# Evaluación de Rendimiento del Protocolo SCTP para Aplicaciones de Robótica Móvil

Eduardo González Vidal, *Estudiante, Universidad Técnica Federico Santa María*

***Abstract*—En este trabajo se hace un estudio de las características y rendimiento del protocolo SCTP (Stream Control Transmission Protocol) para transmisión de datos con multi-homing y multi-streaming en aplicaciones de robótica móvil. Se analiza la mejora en robustez que ofrece el protocolo por sobre otros similares como TCP (Transmission Control Protocol) y la factibilidad de aplicar este protocolo en la transmisión confiable de información. Como caso de uso se propone una simulación de comunicación cliente-servidor entre un robot y un servidor remoto, cada uno con dos interfaces de red, en donde éste primero deba enviar datos de múltiples sensores en tiempo real. Los resultados muestran que el protocolo ofrece una mejora en robustez, a través de una mayor disponibilidad de servicio y menores retardos de envío por head-of-line blocking.**

***Index Terms*—SCTP, Robótica Móvil, Multi-homing, Multi-streaming.**

## I. Introducción

En el área de robótica, es común tener sistemas que requieran comunicar datos hacia un dispositivo lejano. Este dispositivo puede ser un computador, que haga funciones de servidor central para la recepción de la información, en cuyo caso el método de preferencia para el envío de datos es Internet. En estas arquitecturas cliente-servidor, el servidor se encuentra en un punto fijo (IP fija) y tiene acceso a Internet por algun medio confiable. Sin embargo, este no es en general el caso para un robot móvil que haga las veces de cliente, ya que en primer lugar debe utilizar medios inalámbricos para enviar sus datos, y en segundo lugar su movilidad da origen a conexiones poco estables. Para dar mayor robustez a la comunicación se puede considerar entregar al robot más de una interfaz de red para acceder a Internet, e.g, wifi y 3G, de modo de tener un respaldo en caso de que una se deteriore o no se encuentre disponible. Sin embargo, para que esto efectivamente mejore la disponibilidad de conexión, es necesario tener un protocolo de comunicación que tenga tolerancia a fallos (fallover), y de esta forma la redundancia implementada se traduzca en menor ”down time” entre el cliente y el servidor. Dado que es deseable (salvo quizá en comunicación de datos multimedia) tener un protocolo que asegure entrega confiable y en orden de paquetes, TCP (Transmission Control Protocol) aparece como una solución natural. Sin embargo, no entrega servicios que permitan sacar ventaja de poseer más de una interfaz de red.

Stream Control Transmission Protocol (SCTP) es un protocolo de capa de transporte que tiene tolerancia a fallos, ofrece entrega confiable y en orden de paquetes, y tiene otras características que lo hacen la opción ideal para comunicar información cuando maximizar la disponibilidad de conexión es un tema de importancia. Este es el caso en general para robótica móvil, donde los datos son generalmente mediciones de sensores, mensajes de alerta o comandos de teleoperación que no admiten largos periodos sin conexión.

Al tratarse de un protocolo relativamente nuevo, su difusión hoy no es masiva. Sin embargo, se han realizado aplicaciones y hecho estudios relativos a evaluar su desempeño en comparación con TCP y UDP. En [1], se analiza el ”handover” realizado por SCTP, focalizado en el control de flujo y verificando que las tasas de transmisión se reducen drásticamente inmediatamente después del mismo. Los autores proponen un método de control de flujo eficiente para manejar mejor estas transiciones, considerandolo apropiado para entrega de mejor calidad de servicio (QoS). Se han propuesto diferentes aplicaciones para el protocolo, como en [2], donde se mejoró los retardos de transmisión y mejoró el thoughput en una red eléctrica inteligente, en comparación con una implementación con TCP. Otra aplicación es el envío de datos usando FTP sobre SCTP, para evitar los overheads que aparecen al usar TCP. Esto puede mejorar los tiempos de transferencia significativamente, y en particular si se explotan las cualidades multi-streaming de SCTP, mejorando el rendimiento por factores de hasta 1.5 [3]. Por otro lado, otros autores señalan que SCTP muestra tasas de transmisión inferiores a TCP cuando la conexión es estable, señalando que la razón puede deberse a que no ha sido optimizado (como TCP) y que algunos mecanismos que se adecuan bien a TCP no necesariamente son óptimos para SCTP, tales como el sistema de control de congestión [4].

