The Impact of SCTP on SIP Server Scalability and Performance

Kumiko Ono and Henning Schulzrinne
{kumiko, hgs}@cs.columbia.edu
Outline

- What is SCTP?
- How does choosing SCTP as a transport protocol for SIP impact the SIP server?
  - Scalability
    - Number of sustainable SCTP associations compared with the number of TCP connections
  - Performance
    - Setup and transaction response times compared with TCP
What is SCTP?

- Stream Control Transmission Protocol
  - RFC 4960 in 2007 (RFC 2960 in 2000)
  - Originally designed for carrying SS7 (Signaling Systems No.7) over IP
    - Reliability
    - Congestion control
    - Multi-streaming
    - Multi-homing
- An alternative transport protocol for SIP
  - TCP, UDP or SCTP (RFC 4168 in 2005)
## Comparison of transport protocols

<table>
<thead>
<tr>
<th></th>
<th>UDP</th>
<th>TCP</th>
<th>SCTP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection-oriented</strong></td>
<td>No</td>
<td>Yes: create with a three-way handshake, and terminate with half-close. → SYN flooding attacks</td>
<td>Yes: create with a four-way handshake. No half-close in termination. → Resist SYN/INIT flooding attacks using cookies</td>
</tr>
<tr>
<td><strong>large message &gt;MTU</strong></td>
<td>No: lean on IP fragmentation</td>
<td>Yes: segmentation</td>
<td>Yes: segmentation</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td>No: lean on App. features</td>
<td>Yes: support ack., T.O., and re-transmission</td>
<td>Yes: support ack., T.O., and re-transmission</td>
</tr>
<tr>
<td><strong>Congestion control</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Flow control</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Message-oriented</strong></td>
<td>Yes: preserve the boundary</td>
<td>No: byte-stream</td>
<td>Yes: preserve the boundary</td>
</tr>
<tr>
<td><strong>Multi-homing</strong></td>
<td>No</td>
<td>No</td>
<td>Yes: failover tolerant</td>
</tr>
<tr>
<td><strong>Multi-streaming</strong></td>
<td>No</td>
<td>No</td>
<td>Minimize head-of-line blocking</td>
</tr>
</tbody>
</table>
Outline

- What is SCTP?
- How does choosing SCTP as a transport protocol for SIP impact the SIP server?
  - Scalability
    - Number of sustainable SCTP associations compared with the number of TCP connections
  - Performance
    - Setup and transaction response times compared with TCP
How does choosing SCTP impact SIP servers?

**SCTP features**
- Resist SYN flooding attacks
  - by cookies in the four-way handshake

**Expected impacts**
- More RTTs cause longer setup time, but the **piggyback setup option** could mitigate it.
TCP vs. SCTP: Handshake to initiate a connection/association

TCP three-way handshake
- Client: connect() (blocks)
- Server: SYN
- Client: SYN, ACK
- Server: SYN-RCVD
- Client: ACK
- Server: ESTABLISHED

SCTP four-way handshake
- Client: connect() (blocks)
- Server: INIT
- Client: INIT-ACK (cookie)
- Server: COOKIE-ECHO [data]
- Client: COOKIE-ACK
- Server: ESTABLISHED

Connection status:
- LISTEN
- CONNECTED
- ESTABLISHED

Association status:
- CLOSED
- ESTABLISHED

vulnerable to SYN flooding attack
How does choosing SCTP impact SIP servers?

SCTP features

- Resist SYN flooding attacks
  - by cookies in the four-way handshake
- Minimizing HOL blocking
  - by multi-streaming
- Failover tolerance
  - by multi-homing
- Easier parsing
  - by preserving message boundaries

Expected impacts

😊 More RTTs cause longer setup time, but the piggyback setup option could mitigate that.

😢 More data structures make it less scalable, but one-to-many style sockets could mitigate that.

