IK1350 Protocols in Computer Networks/ Protokoll i datornätverk Spring 2008, Period 3

Module 7: SCTP

Lecture notes of G. Q. Maguire Jr.

KTH Information and Communication Technology

For use in conjunction with *TCP/IP Protocol Suite*, by Behrouz A. Forouzan, 3rd Edition, McGraw-Hill, 2006.

For this lecture: Chapter 13

Last modified: 2008.02.03:15:18

Transport layer protocols

- User Datagram Protocol (UDP)
  - Connectionless unreliable service
- Transmission Control Protocol (TCP)
  - Connection-oriented reliable stream service
- Stream Control Transmission Protocol (SCTP) <<< today's topic
  - Reliable message oriented service - a modern transmission protocol

Stream Control Transmission Protocol (SCTP) [33]

Provides a reliable message-oriented service; combining best of TCP & UDP.

- SCTP utilizes full-duplex associations
- SCTP applications write messages to one of several streams and read messages from these streams
  - each unit is a **chunk**
  - here are record makers ⇒ the receiver **can** tell how much the sender wrote into the stream at any given time
  - multiple streams prevents a loss on one stream from affecting other streams
- SCTP supports multihoming
  - the sender and receiver can utilize multiple interfaces with multiple IP addresses ⇒ increased fault tolerance
  - current implementations do **not** support *load balancing* (i.e., only supports failover)
- SCTP provides reliability
  - via acknowledgements, timeouts, retransmission, …
- SCTP provides flow control
- SCTP tries to avoid causing congestion

SCTP Applications

- Initial goal of IETF Sigtran WG was to support SS7 applications over IP
  - For example, SMS transfer!
  - For an example see [31], [35]
- new applications being developed to use SCTP
  - SIP over SCTP
  - HTTP over SCTP
  - recommended transport protocol for DIAMETER
- Strong security can be provided via TLS [42]

SCTP Header

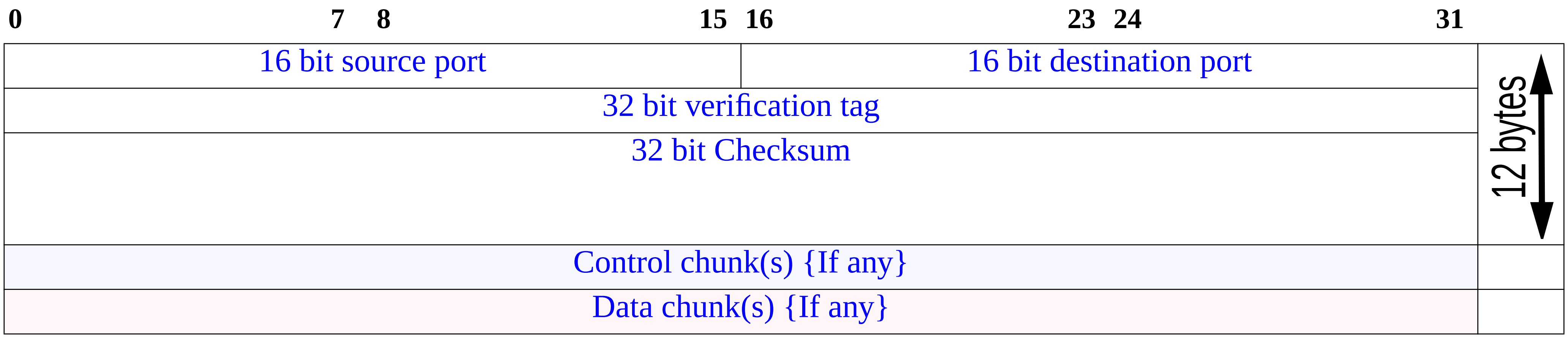


Figure 54: SCTP packet (see Forouzan figure 13.4 pg. 350)

IP protocol x84 = SCTP

- General Header
  - As with UDP & TCP, SCTP provides de/multiplexing via the 16 bit source and destination ports.
  - Associations between end points are defined by a unique **verification tag**
    - A separate verification tag is used in each direction
  - SCTP applies a CRC-32 end-to-end checksum to its general header and all the chunks
    - Previously it used Adler-32 checksum [41]
- Chunks
  - Control information is contained in Control Chunks (these always **precede** any data chunks)
  - Multiple data chunks can be present - each containing data for different streams

SCTP Chunk

0 7	8 15	16 23	24 31
Type	Flag	Length	
Chunk information padded to a multiple of 4 bytes			

