Implementación de protocolo SCTP en JAVA

César Reyes
Estudiante de Ing. Civil Electrónica UTFSM
e-mail: cesar.reyesp@alumnos.usm.cl

13 de enero de 2014

1. Introducción

Este es un proyecto desarrollado para el ramo programación de sistemas (ELO330) el cual esta orientado a investigar más en profundidad un protocolo "Stream Control Transmicion Protocol (SCTP).estudiado en clases. Existen varias aplicaciones que actualmente usan este protocolo como por ej la telefonía IP, particularmente este proyecto busca implementar un servicio Cliente-Servidor usando este protocolo en JAVA y poder mostrar las herramientas que facilita este lenguaje para el desarrollo de nuevas aplicaciones con este protocolo.
Además se busca poder dejar registro de aplicaciones escalables para el futuro, dado que actualmente no se encuentra una gran cantidad de material demostrativo que facilite el desarrollo de nuevas aplicaciones y la migraciones de varias plataformas a este protocolo.

2. ¿Qué es SCTP?

Stream Control Transmission Protocol (SCTP) es un protocolo de comunicación de capa de transporte que fue definido por el grupo SIGTRAN de IETF en el año 2000. El protocolo está especificado en la RFC 2960, y la RFC 3286. Este protocolo posee algunas características similares a UDP y TCP, pero además reúne otras características y otras funcionalidades prácticas como el multi-homing y el multi-streaming, mejorar en seguridad entre otras cosas. SCTP es orientado a la coneccion (similar a TCP), a una asociación entre los extremos la cual debe establecerse antes de transmitir datos y también orientada a mensajes (similar a los que hace UTP).

3. Características

SCTP es una alternativa a los protocolos de transporte TCP y UDP pues provee confiabilidad, control de flujo y secuenciación como TCP. Sin embargo, SCTP opcionalmente permite el envío de mensajes fuera de orden y a diferencia de TCP, SCTP es un protocolo orientado al mensaje (similar al envío de datagramas UDP).

- Soporte de multihoming
- Soporte de multistreaming
- Orientado a la coneccion
- Delimitadores de mensajes
- Provee servicio de mensaje no ordenados y confiables (Ordering)

- Mecanismos de validación y asentimiento como protección ante ataques por inundación, proveyendo notificación de trozos de datos duplicados o perdidos.
- Entrega de los datos en trozos que forman parte de flujos independientes y paralelos —eliminando así el problema de head of the line blocking que sufre TCP.

3.1. Asociaciones y Endpoints

En SCTP hay 2 conceptos claves y que lo hacen diferente a sus pares estos son los Endpoints (extremos de comunicación) y la Asociación (relacion de comunicación).
Un SCTP endpoint puede ser representado como una lista de direcciones con el mismo puerto, por ej:

$$endpoints = [10{,}1{,}4{,}2, 10{,}1{,}5{,}3 : 80]$$

La Asociación es el análogo a lo que en TCP sería denominado çonexión", pero en SCTP es llamado asociación dado que es un concepto más fuerte que una simple conexión, referentemente a la aplicación de Multihoming. Una Asociación en SCTP puede ser representada como un par de Endpoints:

$$as = \{[10{,}1{,}4{,}2, 10{,}1{,}5{,}3 : 80], [161{,}10{,}8{,}221 : 4567]\}$$

Como se dijo anteriormente este protocolo es orientado a la conexión, durante el establecimiento de la asociación ambos extremos intercambian 4-way handshake (INIT, INIT-ACK, COOKIE-ECHO, COOKIE-ACK). Esta asociación permite transferencias de datos confiables de mensajes, siendo estos ordenados o fuera de orden.