En este trabajo se analiza el rendimiento y las capacidades de SCTP pensando en aplicaciones de robótica móvil, haciendo especial énfasis en las características de multi-homing y multi-streaming del protocolo. Para ello se simula la transmisión de datos desde un robot que posee múltiples sensores, y que tiene disponibles dos interfaces de red, hacia un servidor central. La simulación se hace enviando datos por diferentes streams desde un computador a otro a través de Ethernet y wifi, y evaluando si la conexión se mantiene estable ante la pérdida de una de estas interfaces. La sección II da un breve resumen de las características más importantes de SCTP y sus diferencias con TCP y UDP; la sección III explica la simulación realizada y muestra sus resultados; y finalmente la sección IV presenta las conclusiones.

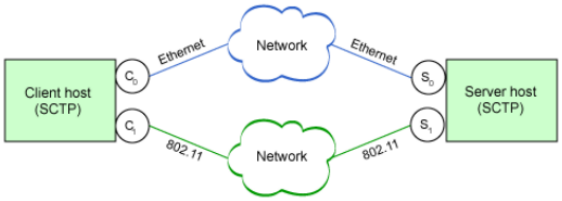


Fig. 1. Dos hosts multi-homed conectados a internet por dos interfaces de red diferentes. Fuente: www.ibm.com/developerworks/linux/library/l-sctp/

## II. Características de SCTP

El protocolo SCTP toma muchas de las características de TCP y UDP, agregando otras propias para adaptarse a las necesidades de los últimos tiempos y los avances en tecnología y redes. Algunas de estas son:

- Entrega confiable de paquetes (por stream)
- Multi-homing
- Multi-streaming
- Protección al iniciar comunicación (contra ataques SYN)
- Encuadramiento de mensajes
- Envío de datos en orden (se puede configurar para que no sea restricción)

Se analizará en más detalle dos de estas características por el rol que juegan en dar robustez a las comunicaciones, multi-homing y multi-streaming, y se verá el esquema básico para programar un socket SCTP.

### A. Multi-homing

La idea es ofrecer mayor disponibilidad de conexión a las aplicaciones. Un host "multi-homed" es aquel que tiene más de una interfaz de red, y por tanto más de una dirección IP a la cual se le puede dirigir tráfico de datos. Esta idea se refleja en la Fig. 1, donde ambos el cliente y el servidor se encuentran conectados a Internet por un medio cableado (Ethernet) e inalámbricamente (wifi), cada una con sus respectivas direcciones IP.

En SCTP ya no se usa el concepto de conexión, sino que se habla de *asocicación* entre dos máquinas. Este concepto es más amplio que una conexión, pues relaciona un conjunto de direcciones IP en una máquina, con otro conjunto de una o más direcciones IP en la otra (pero asociados a un mismo puerto). SCTP monitorea los caminos de la asociación a través de un mensaje de latido, *heartbeat*. Se puede programar la frecuencia con que se envía este mensaje, de manera que apenas se detecta que se ha caído un camino, el protocolo envía el tráfico de datos a través de la ruta alternativa. Este comportamiento es conocido como *fallover*, y es completamente transparente para las aplicaciones.

Para el caso de robótica móvil, será común que se tengan dos interfaces de red disponibles (wifi y 3G), cada una con sus respectivas IPs asociadas. Gracias a multi-homing, el robot puede usar wifi cuando éste se encuentre disponible y cambiarse a 3G si la potencia de la señal wifi baja de cierto umbral o se pierde, sin perder en ningun momento la transmisión confiable de datos.

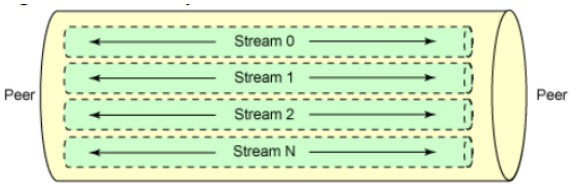


Fig. 2. Envío de datos a través de múltiples streams en una misma asociación. Fuente: www.ibm.com/developerworks/linux/library/l-sctp/

### B. Multi-streaming

El protocolo SCTP soporta el envío de paquetes a través de multiples *streams* dentro de una misma asociación. Los streams se deben ver como transferencias independientes de datos a través de una misma asociación. Cada stream se identifica con un número entero, y cada paquete enviado debe incluir en su encabezado a qué stream pertenece. Dado que cada stream se maneja en forma independiente, el protocolo sólo asegura que los paquetes lleguen a destino en orden relativo a su stream, y no a la totalidad de los paquetes enviados. De esta forma se supera parcialmente el problema "head-of-line-blocking" de TCP. Éste consiste en que, ante una pérdida, los demás paquetes enviados no son pasados a la aplicación destino, incluso si han llegado correctamente, hasta que el paquete perdido sea retransmitido y bien recibido. En SCTP la pérdida de un paquete sólo genera efectos a nivel de su stream, y no afecta a los demás flujos de datos. Evidentemente, enviar todos los datos por un único stream equivale al servicio ofrecido por TCP. La Fig. 2 ilustra el concepto de multi-streaming.