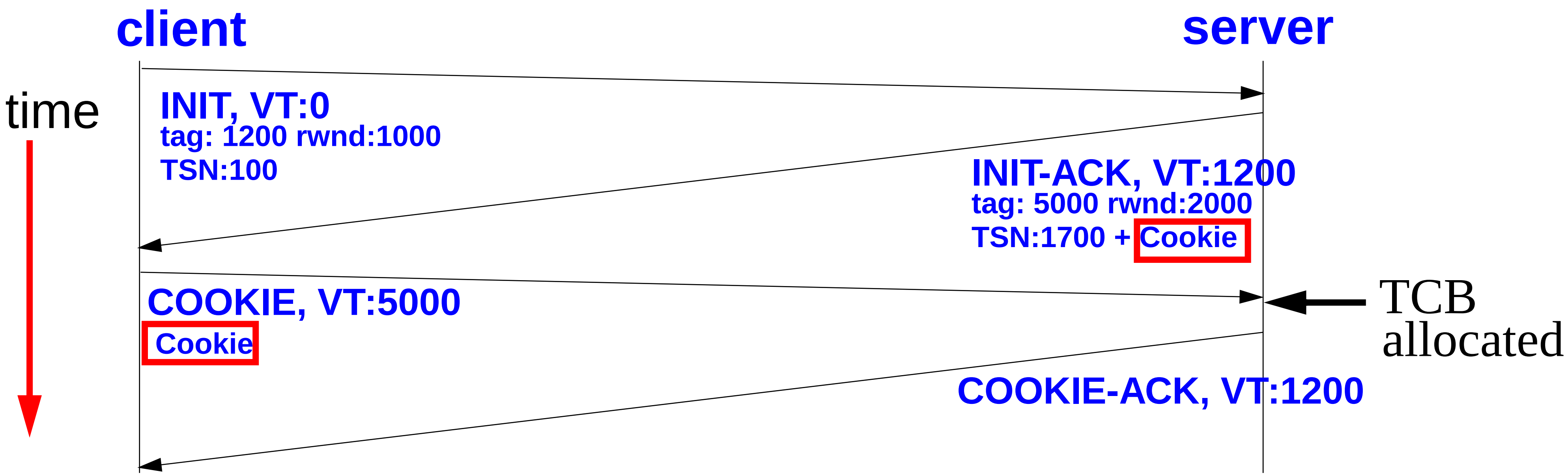
Figure 55: SCTP packet (see Forouzan figure 13.8 pg. 354)

- Type

Type	Chunk	Description	Type	Chunk	Description
0	DATA	User data	7	SHUTDOWN	Terminate an association
1	INIT	Setup an association	8	SHUTDOWN-ACK	Acknowledge SHUTDOWN chunk
2	INIT-ACK	Acknowledge an INIT chunk	9	ERROR	Reports errors without shutting down
3	SACK	Selective Acknowledgement	10	COOKIE ECHO	Third packet in establishment of an association
4	HEARTBEAT	Probe to see if peer is alive	11	COOKIE ACK	Acknowledges COOKIE ECHO chunk
5	HEARTBEAT-ACK	Acknowledgement of a HEARTBEAT chunk	14	SHUTDOWN COMPLETE	Third packet in an association terminations
6	ABORT	Abort an association	192	FORWARD TSN	To adjust the cumulative TSN

- Flag - 8 bit field defined per chunk type
- Length - 16 bit length of chunk including chunk header (i.e., smallest value is 4) - does **not** include any padding bytes (hence you know just how much padding there is)

Association establishment - 4-way handshake



See Forouzan figure 13.19 page 363

The entity initiating the connection is (normally) called the "client" and does an active open, where as the server needed to previous do a passive open.

INIT Chunk

0 – 7	8 – 15	16 – 23	24 – 31
Type = 1	Flag = 0	Length	
Initiation tag			
Advertised receiver window credit (rwnd)			
Outbound streams		Maximum inbound streams	
Initial Transmission sequence number (TSN)			
variable-length parameters (optional)			

Figure 56: SCTP INIT chunk (see Forouzan figure 13.10 pg. 357)

- Initiation tag
  - defines the tags for this association to be used by the other party
  - reduce the risk due to a blind attacker (since there is only a 1 in 2^{32} chance of guessing the right tag)
  - can reject delayed packets - thus avoiding the need for TCP's TIME-WAIT timer
- Advertised receiver window credit
  - defines rwnd (i.e., how much the receiver can send to this party)
- Outbound streams
  - suggested upper number of streams **from** this sender (can be reduced by receiver)
- Maximum inbound streams
  - upper limit of streams **to** this sender

- Transmission sequence number (TSN)
  - Initializes the TSN in the outbound direction, initialized to a random value
- Variable-length parameters
  - IP address(es) of endpoint
    - Multiple addresses are used to support multihoming
    - The **receiver** selects the primary address for the other endpoint
  - Type of addresses
  - Support for Explicit Congestion Notification (ECN)
  - …

INIT ACK Chunk

0 7	8 15	16 23	24 31
Type = 2	Flag = 0	Length	
Initiation tag			
Advertised receiver window credit (rwnd)			
Outbound streams		Maximum inbound streams	
Initial Transmission sequence number (TSN)			
Parameter type: 7		Parameter length	
State Cookie			
variable-length parameters (optional)			

Figure 57: SCTP INIT ACK chunk (see Forouzan figure 13.11 pg. 358)

The same fields as in the INIT chunk (with Initiation tag value set to that of the INIT) - but with the addition of a required parameter with a state cookie.