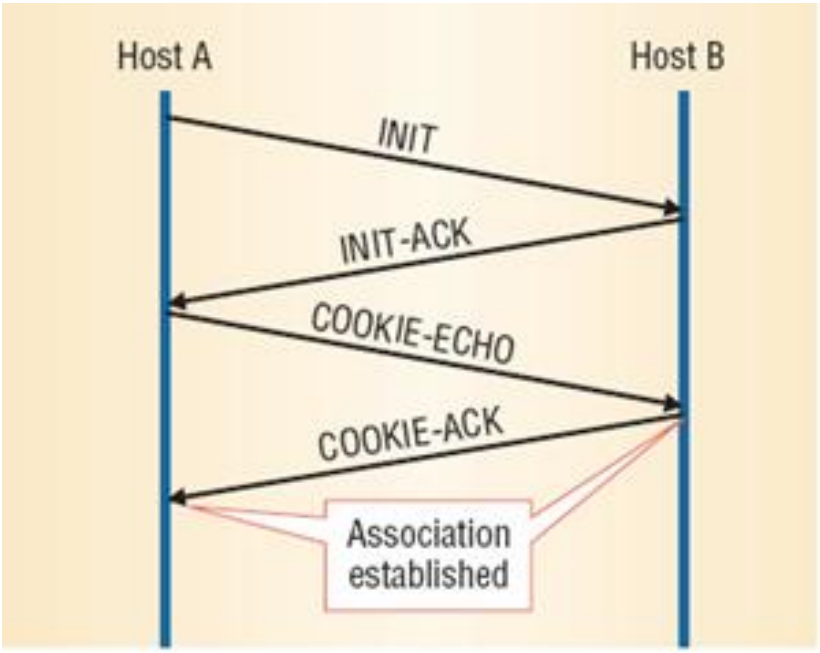

Figura 1: SCTP 4-way handshake. [1]

Lo importante de esto es que cuando el servidor envía el init-ack, dentro de este mensaje va una cookie con toda la información de seguridad, para que cuando un cliente envié el cookie-echo en ella venga de vuelta la misma cookie con la información entregada por el servidor, y así se identifique

[1] http://www2006.org/programme/files/xhtml/2035/2035-natarajan-xhtml/2035-natarajan_files/image003.jpg

correctamente al cliente como el que mando la solicitud de conexión.
Otra ventaja en cuanto a seguridad de este handshake, es que cuando el servidor envía el init-ack, no se queda en espera de respuesta por parte del cliente, por lo que no existe un tiempo de espera ni tampoco se le reservan recursos antes de tiempo.

Otra particularidad es que desde el envió del cookie-echo por parte del cliente, se pueden enviar datos dentro de este mensaje, si es que el tamaño de la información enviada (tamaño del chunk) cabe en el espacio "sobrante".

3.2. Opciones de orden

TCP básico contiene una sola forma de transmisión. La cual es ordenada y descarta todos los paquetes que no lleguen en orden. Este sistema en si no es muy práctico ya que una pérdida de 1 paquete puede desencadenar la pérdida de muchos más.

SCTP propone alternativas. La cual puede trabajar de forma no ordenada usando canales o mejor dicho MultiStream. Aunque cada canal mantiene conceptos de sistema ordenado, y generando los mismos tipos de pérdidas, el sistema total de los canales no es ordenado.

Ejemplo, suponemos la transferencia de 4 paquetes, y se transfieren en 4 canales diferentes. La pérdida de uno de estos paquetes no daña la transferencia de los otros y el orden de llegada de los paquetes puede variar. Pero la pérdida de un paquete puede causar la pérdida de los paquetes posteriores destinados a su mismo canal.

4. Estructura de paquete SCTP

SCTP tiene un paquete similar a la estructura de TCP, esta cuenta con 2 secciones, una cabecera común y una pila de bloques llamado Chunks. A su vez cada Chunk posee su propios campos de información, estos están limitados al MTU de la red.

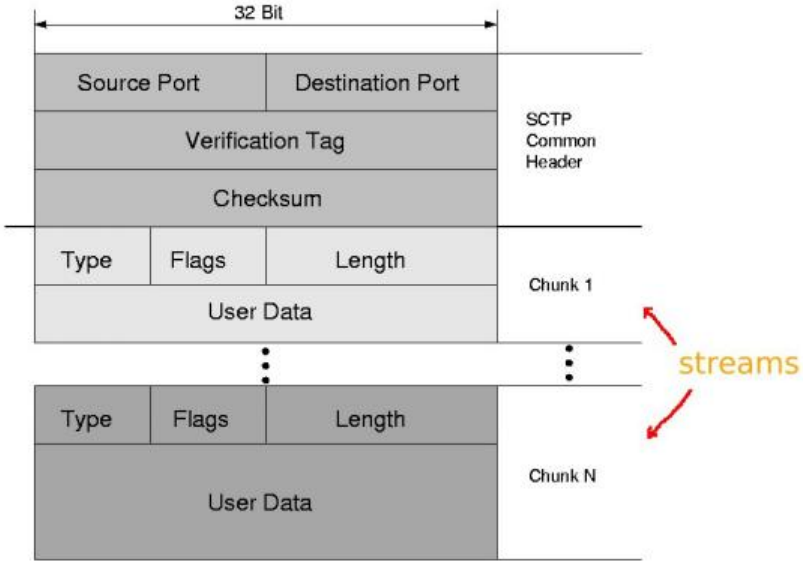

Figura 2: Estructura de datos SCTP [2]

La cabecera común o SCTP Common Header está compuesta por los siguientes campos:

Source Port: Indica el puerto fuente de la asociación.

Destination Port: Indica el puerto destino de la asociación.

Verification Tag: Valor aleatorio escogido por el cliente y el servidor de forma independiente uno de otro. Su utilidad es discriminar entre una asociación y otra, como también otros temas de seguridad.

