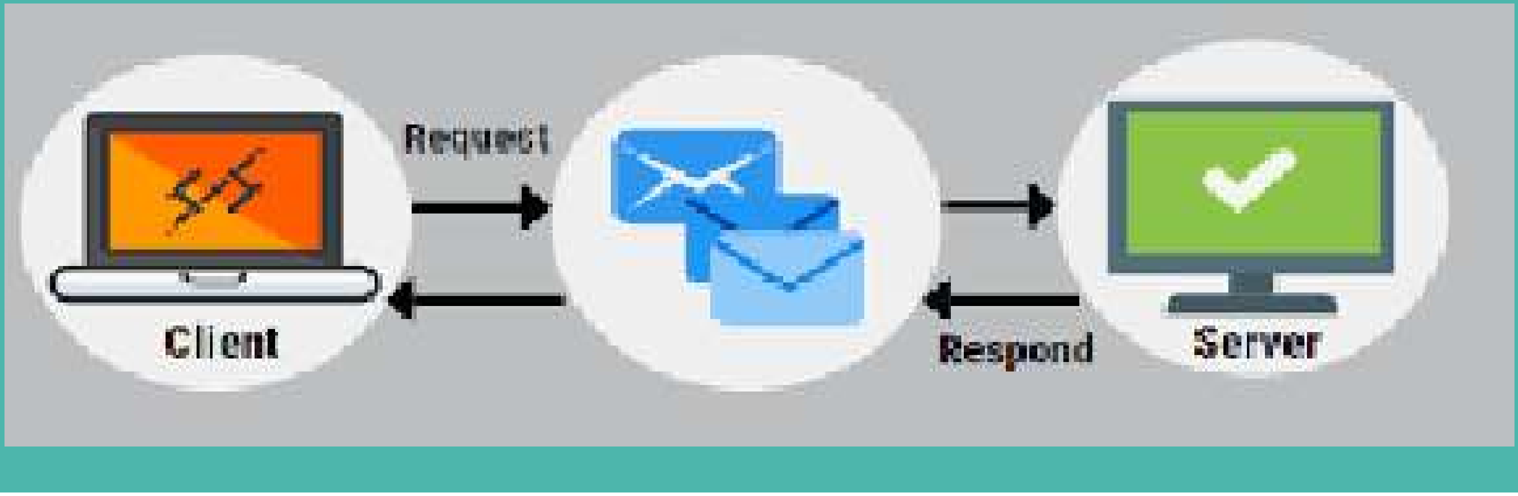
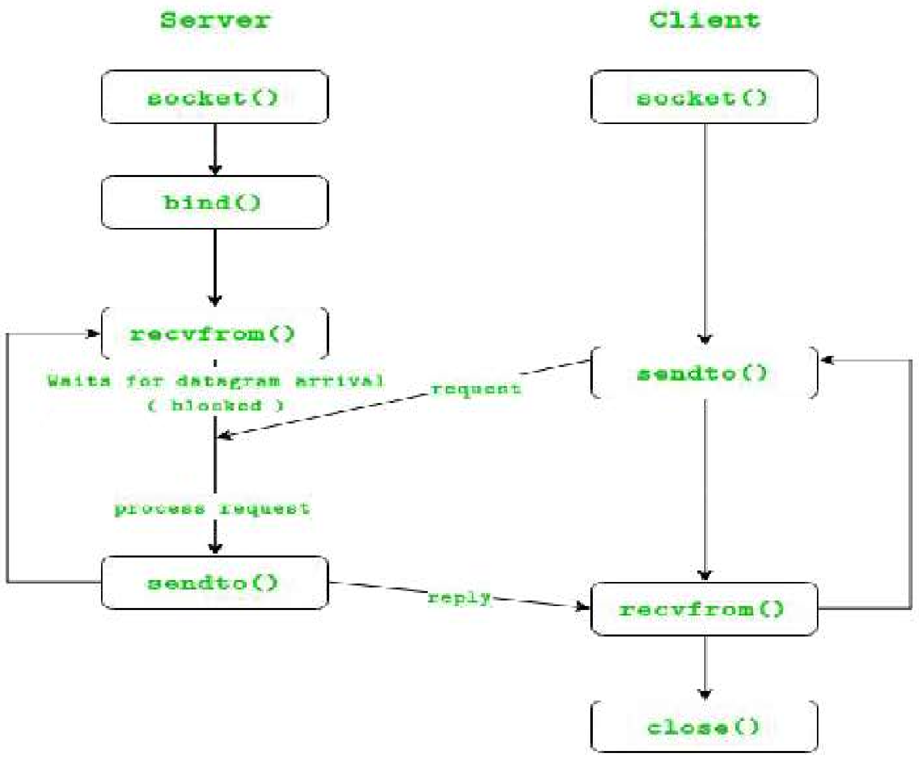
UDP CLIENT – SERVER AND PACKAGES