- Parameter type: 7 = State Cookie
- Parameter length = size of State Cookie + 4 (the parameter type and length fields)

A packet carrying this INIT ACK chunk can not contain any other control or data chunks.

State Cookie

Use of the COOKIE prevents a SYN flood like attack - since resources are not allocated until the COOKIE ECHO chuck is received.

However, state has to be saved from the initial INIT chunk - therefore it is placed in the cookie in a way that only the server can access it (hence the cookie is sealed with an HMAC {aka digest} after being created {aka “baked”}). This requires that the server has a secret key which it uses to compute this digest.

If the sender of the INIT is an attacker located on another machine, they won’t be able to receive the cookie if they faked the source address in the INIT - since the INIT ACK is sent to the address and contains the cookie!

- Without a cookie $\Rightarrow$ no association is created and no resources (such as TCB) are tied up!

COOKIE ECHO Chunk

0 7	8 15	16 23 24 31
Type = 10	Flag = 0	Length
State Cookie		

Figure 58: SCTP COOKIE ECHO chunk (see Forouzan figure 13.12 pg. 359)

- (chunk) Type: 10 = COOKIE ECHO
- (chunk) length = size of State Cookie + 4 (the parameter type and length fields)
- State Cookie
  - simply a copy of the COOKIE data from the INIT ACK chunk
  - The COOKIE data is opaque (i.e., only the sender can read the cookie)

A packet carrying this COOKIE ECHO chunk can contain other control or data chunks -- in particular it can care the first user (client) data!

COOKIE ACK Chunk

0 – 7	8 – 15	16 – 23 / 24 – 31
Type = 11	Flag = 0	Length = 4

Figure 59: SCTP COOKIE ACK chunk (see Forouzan figure 13.13 pg. 359)

Completes the 4 way handshake.

A packet with this chunk can also carry control and data chunks (in particular the first of the user (server) data.

Data Chunk

0 … 7	8 … 15				16 … 23	24 … 31
Type = 0	Reserved	U	B	E	Length	
Transmission sequence number (TSN)						
Stream identifier (SI)					Stream Sequence number (SSN)	
Protocol Identifier						
User data						

Figure 60: SCTP Data Chunk (see Forouzan figure 13.9 pg. 356)

- Flags:
  - U - Unordered - for delivery to the application right away
  - B - Beginning (chunk position - for use with fragmentation)
  - E - End chunk
- Transmission sequence number (TSN)- only data chunks consume TSNs
- Stream identifier (SI)
- Stream Sequence number (SSN)
- Protocol Identifier
- User data
  - at least 1 byte of user data; padded to 32 bit boundaries
  - although a message can be spread over multiple chunks, each chunk contains data from **only** a single message (like UDP, each message results in one or more data SCTP chunks)

Multiple-Streams

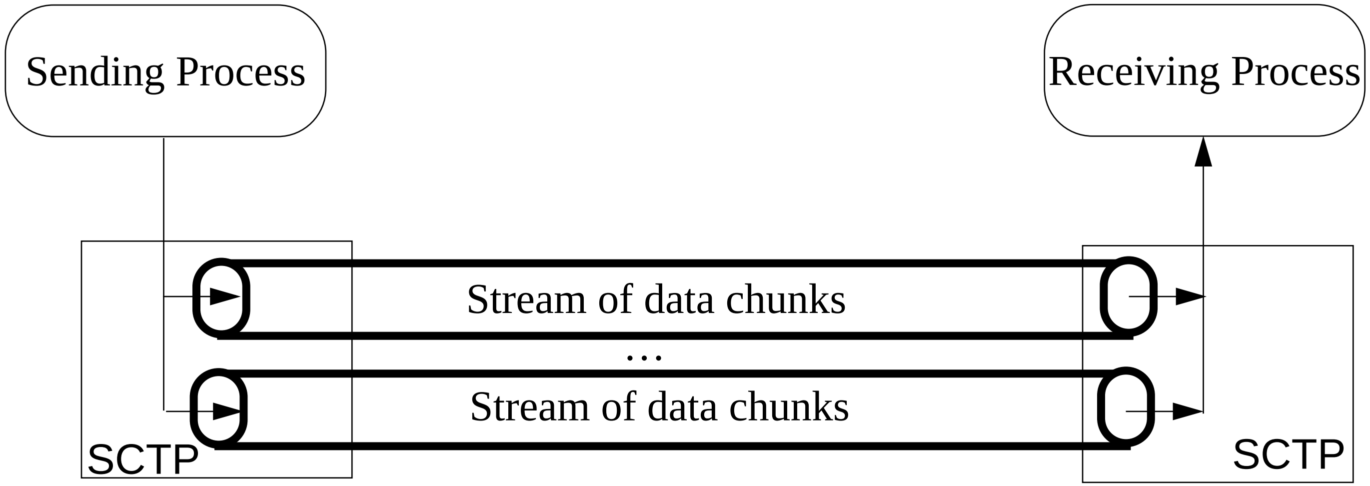


Figure 61: Multiple-streams (see Forouzan figure 13.2 pg. 347)

The figure above shows a **single** association.

