SCTP: An Overview

Randall Stewart, Cisco Systems
Phill Conrad, University of Delaware
Our Objectives

• Be able to explain
  what SCTP is, and what its major features are
  when and why you might use it (instead of TCP or UDP)
  where to find it, and find out more about it

• Be able to write code to use SCTP

  Sockets API

But first, lets find out:
  who are you, and what are your objectives?
Prerequisites

- Basic understanding of IP, and transport protocols TCP and UDP socket programming in C under Unix
- Willingness to put up with an engineer attempting to teach a tutorial :-D

Also, please note:

- *Please* interrupt to ask questions if you get lost.
- We will cover a lot of ground in a limited time so hold on to your seats :-D
These slides will be online at:

• [http://pel.cis.udel.edu/tutorial](http://pel.cis.udel.edu/tutorial)
  Faster download here

• [http://www.sctp.org](http://www.sctp.org)
  Also reachable with HTTP over SCTP!
  Downloads may be slow for while... be patient
  (hosted in Randy's barn while the Stewart family redecorates).
## Outline

<table>
<thead>
<tr>
<th>Time</th>
<th>Topic</th>
<th>Presenter(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10h00-11h00</td>
<td>intro (almost finished)</td>
<td>Randy</td>
</tr>
<tr>
<td></td>
<td>overview of SCTP</td>
<td>Phill</td>
</tr>
<tr>
<td></td>
<td>What is SCTP? What are the major features?</td>
<td></td>
</tr>
<tr>
<td>11h15-12h15</td>
<td>SCTP details</td>
<td>Randy</td>
</tr>
<tr>
<td>13h15-14h15</td>
<td>details of sockets API</td>
<td>Randy or Phill</td>
</tr>
<tr>
<td></td>
<td>open Q and A</td>
<td>Both</td>
</tr>
</tbody>
</table>
What is SCTP? Why SCTP?

- SCTP is a new IETF standard transport protocol (RFC2960)
  
  Stream Control Transmission Protocol

- An alternative to TCP and UDP

- It came out of the "signaling transport" community...
  
  doing telephone switching over IP networks

- .. but it emerged as a general-purpose transport protocol

- Why?
  
  because TCP and UDP lacked some features that were needed

- What was so special about sigtran?
  
  small message sizes
  
  need for high availability, absolute minimum delay

- Why talk about SCTP in this form?
  
  Because SCTP is coming soon to a Linux kernel near you (LK-SCTP)
What was special about sigtran?

- Aspects of signaling transport driving SCTP design
  - need for high availability
    - failover between multiple redundant network interfaces
  - need to monitor reachability status
  - message oriented
  - small message sizes
  - real-time (need absolute minimum delay)
  - upper layer timers
  - need for tunability (Big-I internet vs. engineered networks)
Why SCTP? The big picture...
The transport layer...

Application

Socket API

UDP | TCP

IP

Wifi | Eth

Transport

user-level
kernel

Application

Socket API

UDP | TCP

IP

Wifi | Eth

Transport
Why SCTP? The big picture…

TCP and UDP…

• The transport layer sits between IP and the application

• Traditionally, just two choices: TCP and UDP

• UDP: bare minimum
  just port numbers, and an optional checksum
  no flow control, no congestion control, no reliability or ordering

• TCP: a package deal
  flow control, congestion control, *byte-stream orientation*
  *total* ordering and *total* reliability

However, there are some things you can't get with either!

*(see next slide...)*
With SCTP you can do...

- Almost everything you can do with TCP and UDP
  (a very few minor exceptions we will note later but for instance:
   Can do reliable, flow controlled, congestion controlled data exchange, like TCP
   Can also do unordered, unreliable data exchange, like UDP)

- Plus the following features NOT available in UDP or TCP.
  (A quick list only; details follow!)
  - Multi-homing
  - Multi-streaming
  - Message boundaries (with reliability*)
  - Improved SYN-flood protection
  - Tunable parameters (Timeout, Retrans, etc.)
  - A range of reliability and order (full to partial to none)
    along with congestion control
  - and more...

*UDP: msg boundaries, not reliable
TCP reliable, no msg boundaries
Slightly more detail about key SCTP features... (1 of 3)

• Multi-homing *improved robustness to failures*
  
  In TCP, connections made between <IP addr, port> and <IP addr, port>
  
  If a host is multi-homed, you have to choose ONE IP Addr only, at each end
  
  If that interface goes down, so does the connection
  
  With SCTP, you can list as many IP addresses per endpoint as you like
  
  If host is still reachable through ANY of those addresses, connection stays up!

• Multi-streaming *reduced delay*
  
  A.k.a. partial ordering. Eliminates Head of Line (HOL) blocking
  
  In TCP, all data must be sent in order;
  loss at head of line delays delivery of subsequent data
  
  In SCTP, you can send over up to 64K independent streams, each ordered independently
  
  A loss on one stream does not delay the delivery on other streams
  i.e. multi-streaming eliminates HOL blocking
Slightly more detail about key SCTP features... (2 of 3)

- Message boundaries preserved *easier coding*
  
  TCP repacketizes data any old way it sees fit
  (message boundaries not preserved)

  SCTP preserves message boundaries

  Application protocols easier to write, and application code simpler.