En el contexto de robótica móvil es común que la información enviada sea de múltiples tipos, e.g, sensores de distinta naturaleza. Una forma de distinguir el tipo de dato siendo recibido es etiquetar el paquete al momento de enviarlo indicando a qué sensor corresponde. Sin embargo, con multi-streaming es posible solucionar este problema en forma nativa a nivel de transporte. Por lo demás, se evita que una falla en un sensor afecte la transmisión confiable de los datos de otros sensores, lo cual es altamente deseable si se quiere tener un sistema escalable.

### C. Programación de un Socket SCTP

La programación de un socket SCTP es muy similar a la de un socket TCP. En la Fig. 3 se resume para una arquitectura cliente-servidor. Las funciones *sctp_sendmsg* y *sctp_recvmsg* permiten controlar los flujos de datos, así como los streams por los que se envían los paquetes. Una cualidad importante es que la elección de a qué ruta se escogerá para transmitir el paquete, es decir, a cuál de todas las direcciones IP disponibles se debe enviar los datos, es transparente para el programador. Existen dos formas de crear sockets SCTP, *uno-a-uno* o *uno-a-muchos*. Éste primero es para asociaciones de una máquina a otra, mientras que la segunda es para situaciones donde se quiere que un socket controle múltiples asociaciones, e.g, un servidor. De esta forma la aplicación no necesita crear y

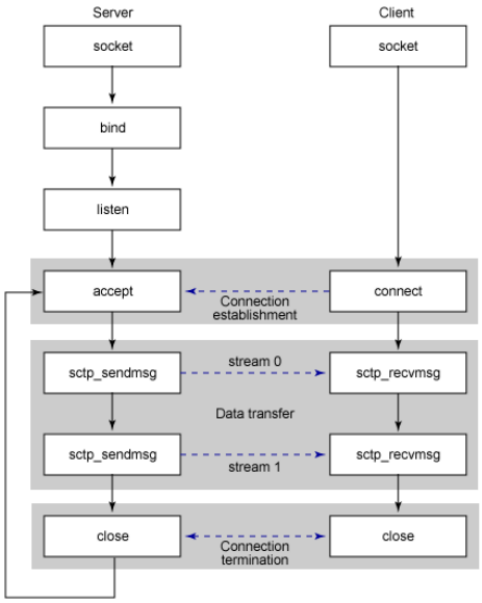


Fig. 3. Secuencia de programación de un socket SCTP. Fuente: www.ibm.com/developerworks/linux/library/l-sctp/


Fig. 4. Escenario de aplicación de SCTP en robótica móvil.

manejar un socket distinto para cada cliente que se conecte a él.

## III. Simulación

Un escenario real de aplicación se muestra en la Fig. 4. El recuadro enmarca la parte estrictamente asociada a redes, y que fue simulada en este trabajo. Para efectos de simulación se consideran dos computadores, uno haciendo las veces de robot (cliente) y otro actuando como servidor para la recepción y despliegue de los datos recibidos por pantalla. Para ello se programa un socket SCTP *uno-a-uno* que envíe dos tipos de datos al servidor, unos simulando mediciones de temperatura, y otro simulando datos de ubicación (GPS), cada uno por un stream independiente.

Cada uno de los dos computadores se conecta a Internet por Wifi y Ethernet, y en el momento de la asociación inicial ambos se comparten sus respectivas direcciones IP. Esto ocurre a nivel de capa transporte, y no requiere de ser programada. Una vez comenzada la comunicación el mensaje de latido se envía a una frecuencia fija para verificar el estado de las conexiones. Las pruebas realizadas fueron escencialmente dos:

- Verificar si la conexión se mantiene al desconectar la interfaz de red siendo utilizada para el envío/recepción de datos.