Each stream has a unique stream identifier (SI) and maintains its own stream sequence number (SSN).

Unordered data chunks (i.e., with U = 0) - do **not** consume a SSN and are delivered when they arrive at the destination.

Multiple streams and unordered data avoid TCP's head of line blocking.

Selective Acknowledgement (SACK) Chunk

0 … 7	8 … 15	16 … 23	24 … 31
Type = 3	Flag = 0	Length	
cumulative TSN acknowledgement			
Advertised receiver window credit			
Number of gap ACK blocks: N		Number of duplicates: M	
Gap ACK block #1 start TSN offset		Gap ACK block #1 end TSN offset	
…		…	
Gap ACK block #N start TSN offset		Gap ACK block #N end TSN offset	
Duplicate TSN 1			
…			
Duplicate TSN M			

Figure 62: SCTP Data Chunk (see Forouzan figure 13.9 pg. 356)

- Cumulative Transmission sequence number (TSN) acknowledgement - the last data chunk received **in sequence**
- Gap = received sequence of chunks (indicated with start .. end TSNs)
- Duplicate TSN - indicating duplicate chunks (if any)
- SACK always sent to the IP address where the corresponding packet originated

ERROR chunk

Sent when an endpoint finds some error in a packet

0 ... 7	8 ... 15	16 ... 23	24 ... 31
Type = 9	Flag	Length	
One or more causes (optional)			

Figure 63: SCTP ERROR chunk (see Forouzan figure 13.17 pg. 361)

Error code	Description
1	Invalid Stream identifier
2	Missing mandatory parameter
3	State cookie error
4	Out of resource
5	Unresolvable address
6	Unrecognized chunk type
7	Invalid mandatory parameters
8	Unrecognized parameter
9	No user data
10	Cookie received while shutting down

Association Termination

Two forms of termination

- Association Abort
  - Used in the event of a fatal error
  - uses same error codes as the ERROR Chunk
  - Chunk format

0 ... 7	8 ... 15	16 ... 23 24 ... 31
Type = 6	Flag = 6	Length
One or more causes (optional)		

Figure 64: SCTP ABORT chunk (see Forouzan figure 13.18 pg. 362)

- Association Shutdown - graceful termination

Association Shutdown

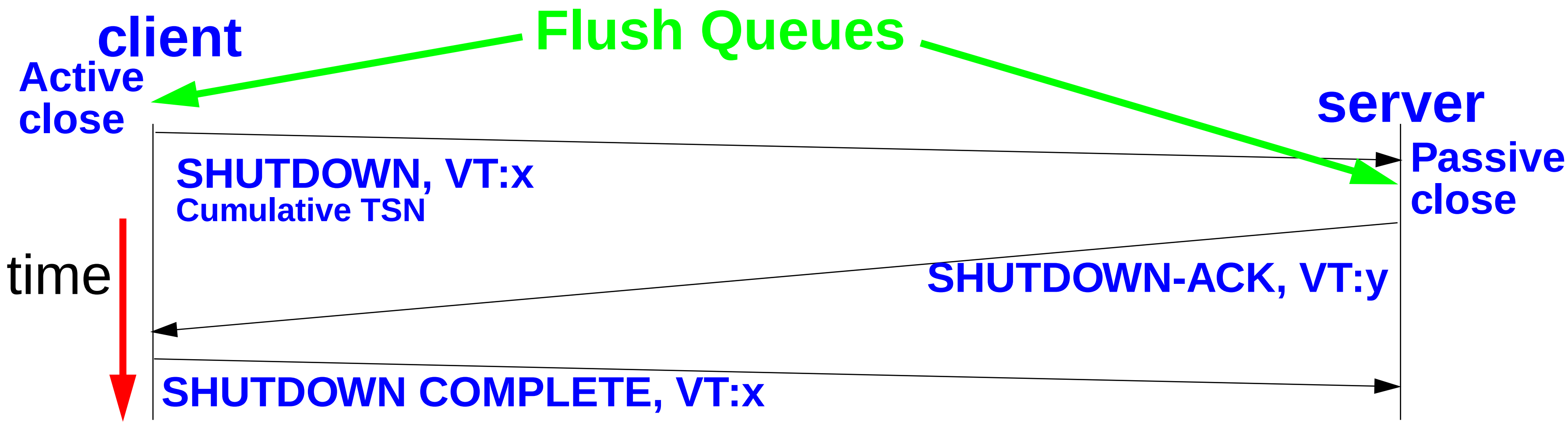


Figure 65: Adapted from Forouzan figure 13.21 page 368 and slide 15 of [32]