Client-Server Program



Source:
https://www.geeksforgeeks.org/udp-server-client-implementation-c/

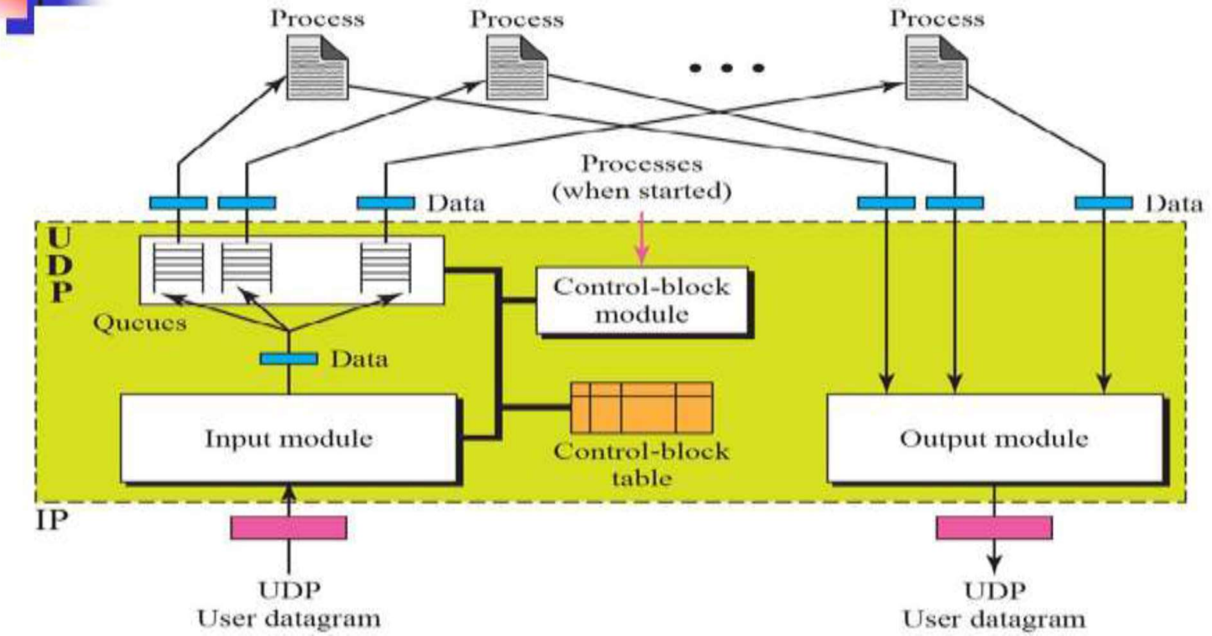
Server Processing using UDP

1. Create UDP socket.
2. Bind the socket to server address.
3. Wait until datagram packet arrives from client.
4. Process the datagram packet and send a reply to client.
5. Go back to Step 3.

Client Processing using UDP

1. Create UDP socket.
2. Send message to server.
3. Wait until response from server is received.
4. Process reply and go back to step 2, if necessary.
5. Close socket descriptor and exit.

UDP Package



Source: https://www.slideshare.net/Thanveen/user-datagram-protocol-for-msc-cs-42037663

Five components

- Control-block table
- Input queues
- Control-block module
- Input module
- Output module

Control – block Table

- Used to keep track of open ports
- Each entry in the table has 4 entries
    - ➢ State – FREE / IN-USE
    - ➢ Process Id
    - ➢ Port number
    - ➢ Corresponding queue number

Input Queues

- Each process has a set of input queue.
- We do not use output queue

Control- block module

- It is responsible for management of control block table.
- When a process starts, it asks for port no from OS.
- The OS assigns well known ports for server and ephemeral ports for clients.
- The process passes the Process Id and port no to the control block module to create an entry in the table for a process.
- The value of the field queue is zero since the control- block module does not create queues.

Control – block module contd.

```
UDP_Control_Block_Module (process ID, port number)
{
    Search the table for a FREE entry.
    if (not found)
        Delete one entry using a predefined strategy.
    Create a new entry with the state IN_USE
    Enter the process ID and the port number.
    Return.
} //end module
```

Input module

- It receives a user datagram from IP.
- It searches the control block table to find an entry having the same port number as this user datagram
  - ➢ If found -> the module uses the info in the entry to enqueue the data.
  - ➢ If not found -> it generates an ICMP message.