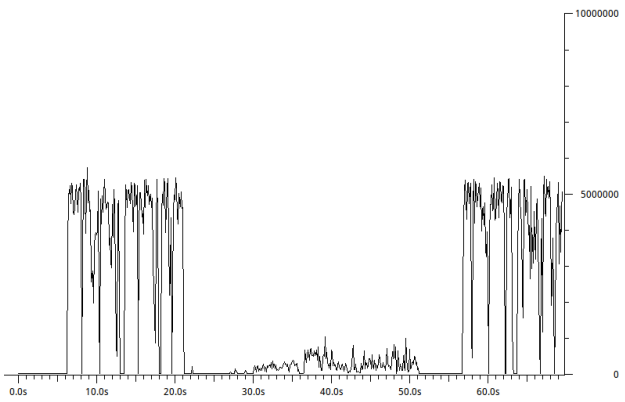


Fig. 5. Throughput en el tiempo en el servidor [Bytes/seg].

- Comprobar si se evita *head-of-line-blocking*. Para ello se envían en forma intercalada los dos tipos de mensajes por dos streams distintos. Si no hay pérdidas, el servidor recibe en forma intercalada ambos tipos de datos. Si llegan a aparecer dos datos del mismo tipo en forma consecutiva, significa que se ha perdido un paquete pero no ha retrasado el envío de información por el otro flujo.

### A. Resultados

Los experimentos indicados se realizaron para el cliente y para el servidor, en ambos casos obteniéndose los mismos resultados. Sin embargo, en esta sección se presenta el caso en que el servidor debe realizar el fallover, a pesar de que el caso más común en robótica es que el cliente sea quien tenga conexión menos estable. Se hizo así ya que el cliente es quien se comunica con el servidor inicialmente. Dado que el cliente envía datos hacia el servidor, es una situación más exigente que el destino al cual envía sus datos de pronto ya no esté disponible, en contraposición a que la dirección destino de sus paquetes sea siempre fija y tenga que simplemente reconocer que su enlace de salida se ha caído y deba rutear por el alternativo. De esta forma requiere que el cliente esté permanentemente atento a posibles direcciones destino alternativas para hacer llegar sus paquetes, o de lo contrario la comunicación se caerá.

Usando wireshark se monitorean los paquetes que llegan a la máquina receptora (server). Los paquetes se dirigen inicialmente desde el cliente a la dirección Ethernet del servidor. Pasado un intervalo de tiempo, se desconecta el cable Ethernet de la máquina servidor, obligando al cliente a redirigir sus paquetes hacia la IP asociada a wifi por el lado del servidor. En la Fig. 5 y 6 se grafican el throughput y los paquetes recibidos por el cliente en el tiempo.

Se puede ver claramente que a 21 segundos se desconecta el cable Ethernet, y la conexión se recupera unos 5 segundos más tarde. Luego a los 51 segundos se vuelve a conectar el cable Ethernet y el cliente nuevamente retoma ésta conexión (primaria) por sobre la alternativa (wifi). La diferencia de throughput evidencia que el primer tramo

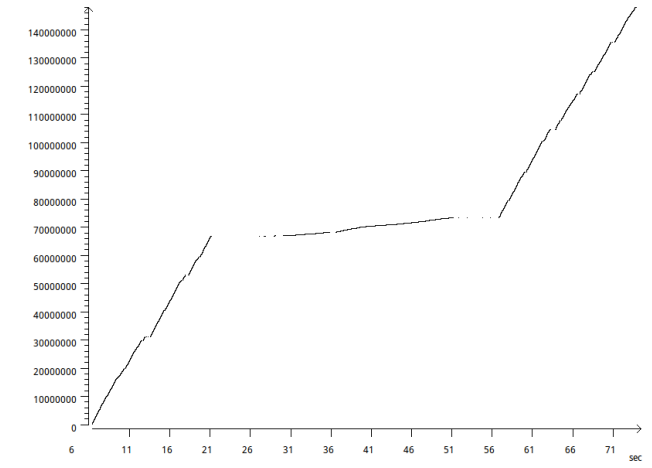


Fig. 6. Bytes recibidos en el tiempo por servidor.

```
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 0 | SSN: 23238 | Data="GPS data                #23238"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 1 | SSN: 23238 | Data="Temperature data      #23238"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 0 | SSN: 23239 | Data="GPS data                #23239"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 1 | SSN: 23239 | Data="Temperature data      #23239"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 0 | SSN: 23240 | Data="GPS data                #23240"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 1 | SSN: 23240 | Data="Temperature data      #23240"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 0 | SSN: 23241 | Data="GPS data                #23241"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 1 | SSN: 23241 | Data="Temperature data      #23241"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 0 | SSN: 23242 | Data="GPS data                #23242"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 1 | SSN: 23242 | Data="Temperature data      #23242"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 0 | SSN: 23243 | Data="GPS data                #23243"
Size: 1000 B | From: 192.168.1.197:50000 | Stream: 1 | SSN: 23243 | Data="Temperature data      #23243"
```