Checksum CRC: Chequeo de datos CRC. Este chequeo se realiza al paquete completo (SCTP Common Header y los chunks).

Ademáss, cada chunk posee los siguientes campos:

Chunk Type: Indica el tipo de chunk que se va a enviar. Los tipos de chunks según su type se muestran más adelante.

Chunk Flags: Indica el flag que posee cada chunk. Este valor depende del chunk type debido a que cada tipo posee sus propios flags. En caso por ejemplo de un chunk de Data, sus flags indican si es que es el primer segmento del mensaje, el último o si es que los datos deben ser enviados en orden.

Chunk Length: Largo del Chunk completo en bytes.

Chunk Data: Data que se enviará por el chunk.

Algunos Chunk types:

- DATA (0x00)
- INITIATION [INIT] (0x01)
- INITIATION-ACKNOWLEDGMENT [INIT-ACK] (0x02)
- SELECTIVE-ACKNOWLEDGMENT [SACK] (0x03)
- HEARTBEAT (0x04)
- HEARTBEAT-ACKNOWLEDGMENT [HEARTBEAT-ACK] (0x05)
- ABORT (0x06)
- SHUTDOWN (0x07)
- SHUTDOWN-ACKNOWLEGMENT [SHUTDOWN-ACK] (0x08)
- OPERATIONAL-ERROR [ERROR] (0x09)
- COOKIE-ECHO (0x0A)
- COOKIE-ACKNOWLEDGMENT [COOKIE-ACK] (0x0B)
- EXPLICIT CONGESTION NOTIFICATION ECHO [ECNE] (0x0C)
- CONGESTION WINDOW REDUCE [CWR] (0x0D)
- SHUTDOWN-COMPLETE (0x0E)

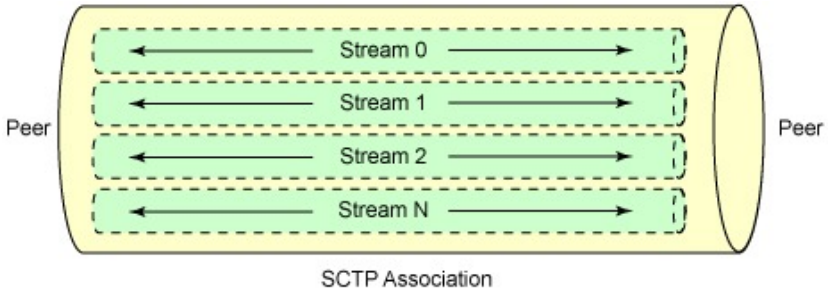

Figura 3: MultiStreaming en SCTP [3]

5. MultiStreaming

Una conexión realizada con el protocolo SCTP, es posible dividirla en varios streams, posibilitando la recepción en forma paralela de diferente información, que podría tratarse por ejemplo de un sitio web con su página html y los anexos como fotos, pudiendo presentar de forma más rápida el contenido de este al usuario.

6. MultiHoming

Una conexión multi-homing se refiere a que es posible la conexión simultánea desde y hacia diferentes enlaces físicos de conexión, posibilitando una mayor velocidad de transmisión y/o una seguridad de enlaces, esto es como respaldo en caso de una pérdida de alguno de ellos.

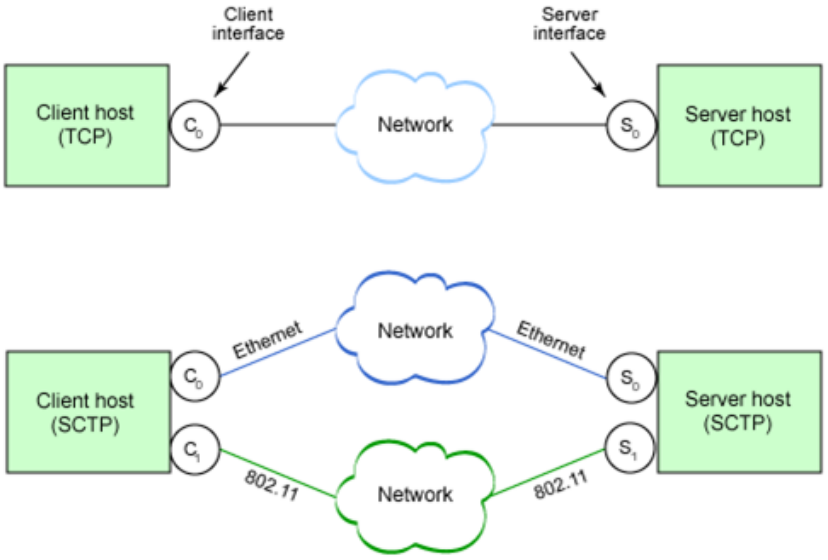

Figura 4: MultiHoming en SCTP v/s TCP [4]

[2] http://files.myopera.com/blu3c4t/blog/pdu.jpeg
[3] http://www.ibm.com/developerworks/library/l-sctp/figure3.gif
[4] http://profesores.elo.utfsm.cl/~agv/elo323/2s10/projects/CaponaAhumada/pics/multihoming.png

7. SCTP en JAVA

El paquete çom.sun.nio.sctp"de Opensdk [5] define las clases/interfases que son usadas para el menejo de del protocolo SCTP.
Las principales clases que maneja son 3 tipos de channels que se pueden dividir en 2 grupos.