Input – module contd.

```
UDP_INPUT_Module(user_datagram)
{
    Look for the entry in the control_block table
    if (found)
    {
        Check to see if a queue is allocated
        Allocate a queue
    else
        Enqueue the data
} //end if
else
{
    Ask ICMP to send an “unreachable port” message
    Discard the user datagram
}//end else
Return
} //end module
```

Output module

It is responsible for creating and sending user datagrams

```
UDP_OUTPUT_MODULE(Data)
{
Create a user datagram
Send the user datagram
Return.
}
```

Stream Control Transmission Protocol (SCTP)

- SCTP is designed as a general-purpose transport layer protocol that can handle multimedia and stream traffic, which are increasing every day on the Internet.
- It is a new reliable, message-oriented transport-layer protocol.

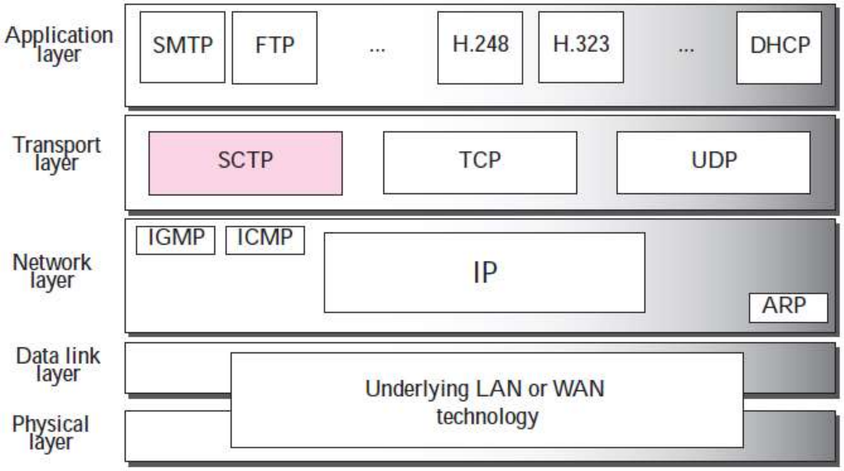


Fig: Relationship of SCTP to the other protocols in the Internet protocol suite

Stream Control Transmission Protocol (SCTP)

Table: Comparison between UDP, TCP, and SCTP

UDP	TCP	SCTP
Message-oriented protocol	Byte-oriented protocol	Best features of UDP and TCP
UDP conserves the message boundaries	No preservation of the message boundaries	Preserves the message boundaries along with detection of lost data, duplicate data, and out-of-order data
UDP is unreliable	TCP is a reliable protocol	SCTP is a reliable message oriented Protocol
Lacks in congestion control and flow control	TCP has congestion control and flow control mechanisms	It has congestion control and flow control mechanisms

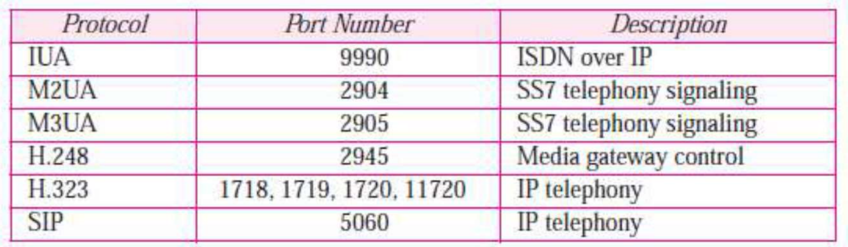
*** SCTP is a *message-oriented, reliable* protocol that combines the best features of UDP and TCP.**

SCTP Services

➢ Process-to-Process Communication

- SCTP uses all well-known ports in the TCP space

Table : Some SCTP applications



Protocol	*Port Number*	*Description*
IUA	9990	ISDN over IP
M2UA	2904	SS7 telephony signaling
M3UA	2905	SS7 telephony signaling
H.248	2945	Media gateway control
H.323	1718, 1719, 1720, 11720	IP telephony
SIP	5060	IP telephony

➢ Multiple Streams

- SCTP allows multistream service in each connection called as association
- If one of the streams is blocked, the other streams can still deliver their data