Fig. 7. Recepción de paquetes en forma confiable y en orden por el lado del servidor. Cada tipo de dato llega por un stream distinto.

```
1000 bytes received from 192.168.2.11:50000, stream: 1, ssn: 1221              Data=" Temperature data      #1221"
1000 bytes received from 192.168.2.11:50000, stream: 0, ssn: 1222              Data=" GPS data              #1222"
1000 bytes received from 192.168.2.11:50000, stream: 1, ssn: 1222              Data=" Temperature data      #1222"
1000 bytes received from 192.168.2.11:50000, stream: 0, ssn: 1223              Data=" GPS data              #1223"
1000 bytes received from 192.168.2.11:50000, stream: 1, ssn: 1223              Data=" Temperature data      #1223"
1000 bytes received from 192.168.2.11:50000, stream: 0, ssn: 1224              Data=" GPS data              #1224"
1000 bytes received from 192.168.2.11:50000, stream: 1, ssn: 1224              Data=" Temperature data      #1224"
1000 bytes received from 192.168.2.11:50000, stream: 0, ssn: 1225              Data=" GPS data              #1225"
1000 bytes received from 192.168.2.11:50000, stream: 0, ssn: 1226              Data=" GPS data              #1226"
1000 bytes received from 192.168.2.11:50000, stream: 0, ssn: 1227              Data=" GPS data              #1227"
1000 bytes received from 192.168.2.11:50000, stream: 0, ssn: 1228              Data=" GPS data              #1228"
```

Fig. 8. Eliminación de head-of-line-blocking usando multi-streaming.

en efecto corresponde a Ethernet, mientras que el tramo central es significativamente más lento por tratarse de wifi. Durante el transcurso del experimento, en ningun momento se perdió la conexión entre ambos computadores, y todos los mensajes arriban en orden, como se puede ver en la Fig. 7.

Finalmente, en la Fig. 8 se puede ver el comportamiento del protocolo ante una pérdida de paquete. Ante la perdida de un paquete de temperatura, se continua enviando los paquetes de GPS, por lo que no se interrumpe el flujo de datos de los demás sensores.

## IV. Conclusión

El protocolo SCTP ofrece servicios útiles para dar mayor robustez y disponibilidad de conexión a sistemas que posean más de una intefaz de red disponible para conectarse a Internet. Además, permite particionar el envío de datos en flujos independintes que efectivamente eliminan el problema ”head-of-line-blocking” presente en TCP. Sus cualidades lo hacen una muy buena alternativa en aplicaciones como envío de datos de sensores en robótica móvil, pudiendo adaptarse a las condiciones del ambiente – señal y medios de acceso a Internet disponibles – en forma dinámica y transparente para las aplicaciones que corran en la máquina.

Este protocolo no fue evaluado en términos de throughput respecto a TCP, dado que el flujo de datos de sensores no es muy grande, y por tanto no es de gran relevancia tener un posible overhead mayor a TCP. Sin embargo, sería interesante comprobar los resultados expuestos por otros autores que aseguran SCTP es menos eficiente en este sentido.
Finalmente mencionar que el protocolo no permite el envío de datos por dos interfaces en forma simultánea. Protocolos aún mas recientes, están en desarrollo en esta línea, tales como Multipath TCP.
Queda como trabajo futuro la implementación de un sistema real con SCTP y usando meramente medios inalámbricos, así como una evaluación del rendimiento en términos de throughput en comparación con TCP.

## References

[1] Keun Jae Lee, Sang Su Nam and Byung In Mun, ”SCTP efficient flow control during handover”, *Wireless Communications and Networking Conference*, WCNC, 2006.
[2] Majed Alowaidi, F. Richard Yu, Abdulmotaleb El Saddik and Abdullah Aljanah, ”Improving Performance of Smart Grid Communications using Multi-homing and Multi-streaming offered by SCTP”, *IEEE SmartGridComm 2012 Symposium*, Tainan, 2012.
[3] Sourabh Ladha, Paul D. Amer, ”Improving File Transfers Using SCTP Multistreaming”, *IEEE International Conference on Performance, Computing and Communications*, 2006.
[4] Tan Wen-Yuan, Xu De-Wu, Chen Wei, ”Research on the Transport Performance of SCTP”, *International Symposium on Computer Network and Multimedia Technology*, CNMT, 2009.