1. El primer grupo es *SctpChannel* y *SctpServerChannel*. Un *SctpChannel* puede controlar solo 1 asociación simple, esto es enviar y recibir datos a un endpoints simple. *SctpServerChannel* escucha y acepta nuevas asociaciones inicializadas desde un socket address.

2. El segundo grupo con consiste solo de $SctpMultiChannel$. Instancias de este tipo de channels puede controlar múltiples asociaciones, por lo tanto puede enviar y recibir datos de y hacia muchos Endpoints diferentes.

El SCTP stack es controlado por eventos , y las aplicaciones pueden recibir notificaciones de ciertos eventos de SCTP. Estos eventos son más útiles para $SctpMultiChannel$ ya que puede controlar múltiple asociaciones, es necesario realizar un seguimiento de la situación de estas notificaciones. Por ejemplo .ªssociationChangeNotification"le permite saber cuando se inicia o se terminan nuevas asociaciones. Si la asociacion permite la configuración dinámica de direcciones entonces $PeerAddressChangeNotification$ le permite saber acerca de las direcciones IP que se han añadido o eliminado desde un peer endpoint. $MessageInfo$ proporciona datos auxiliares del mensaje ya sea enviado o recibido.

Además es necesario manejar las clases de Socket y y manejo de IP, $java.net.InetAddress$, $java.net.InetSocketAddress$ y $java.net.SocketAddress$.

7.1. Cliente

Las partes claves para un cliente se describen en el siguiente código donde se debe crear el socket con la IP del servidor y el puerto al cual se debe conectar, posteriormente se crea un objeto *SctpChannel* con el socket creado y además se crea el objeto de $MessageInfo$ que contiene información adicional del mensaje que se envía. Luego se usa el método *send* de *SctpChannel* con el mensaje y la información adicional.

```
InetSocketAddress socketAddress =
     new InetSocketAddress(IP_SERVER, SERVER_PORT);
...
SctpChannel sctpChannel = SctpChannel.open(socketAddress, 1 ,1)
MessageInfo messageInfo = MessageInfo.createOutgoing(null, 0);
...
sctpChannel.send(byteBuffer, messageInfo);
```

7.2. Servidor

Para el funcionamiento del servidor se tiene que crear un socket con el puerto por donde se escuchara (*SocketAddress*) y con el crear un objeto del tipo *SctpServerChannel* el cual tienen el método *accept*() que espera una nueva conexión, cuando esto ocurre devuelve un objeto del tipo *SctpChannel* el cual tiene implementado el método *recive*() para capturar los mensajes que lleguen al servidor.

```
SocketAddress serverSocketAddress =
    new InetSocketAddress(SERVER_PORT);
SctpServerChannel sctpServerChannel =
```

[5] http://openjdk.java.net/projects/sctp/javadoc/

```
    SctpServerChannel.open().bind(serverSocketAddress);
...
sctpServerChannel.accept();
...
sctpChannel.receive(buf , System.out, null);
```

8. Requisitos de sistema

En linux el módulo del SCTP no viene instalado por defecto en el kernel por lo que es necesario instalar los módulos extras del kernel

$$kernel - modules - extra$$

junto con el paquete específico de SCTP $lksctp - tools$.
En fedora la instalación sería

```
yum install kernel-modules-extra lksctp-tools
```

Para verificar la correcta instalación y funcionamiento de este, podemos revisar con los comandos *checksctp*, $modinfosctp$ y

```
sctp_test -H 127.0.0.1 -P 2222 -l
sctp_test -H 127.0.0.1 -P 2223 -h 127.0.0.1 -p 2222 -s
```

9. Referencias

http://openjdk.java.net/projects/sctp/javadoc/
http://thegeekhead.blogspot.com/2009/07/stream-control-transport-protocol-sctp.html
http://www.frlp.utn.edu.ar/materias/internetworking/apuntes/SCTP/SCTP.pdf
http://profesores.elo.utfsm.cl/~agv/elo330/2s10/projects/BenjaminGinouves/Documentacion.html
http://profesores.elo.utfsm.cl/~agv/elo330/2s11/projects/Reports/BarriosMedinaPerez/index.html