😊 Shorter transaction time
Outline

- What is SCTP?
- How does choosing SCTP as a transport protocol for SIP impact the SIP server?
  - Scalability
    - Number of sustainable SCTP associations compared with the number of TCP connections
  - Performance
    - Setup and transaction response times compared with TCP
SCTP measurement: Scalability

- **Background:**
  - available as a kernel module in Linux
    - can use between a server and clients

- **Goals:**
  - to establish an upper limit of concurrent associations
    - SCTP one-to-one socket: TCP-like
    - SCTP one-to-many socket: UDP-like
  - to clarify the effect of SCTP one-to-many sockets
Measurement environment

- **Server:** an echo server
  - CPU: Pentium IV, 3GHz (dual core) 32-bit
  - RAM: 4GB
  - OS: Linux 2.6.23 (default VM split, 1G/3G)

- **Clients:**
  - CPU: Pentium IV, 3GHz 32-bit
  - RAM: 1GB
  - OS: Redhat Linux 2.6.9

- **SCTP**
  - enable a kernel module for SCTP
  - enable SCTP object count
Echo server measurement:
Number of sustainable assoc. for SCTP

- one-to-one socket
  - TCP-like
    - 1 socket : 1 assoc.

- one-to-many socket
  - UDP-like
    - 1 socket : N assoc.

Overviews of the data structures for 3 SCTP associations
Echo server measurement:
Number of sustainable assoc. for SCTP

- **one-to-one socket**
  - TCP-like
    - 1 socket : 1 assoc.
  - Upper limit
    - 74,000 assoc.
    - 11.1 KB/assoc
    - Ends by out-of-memory
    - [Ref] TCP connections: 419,000

- **one-to-many socket**
  - UDP-like
    - 1 socket : N assoc.
  - Upper limit
    - 90,000 assoc.
    - 8.9 KB/assoc
    - Ends by out-of-memory

Overviews of the data structures for 3 SCTP associations
Detailed data structures: SCTP vs. TCP

- The `sctp_association` data structure
  - dominates the memory usage of a SCTP socket: 5,120 bytes, but allocated at size-8192 slab object
  - The dominant sub member is `tsn_map` to trace received TSNs for unordered data delivery.

---

Protocol [SCTP_TSN_MAP_SIZE] *allocated from general purpose slab

12/2/2008
Detailed data structures: SCTP vs. TCP

- The sctp_association data structure
  - dominates the memory usage of a SCTP socket: 5,120 bytes, but allocated at size-8192 slab object
  - The dominant sub member is tsn_map to trace received TSNs for unordered data delivery.

![Memory usage graph](image)

Protocol [SCTP_TSN_MAP_SIZE] *allocated from general purpose slab
How does choosing SCTP impact SIP servers?: Results

SCTP features
- Resist SYN flooding attack
  - by cookies in the four-way handshake
- Minimizing HOL blocking
  - by multi-streaming
- Failover tolerance
  - by multi-homing
- Easier parsing
  - by preserving message boundaries

Expected impacts
- More RTTs cause longer setup time, but the piggyback setup option could mitigate it.
- More data structures make it less scalable, but one-to-many style sockets could mitigate it.

Number of sustainable associations
- One-to-one sockets: 74,000 (17%)
- One-to-many sockets: 90,000 (21%)
- [Ref] TCP connections: 419,000
- Improvable to up to 50% of TCP by adjusting the size of the tsnP_map
Outline

- What is SCTP?
- How does choosing SCTP as a transport protocol for SIP impact the SIP server?
  - Scalability
    - Number of sustainable SCTP associations compared with the number of TCP connections
  - Performance
    - Setup and transaction response times compared with TCP
SCTP measurement: Performance

Goals:
- to clarify the effect of SCTP piggyback setup option
  - using an echo server
  - measuring the setup and transaction times
- to clarify the effect of message-orientation
  - using a SIP front-end server, which focusing on message parsing
One-to-one vs. one-to-many for client

**[Seq. 1]**

Client: one-to-one  
Server: one-to-one

- `connect()` (blocks)
- `setup time`
- `connect()` returns
- `sendmsg()`

**[Seq. 2]**

Client: one-to-many  
Server: one-to-one

- `sendmsg()`
- `accept()` (blocks)
- `connect()` (blocks)
- `accept()` returns
- `recvmsg()`
- `sendmsg()`
- `accept()` returns
- `recvmsg()`

**Transaction time**

- `recvmsg()`
- `12/2/2008`
- `SACK`
Results of setup and transaction time using echo server

- SCTP one-to-one vs. TCP
  - Longer setup delay for SCTP by 0.23 ms than TCP
  - Piggyback setup of SCTP can slightly mitigate the delay in our environment.
    - Expensive cookie handling
  - Similar transaction time