0 ... 7	8 ... 15	16 ... 23	24 ... 31
Type = 7	Flag	Length = 8	
cumulative TSN acknowledgement			

Figure 66: SCTP SHUTDOWN chunk (see Forouzan figure 13.16 pg. 361)

0 ... 7	8 ... 15	16 ... 23	24 ... 31
Type = 8	Flag	Length = 4	

Figure 67: SCTP SHUTDOWN ACK chunk (see Forouzan figure 13.16 pg. 361)

0 7	8	15	16 23 24 31
Type = 14	Flag	T	Length = 4

Figure 68: SCTP SHUTDOWN COMPLETE chunk (see Forouzan figure 13.16 pg. 361)

- T bit indicates the sender did **not** have a Transmission Control Block (TCB)

SCTP Example - Daytime [36]

```
server# ./daytime_server -s 192.168.1.2 -vv
1       : Communication up (1 paths)
1       : Network status change: path 0 is now REACHABLE
1       : Shutdown complete

client# ./terminal -vv -r 13 -d 192.168.1.2 -s 192.168.1.1
1       : Communication up (1 paths, 1 In-Streams, 1 Out-Streams)
1       : Network status change: path 0 (towards 192.168.1.2) is now REACHABLE
Wed Apr 27 11:52:04 2005
1       : Shutdown received
```

11	74.864232	192.168.1.1	192.168.1.2	SCTP	INIT
12	74.864552	192.168.1.2	192.168.1.1	SCTP	INIT_ACK
13	74.864808	192.168.1.1	192.168.1.2	SCTP	COOKIE_ECHO
14	74.865073	192.168.1.2	192.168.1.1	SCTP	COOKIE_ACK
15	74.865273	192.168.1.2	192.168.1.1	SCTP	DATA
16	74.865733	192.168.1.1	192.168.1.2	SCTP	SACK
17	74.865933	192.168.1.2	192.168.1.1	SCTP	SHUTDOWN
18	74.866132	192.168.1.1	192.168.1.2	SCTP	SHUTDOWN_ACK
19	74.866195	192.168.1.2	192.168.1.1	SCTP	SHUTDOWN_COMPLETE

Figure 69: SCTP Daytime example - output from Ethereal

ethereal capture - daytime - INIT

```
Frame 11 …
Stream Control Transmission Protocol
    Source port: 10777
    Destination port: 13
    Verification tag: 0x00000000
    Checksum: 0x2b84fdb0[1]
    INIT chunk (Outbound streams: 10, inbound streams: 10)
        Chunk type: INIT (1)
        Chunk flags: 0x00
        Chunk length: 32
        Initiate tag: 0x43d82c5d
        Advertised receiver window credit (a_rwnd): 131071
        Number of outbound streams: 10
        Number of inbound streams: 10
        Initial TSN: 771212194
        Forward TSN supported parameter
            Parameter type: Forward TSN supported (0xc000)
…            Parameter length: 4
        Supported address types parameter (Supported types: IPv4)
            Parameter type: Supported address types (0x000c)
            Parameter length: 6
            Supported address type: IPv4 address (5)
```

1. Ethereal complains about this checksum saying “(incorrect Adler32, should be 0x973b078d)”, but this is in error see [41].

ethereal capture - daytime - INIT-ACK

```
Frame 12 … Stream Control Transmission Protocol
    Source port: 13
    Destination port: 10777
    Verification tag: 0x43d82c5d
    Checksum: 0x7f61f237
    INIT_ACK chunk (Outbound streams: 1, inbound streams: 1)
        Chunk type: INIT_ACK (2)
        Chunk flags: 0x00
        Chunk length: 128
        Initiate tag: 0x5d581d9a
        Advertised receiver window credit (a_rwnd): 131071
        Number of outbound streams: 1
        Number of inbound streams: 1
        Initial TSN: 1514529259
        State cookie parameter (Cookie length: 100 bytes)
            Parameter type: State cookie (0x0007)
               Parameter length: 104
            State cookie: 5D581D9A0001FFFF000100015A45E1EB...
        Forward TSN supported parameter
            Parameter type: Forward TSN supported (0xc000)
                1... .... .... .... = Bit: Skip parameter and con-
tinue prosessing of the chunk
                .1.. .... .... .... = Bit: Do report
            Parameter length: 4
```

ethereal capture - daytime - COOKIE-ECHO

```
Frame 13 …
    Source port: 10777
    Destination port: 13
    Verification tag: 0x5d581d9a
    Checksum: 0x3af3f579
    COOKIE_ECHO chunk (Cookie length: 100 bytes)
        Chunk type: COOKIE_ECHO (10)
            0... .... = Bit: Stop processing of the packet
            .0.. .... = Bit: Do not report
        Chunk flags: 0x00
        Chunk length: 104
        Cookie: 5D581D9A0001FFFF000100015A45E1EB...
```