- Improved SYN-flood protection *more secure*

  TCP vulnerable to SYN flood;
  (techniques to combat are "bags on the side")

  Protection against SYN floods is built-in with SCTP
  (Four way handshake (4WHS) vs 3WHS in TCP)

  Listening sockets don't set up state until a connection is validated
Slightly more detail about key SCTP features... (3 of 3)

- Tunable parameters (Timeout, Retrans, etc.) *more flexibility*
  - Tuning TCP parameters requires system admin privs, kernel changes, kernel hacking
  - SCTP parameters can be tuned on a socket by socket basis

- Congestion controlled unreliable/unordered data *more flexibility*
  - TCP has congestion control, but can't do unreliable/unordered delivery
  - UDP can do unreliable/unordered delivery, but not congestion controlled
  - SCTP is always congestion controlled, and offers a range of services from full reliability to none, and full ordering to none.
  - With SCTP, reliable and unreliable data can be multiplexed over same connection.
Features of SCTP (review)

Reliable data transfer w/SACK
- Congestion control and avoidance
- PMTU discovery and message fragmentation

- Message boundary preservation (with bundling)
- Multi-homing support

- Multi-stream support
- Unordered data delivery option
- Security cookie against connection flood attack (SYN flood)
- Built-in heartbeat (reachability check)
- Extensibility
What you can't do with SCTP

- byte-stream oriented communication
  SCTP inserts message boundaries between each "write()"
  write() a length field, then a data field ➔ result is 2 messages
  not a major issue, but need to be aware of when coding apps

- avoid congestion control
  UDP lets you blast away, but SCTP won't let you
  (you shouldn't be doing that anyway)

- true on-the-wire connectionless communication
  sockets API will let you send without doing a connection setup first (as with UDP), but connection setup still occurs on the wire
  connection setup required for congestion control (see above)
Now a VERY brief example: daytime client/server
(full socket API discussion comes later)

Network applications are typically client/server

daytime server

  open a socket and bind it to a port
  listen for connections
  while (1)
  {
    accept a connection
    send a string containing current date/time
    close the connection
  }

daytime client

  create a socket
  open a connection to daytime server
  read bytes until EOF (meaning connection was closed)
  close connection
A TCP daytime client becomes an SCTP daytime client...

TCP daytime client (many details omitted, including error checking; see Stevens et al. 2004 for details (and don't omit the error checking!))

```c
int sockfd, n;
char recvline[MAXLINE + 1]; /* read buffer*/
struct sockaddr_in servaddr;

sockfd = socket(AF_INET, SOCK_STREAM, 0); /* create TCP socket */
sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP); /* SCTP socket */
/* fill in socket address structure */
servaddr.sin_family = AF_INET; servaddr.sin_port = htons(13);
inet_pton(AF_INET, argv[1], &servaddr.sin_addr); /*dot dec to n.b.o.*/
connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr));
while ( (n = read(sockfd, recvline, MAXLINE)) > 0 )
{
    recvline[n]=0; /* null terminate */
    fputs(recvline, stdout);
}

close(sockfd);
```

Note: 0 implies IPPROTO_TCP
But... what is different?

(among other things) changing IP_PROTO_TCP to IP_PROTO_SCTP means...

• There has to be a listening SCTP process on the other side (port number spaces independent, just like UDP vs TCP)

• Bits on wire flow in IP datagrams with protocol number 132 (vs. 6 (TCP)or 17 (UDP)

• Four-way handshake instead of three-way handshake

• By default, both sides will exchange the IP addresses of ALL available network interfaces (both IPv4 AND IPv6!) and will use alternate IP addresses for retransmission in case of errors.

• Each read() on client side will correspond to exactly one write() on the server side.

  With TCP, sizes of write() and read() system calls on the two sides are independent

  With SCTP, each write() by the sending side produces a message with a specific length. That message becomes a "data chunk" inside the packet on the wire, and is delivered as a single message

  If the message is too large to be put in a single packet, it will be fragmented and reassembled. Likewise, small messages may be bundled in a single pkt.

  If it is too large to be delivered all at once, the "partial delivery" API will be invoked (details later).

  But, these last two items are exceptions, not the rule.
OK, so what?

• What you should know at this point is...
  SCTP is a new transport protocol
  It's available now in bleeding edge Linux and BSD kernels, and will make its way into the mainstream
  It has some cool new features (reviewed on next slide)
  If you know how to do socket programming with TCP, you can just change one field, and start using SCTP quickly

• But...
  to really take advantage of the cool new features, you need to know a bit more about them, and about the socket API for SCTP.

• So, that's what's next...
Stuff we'll talk about how to do...

The stuff you need advanced socket API features to take advantage of...

- implicit connection establishment
- using multiple interfaces (multi-homing)
- parallel streams
- unordered data
- mixing reliable and best effort traffic
- timed reliability
Outline

<table>
<thead>
<tr>
<th>Time</th>
<th>Topic</th>
<th>Presenter</th>
</tr>
</thead>
<tbody>
<tr>
<td>10h00-11h00</td>
<td>intro (almost finished)</td>
<td>Randy</td>
</tr>
<tr>
<td></td>
<td>overview of SCTP</td>
<td>Phill</td>
</tr>
<tr>
<td></td>
<td>What is SCTP? What are the major features?</td>
<td></td>
</tr>
<tr>
<td>11h15-12h15</td>
<td>SCTP details</td>
<td>Randy</td>
</tr>
<tr>
<td>13h15-14h15</td>
<td>details of sockets API</td>
<td>Randy or Phill</td>
</tr>
<tr>
<td></td>
<td>open Q and A</td>
<td>Both</td>
</tr>
</tbody>
</table>