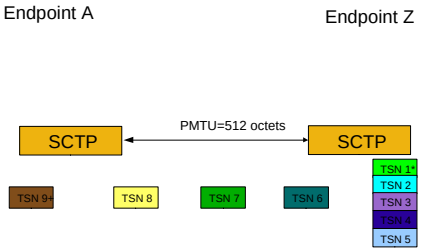
- Los mensajes deben dividirse en múltiples partes llamadas fragmentos.
- Todos los fragmentos deben utilizar el mismo Stream Identifier (SID) y Stream Sequence Number (SSN).
- Cada fragmento usa un único TSN y los flags apropiados para identificar si es el primero, ultimo, medio.

Transferencia de mensajes grandes



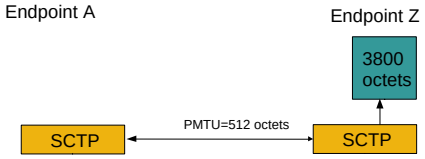
* - B bit set to 1

Transferencia de mensajes grandes



* - B bit set to 1
+ - E bit set to 1

Transferencia de mensajes grandes



Control de flujo

- Como en TCP, la ventana de congestión (cwnd) y el tamaño de ventana recibida (rwnd) se utilizan para el control de envío de datos.
- El origen no debe transmitir mas que rwnd.
- Si el receptor indica rwnd en 0, el orígen debe detener el envío de datos, sin embargo puede enviar mensajes esperando una actualización de rwnd

Control de Flujo

- Para evitar la congestión se utiliza la técnica Additive Increase Multiplicative Decrease (AIMD) cómo en TCP.
- Básicamente consiste en el incremento lento de cwnd, al detectar una pérdida de datos se decrementa cwnd rápidamente.

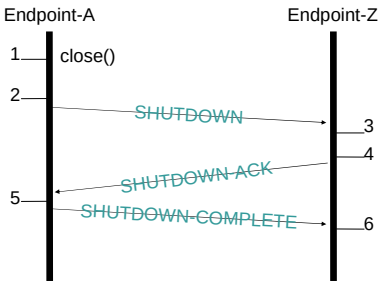
Orden y Streams

- SCTP permite el envío de datos en orden o fuera de orden.
  - En el envío fuera de orden los datos deben ser pasados inmediatamente a la aplicación.
  - En el envío de datos en orden los mensajes deben ser retenidos para ser entregados a la aplicación en orden.
- El origen indica en las funciones sndmsg() o sctp_sndmsg() que stream utilizar.

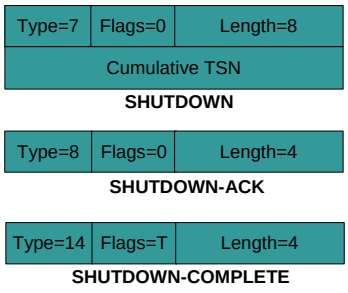
Cierre de una Asociación

- SCTP utiliza 3-way handshake para cerrar una asociación.
- A diferencia de TCP, SCTP NO soporta el cierre de un único extremo.

The Shutdown Handshake



Shutdown Chunks



ABORT Chunk

- Otro mecanismo de cierre es mediante la utilización de ABORT chunks.
- Cuando una aplicación realiza un cierre anormal inmediatamente es enviado un ABORT chunk.

Extensiones de SCTP

- SCTP, como hemos visto es un protocolo fácilmente extensible.
- Para extender el protocolo nuevos tipos de chunk o parámetros pueden ser agregados en nuevas RFC's.

Extensión PR-SCTP

- Partial Reliability SCTP permite omitir la retransmisión de mensajes al no recibir el acknowledged.
- Ambos extremos deben soportar la extensión esto se índica durante el establecimiento de la asociación.
- Normalmente se establece un timer esperando el acknowledged, cuando este tiempo expira el mensaje no es retransmitido.

Diferencias entre SCTP y TCP

- SCTP utiliza four-way handshake para establecer una asociación. TCP utiliza three-way handshake para establecer una conexión.
- Sin embargo, en SCTP puede enviarse datos en el tercer y cuarto mensaje de la asociación haciendo mas rápido el comienzo de transmisión en relación a TCP.
- SCTP utiliza cookies durante el four-way handshake evitando ataques SYN flooding.
- SCTP permite la transferencia multihoming y multistream.

Diferencias entre SCTP y TCP

- SCTP envía mensajes , no “byte stream”
  - Una aplicación TCP debe definir sus propios delimitadores de mensajes.
- SCTP permite el envío de datos fuera de orden.
- SCTP utiliza ACK selectivos en lugar de acumulativo como TCP.
- SCTP no permite el cierre en un solo sentido.

Socket API

```
int sd, newfd, sosz;
struct sockaddr_in6 sin6;
sosz = sizeof(sin6);
sd = socket(PF_INET6, SOCK_STREAM, 0);
sd = socket(PF_INET6, SOCK_STREAM, IPPROTO_SCTP);
listen(sd, 1);
while (1) {
    newfd = accept(sd, (struct sockaddr *)&sin6, &sosz)
    do_child_stuff(newfd, &sin6, sosz);
}
```

Nota: 0 implica IPPROTO_TCP

Ejemplo de servidor

```
int sd, newfd, sosz, msg_flags;
struct sockaddr_in6 sin6;
struct sndrcvinfo snd_rcv;
char buf[8000];
sosz = sizeof(sin6);
sd = socket(AF_INET6, SOCK_SEQPKT, IPPROTO_SCTP);
listen(sd, 1);
while (1) {
    len = sctp_recvmsg(sd, buf, sizeof(buf), (sockaddr *)&sin6, &sosz,
                       &snd_rcv, &msg_flags);
    do_child_stuff(newfd, buf, len, &sin6, &snd_rcv, msg_flags);
}
```

Referencias

- http://www.sctp.org
- RFC 2960 - Stream Control Transmission Protocol.
- RFC 3286 - An Introduction to SCTP.
- RFC 3578 - Partial Reliability.
- RFC 3257 - Stream Control Transmission Protocol Applicability Statement.