ethereal capture - daytime - COOKIE-ACK

```
Frame 14 …
          Source port: 13
          Destination port: 10777
          Verification tag: 0x43d82c5d
          Checksum: 0x762d80d7
          COOKIE_ACK chunk
              Chunk type: COOKIE_ACK (11)
              Chunk flags: 0x00
              Chunk length: 4
```

ethereal capture - daytime - DATA

```
Frame 15 …
          Source port: 13
          Destination port: 10777
          Verification tag: 0x43d82c5d
          Checksum: 0xf8fb1754
          DATA chunk(ordered, complete segment, TSN: 1514529259, SID: 0,
SSN: 0, PPID: 0, payload length: 25 bytes)
              Chunk type: DATA (0)
              Chunk flags: 0x03
                  .... ...1 = E-Bit: Last segment
                  .... ..1. = B-Bit: First segment
                  .... .0.. = U-Bit: Ordered deliviery
              Chunk length: 41
              TSN: 1514529259
              Stream Identifier: 0x0000
              Stream sequence number: 0
              Payload protocol identifier: not specified (0)
              Chunk padding: 000000
Data (25 bytes)
0000  57 65 64 20 41 70 72 20 32 37 20 31 31 3a 34 33   Wed Apr 27 11:43
0010  3a 32 32 20 32 30 30 35 0a                        :22 2005.
```

ethereal capture - daytime - SACK

```
Frame 16 …
    Source port: 10777
    Destination port: 13
    Verification tag: 0x5d581d9a
    Checksum: 0xfa994e35
   SACK chunk (Cumulative TSN: 1514529259, a_rwnd: 131071, gaps: 0,
duplicate TSNs: 0)
        Chunk type: SACK (3)
        Chunk flags: 0x00
        Chunk length: 16
        Cumulative TSN ACK: 1514529259
        Advertised receiver window credit (a_rwnd): 131071
        Number of gap acknowldgement blocks : 0
        Number of duplicated TSNs: 0
```

ethereal capture - daytime - SHUTDOWN

```
Frame 17 …
          Source port: 13
          Destination port: 10777
          Verification tag: 0x43d82c5d
          Checksum: 0xf447d00f
          SHUTDOWN chunk (Cumulative TSN ack: 771212193)
              Chunk type: SHUTDOWN (7)
              Chunk flags: 0x00
              Chunk length: 8
              Cumulative TSN Ack: 771212193
```

ethereal capture - daytime - SHUTDOWN_ACK

```
Frame 18
        Source port: 10777
        Destination port: 13
        Verification tag: 0x5d581d9a
        Checksum: 0x9f44d056
        SHUTDOWN_ACK chunk
            Chunk type: SHUTDOWN_ACK (8)
            Chunk flags: 0x00
            Chunk length: 4
```

ethereal capture - daytime - SHUTDOWN_COMPLETE

```
Frame 19…
     Source port: 13
     Destination port: 10777
     Verification tag: 0x43d82c5d
     Checksum: 0x3db6e771
     SHUTDOWN_COMPLETE chunk
         Chunk type: SHUTDOWN_COMPLETE (14)
         Chunk flags: 0x00
             .... ...0 = T-Bit: TCB destroyed
         Chunk length: 4
```

Fault Management

- Endpoint Failure Detection
  - Endpoint keeps a counter of the total number of consecutive retransmissions to its peer (including retransmissions to all the destination transport addresses [= port + IP address] of the peer if it is multi-homed). When this counter exceeds 'Association.Max.Retrans', the endpoint will consider the peer endpoint unreachable and shall stop transmitting any more data to it (the association enters the CLOSED state).
  - Counter is reset each time:
    - a DATA chunk sent to that peer is acknowledged (by the reception of a SACK) or
    - a HEARTBEAT-ACK is received from the peer
- Path Failure Detection
  - Each time (1) T3-rtx timer expires on any address or (2) a HEARTBEAT sent to an idle address is not acknowledged within a RTO, then the error counter of that destination will be incremented. When this error counter exceeds 'Path.Max.Retrans' for that destination address, then the endpoint marks the destination transport address as inactive and notifies the upper layer.
  - the endpoint clears the error counter of this destination transport address when:
    - an outstanding TSN is acknowledged or
    - a HEARTBEAT address is acknowledged
  - When the primary path is marked inactive, then the sender **may** automatically transmit new packets to an alternate destination address if one exists and is active
    - If more than one alternate address is active $\Rightarrow$ only **one** transport address is chosen as the new destination transport address.