<table>
<thead>
<tr>
<th>Socket style at server</th>
<th>Setup type</th>
<th>Setup (ms)</th>
<th>Transaction (ms)</th>
<th>Total (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCTP one-to-one</td>
<td>regular</td>
<td>0.34</td>
<td>0.54</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>piggyback</td>
<td>0.84</td>
<td></td>
<td>0.84</td>
</tr>
<tr>
<td>TCP</td>
<td></td>
<td>0.17</td>
<td>0.48</td>
<td>0.65</td>
</tr>
</tbody>
</table>
One-to-one vs. one-to-many for server

[Seq. 1]  
Client one-to-one  
Server one-to-one  
connect() (blocks)  
INIT  
INIT-ACK (cookie)  
COOKIE-ECHO  
COOKIE-ACK  
call connect()  
returns  
sendmsg()  
setup time  
transaction time  
recvmsg()  
12/2/2008

[Seq. 3]  
Client one-to-one  
Server one-to-many  
connect() (blocks)  
INIT  
INIT-ACK (cookie)  
COOKIE-ECHO  
COOKIE-ACK  
accept() (blocks)  
accept() returns  
connect()  
returns  
sendmsg()  
recvmsg()  
Create an assoc., but not a socket  

DATA  
SACK  
recvmsg()  
sendmsg()  
recvmsg()  
sendmsg()  
sendmsg()  
recvmsg()
Results of setup and transaction time using echo server

- SCTP one-to-many sockets
  - Setup and transaction times do not remain constant, but linearly increase with the number of maintaining associations.

<table>
<thead>
<tr>
<th>Socket style at server</th>
<th>Setup type</th>
<th>Setup (ms)</th>
<th>Transaction (ms)</th>
<th>Total (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCTP one-to-one</td>
<td>regular</td>
<td>0.34</td>
<td>0.54</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>piggyback</td>
<td>0.84</td>
<td></td>
<td>0.84</td>
</tr>
<tr>
<td>SCTP one-to-many</td>
<td></td>
<td>0.38-170.91</td>
<td>0.65 – 34.14</td>
<td>1.03-205.05</td>
</tr>
<tr>
<td>TCP</td>
<td></td>
<td>0.17</td>
<td>0.48</td>
<td>0.65</td>
</tr>
</tbody>
</table>
Results of echo server measurement: Setup and transaction times

- **SCTP one-to-many sockets**
  - Setup and transaction times do not remain constant, but linearly increase with the number of maintaining associations.
  - Caused by linear search for an association corresponding to an endpoint.
  - Improvable by using a hash table lookup

<table>
<thead>
<tr>
<th>Socket style at server</th>
<th>Setup type</th>
<th>Setup (ms)</th>
<th>Transaction (ms)</th>
<th>Total (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCTP one-to-one</td>
<td>regular</td>
<td>0.34</td>
<td>0.54</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>piggyback</td>
<td>0.84</td>
<td></td>
<td>0.84</td>
</tr>
<tr>
<td>SCTP one-to-many</td>
<td></td>
<td>0.34</td>
<td>0.53</td>
<td>0.87</td>
</tr>
<tr>
<td>TCP</td>
<td></td>
<td>0.17</td>
<td>0.48</td>
<td>0.65</td>
</tr>
</tbody>
</table>
How does choosing SCTP impact SIP servers?: Results

SCTP features
- Resist SYN flooding attack
  - by cookies in the four-way handshake
- Minimizing HOL blocking
  - by multi-streaming
- Failover tolerance
  - by multi-homing
- Easier parsing
  - by preserving message boundaries

Expected impacts
- More RTTs cause longer setup time, but **the piggyback setup option** could mitigate it.
  - The effect of piggyback setup is slight.
  - Smaller RTT is effective, but depends strongly on network conditions.
  - Expensive cookie handling
- Maintaining associations has no significant impact for SCTP after replacing a linear search with a hash table lookup.
Conclusion

- Using SCTP impacts scalability rather than performance.
  - Recommend to use the one-to-many sockets
    - Number of sustainable connections
      - 17-21% of TCP in the default configuration
      - up to 50% by adjusting the tsn_map size
  - Recommend to use the piggyback setup
    - Setup delay is longer than TCP by 0.17 ms, but would be useful in a wide area network.
- Need to mature implementation on Linux