HEARTBEAT and HEARTBEAT ACK Chunks

0 7	8 15	16 23	24 31
Type = 4 or 5	Flag = 0	Length	
Parameter type: 1		Parameter length	
Sender specific information			

Figure 70: SCTP HEARTBEAT and HEARTBEAK ACK chunks (see Forouzan figure 13.15 pg. 360)

- (chunk) Type: 4 = HEARTBEAT
- (chunk) Type: 5 = HEARTBEAT ACK
- (chunk) length = size of sender specific information + 4 (the parameter type and length fields)
- Sender specific information
  - The sender puts its Local time and transport address in (note that the sctplib implementation 1.0.2 puts the time in as an unsigned 32 bit integer and puts the path index in (also as an unsigned 32 bit integer) and add a HMAC computed over these values [36]
  - The acknowledgement simply contains a copy of this information

Heartbeats every ~30 seconds.

Heartbeat and ACK

```
Frame x …
    Source port: 9
    Destination port: 38763
    Verification tag: 0x36fab554
  Checksum: 0x0e6c8d88 (incorrect Adler32, should be 0xf5340ec5)
    HEARTBEAT chunk (Information: 28 bytes)
        Chunk type: HEARTBEAT (4)
        Chunk flags: 0x00
        Chunk length: 32
        Heartbeat info parameter (Information: 24 bytes)
            Parameter type: Heartbeat info (0x0001)
            Parameter length: 28
            Heartbeat information: 0280351E00000000E1A06CFBC1C6933F...
    Source port: 38763
    Destination port: 9
    Verification tag: 0x57c3a50c
    Checksum: 0xaa2fba80 (incorrect Adler32, should be 0xe7450e58)
    HEARTBEAT_ACK chunk (Information: 28 bytes)
        Chunk type: HEARTBEAT_ACK (5)
        Chunk flags: 0x00
        Chunk length: 32
        Heartbeat info parameter (Information: 24 bytes)
            Parameter type: Heartbeat info (0x0001)
            Parameter length: 28
            Heartbeat information: 0280351E00000000E1A06CFBC1C6933F...
```

Differences from TCP Congestion Control

- Any DATA chunk that has been acknowledged by SACK, including DATA that arrived out of order, are **only** considered fully delivered when the Cumulative TSN Ack Point passes the TSN of the DATA chunk

⇒ cwnd controls the amount of outstanding data, rather than (as in the case of non-SACK TCP) the upper bound between the highest acknowledged sequence number and the latest DATA chunk that can be sent within the congestion window

⇒ different fast-retransmit & fast-recovery than non-SACK TCP

- Retransmission based on both retransmission timer (with an RTO per path)
- Three SACKS (i.e., 4 consecutive duplicate SACKs indicating missing chunks) fi immediate retransmission of these missing chunks

Sender

- uses the same destination address until instructed by the upper layer (however, SCTP may change to an alternate destination in the event an address is marked inactive) ⇒ retransmission can be to a different transport address than the original transmission.
- keeps separate congestion control parameters (cwnd, ssthresh, and partial_bytes_acked) for each of the destination addresses it can send to (i.e., not each source-destination pair)
  - these parameters should decay if the address is not used
  - does slow-start upon the first transmission to each of destination addresses

Path MTU Discovery

- IPv4
  - Based on RFC 1191 [38] each endpoint maintains an estimate of the maximum transmission unit (MTU) along a **each** path and refrains from sending packets along that path which exceed the MTU, other than occasional attempts to probe for a change in the Path MTU (PMTU).
- IPv6
  - Based on RFC1981 [39] an SCTP sender using IPv6 **must** use Path MTU Discovery, unless all packets are less than the minimum IPv6 MTU (see RFC 2460 [40]).

SCTP differs in several ways from the description in RFC 1191 of applying MTU discovery to TCP:

1 SCTP associations can span multiple addresses ⇒ an endpoint does PMTU discovery on a per-destination-address basis
  - The term “MTU” always refers to the MTU associated with the destination address

2 Since SCTP does not have a notion of “Maximum Segment Size”, for each destination $MTU_{initial} \leq MTU_{link}$ for the local interface to which packets for that remote destination address will be routed

3 When retransmitting to a remote address for which the IP datagram appears too large for the path MTU to that address, the IP datagram **should** be retransmitted without the DF bit set, enabling it to be fragmented. While *initial* transmissions of IP datagrams **must** have DF set.

4 Sender maintains an association PMTU (= smallest PMTU discovered for all of the peer's destination addresses); when fragmenting messages this association PMTU is used to calculate the size of each fragment ⇒ retransmissions can sent to an alternate address without encountering IP fragmentation

SCTP header continued

- **Reliability** is provided by a 32 bit SCTP sequence numbers (TSN)
  - The initial sequence number is a random 32 bit number
  - These sequence numbers are in the header of individual chunks
  - This cumulative number is used to provide both flow control and error control
- SCTP resequences data at the receiving side
- SCTP discards duplicate data at the receiving side

The window size (or more exactly the receive window size (rwnd)) - indicates how many bytes the receiver is prepared to receive (this number is relative to the acknowledgement number).

Forward Cumulative TSN

Allows an endpoint to signal to its peer that it should move the cumulative acknowledgement forward [37]. This protocol extension adds a new parameter (Forward-TSN-Supported) to INIT and INIT ACK, and a new FORWARD TSN chunk type. It provides an example of a partially reliable service.

0 … 7	8 … 15	16 … 23	24 … 31
Type = 192	Flag = 0	Length	
New cumulative TSN			
Stream #1		Stream Sequence #1	
…		…	
Stream #N		Stream Sequence #N	

Figure 71: SCTP FORWARD TSN Chunk (see [37])

- Stream_i a stream number that was skipped by this FWD-TSN.
- Stream Sequence_i = the largest stream sequence number in stream_i being skipped

- Receiver can use the Stream_i and Stream Sequence_i fields to enable delivery of (stranded) TSN’s that remain in the stream re-ordering queues.

SCTP Performance

See the upcoming exjobb report by Mia Immonen.

Transport Protocol Functional Overview

From table 1-1 of [34] on page 12; also appears in [43]

Protocol Feature	SCTP	TCP	UDP
State required at each endpoint	Yes	Yes	No
Reliable data transfer	Yes	Yes	No
Congest control and avoidance	Yes	Yes	No
Message boundary conservation	Yes	No	Yes
Path MTU discovery and message fragmentation	Yes	Yes	No
Message bundling	Yes	Yes	No
Multi-homed hosts support	Yes	No	No
Multi-stream support	Yes	No	No
Unordered data delivery	Yes	No	Yes
Security cookie against SYN flood attack	Yes	No	No
Built-in heartbeat (readability check)	Yes	No	No

Summary

This lecture we have discussed:

- SCTP
  - Message framing
  - Multi-homing
  - Multi-streaming
- How SCTP differs from TCP
- Measurements of an implementation (there are other implementations such as that included with [34]):
  - *http://www.sctp.de*
  - *http://www.sctp.org*
  - Linux Kernel SCTP *http://sourceforge.net/projects/lksctp*

References

[31] G. Sidebottom, K. Morneault, and J. Pastor-Balbas, “Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) - User Adaptation Layer (M3UA)”, IETF RFC 3332, September 2002 *http://www.ietf.org/rfc/rfc3332.txt*

[32] Andreas Jungmaier, “A Gentle Introduction to SCTP”, 19th Chaos Communications Congress, Berlin, 2002

http://tdrwww.exp-math.uni-essen.de/inhalt/forschung/19ccc2002/html/slide_1.html

[33] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, “Stream Control Transmission Protocol”, IETF RFC 2960, October 2000 *http://www.ietf.org/rfc/rfc2960.txt*

[34] Randall R. Stewart and Qiaobing Xie, “Stream Control Transmission Protocol: A Reference Guide”, Addison-Wesley, 2002, ISBN 0-201-72186-4.

[35] K. Morneault, S. Rengasami, M. Kalla, and G. Sidebottom, “ISDN

Q.921-User Adaptation Layer", IETF RFC 3057, February 2001
http://www.ietf.org/rfc/rfc3057.txt

[36] Andreas Jungmaier , Herbert Hölzlwimmer, Michael Tüxen , and Thomas Dreibholz, "sctplib-1.0.2", Siemens AG and the Institute of Computer Networking Technology, University of Essen, Germany, August 2004 *http://www.sctp.de/sctp-download.html* {Note that a later version 1.0.3 was released March 4th, 2005}

[37] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", IETF RFC 3758, May 2004 *http://www.ietf.org/rfc/rfc3758.txt*

[38] J. Mogul and S. Deering, "Path MTU Discovery", IETF RFC 1191, November 1990 *http://www.ietf.org/rfc/rfc1191.txt*

[39] J. McCann, S. Deering, and J. Mogul, "Path MTU Discovery for IP version 6", IETF RFC 1981, August 1996 *http://www.ietf.org/rfc/rfc1981.txt*

[40] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6)

Specification", IETF RFC 2460, December 1998 *http://www.ietf.org/rfc/rfc2460.txt*

[41] J. Stone, R. Stewart, and D. Otis, "Stream Control Transmission Protocol (SCTP) Checksum Change", IETF RFC 3309, September 2002

http://www.ietf.org/rfc/rfc3309.txt

[42] A. Jungmaier, E. Rescorla, and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", IETF RFC 3436, December 2002

http://www.ietf.org/rfc/rfc3436.txt

[43] "SCTP Primer", Mon, Mar 1, 2004 03:35:54 PM

http://datatag.web.cern.ch/datatag/WP3/sctp/primer.htm

[44] Mia Immonen, "SIGTRAN: Signaling over IP a step closer to an all-IP network", Masters thesis, Royal Institute of Technology (KTH), Dept. of Communication Systems, June 2005

ftp://ftp.it.kth.se/Reports/DEGREE-PROJECT-REPORTS/050619-Mia-Immonen-with-cover.pdf