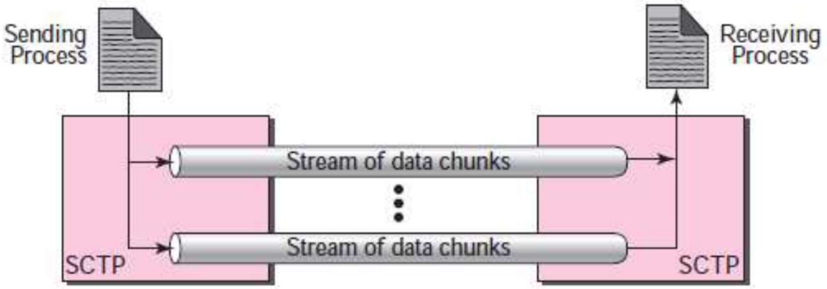


Fig: Multiple-stream concept

SCTP Services

➢ Multihoming

- The sending and receiving host can define multiple IP addresses in each end for an association.
- when one path fails, another interface can be used for data delivery without interruption.
- This fault-tolerant feature is helps on sending and receiving a real-time payload such as Internet telephony.

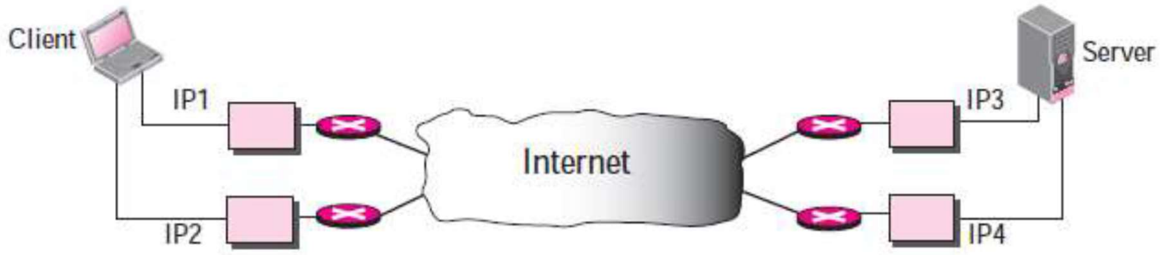


Fig: Multihoming concept

➢ Full-Duplex Communication

- SCTP has sending and receiving buffer, hence packets are sent in both directions.

SCTP Services

➢ **Connection-Oriented Service**

- A connection is called an association in SCTP
- Steps to send and receive data in SCTP
    1. The two SCTPs establish an association between each other.
    2. Data are exchanged in both directions.
    3. The association is terminated.

➢ **Reliable Service**

- It uses an acknowledgment mechanism to check the safe and sound arrival of data.

SCTP Features

➢ **Transmission Sequence Number (TSN)**

- Unit of data in SCTP is a data chunk
- Data transfer in SCTP is controlled by numbering the data chunks
- SCTP uses a TSN to number the data chunks with 32 bits long and randomly initialized between 0 and $2^{32}– 1$
- Each data chunk carry their TSN in its header

➢ **Stream Identifier (SI)**

- Each stream in SCTP needs to be identified using a SI
- Each data chunk carry SI in its header
- when it arrives at the destination, it is placed in order in its stream
- The SI is a 16-bit number starting from 0

SCTP Features

- **Stream Sequence Number (SSN)**

- When a data chunk arrives at the destination SCTP, it is delivered to the appropriate stream in the proper order.
- In addition to an SI, SCTP defines a SSN in each data chunk in each stream

- **Packets**

- Data are carried as data chunks, control information as control chunks
- Several control chunks and data chunks can be packed together in a packet

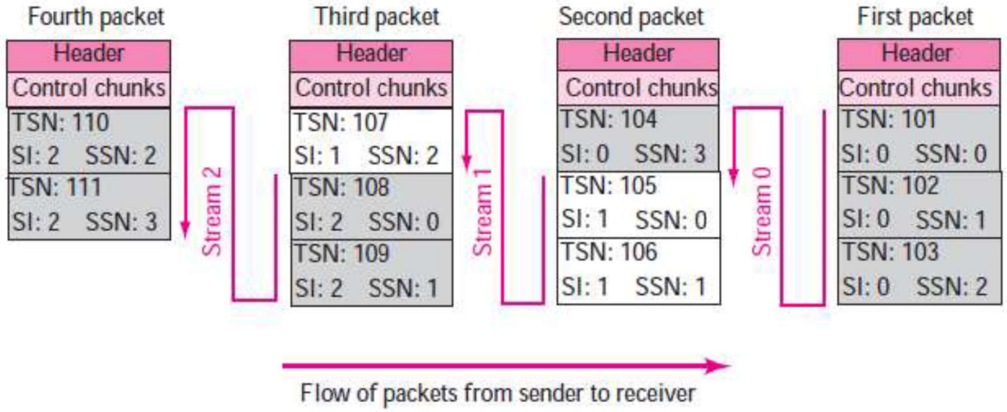
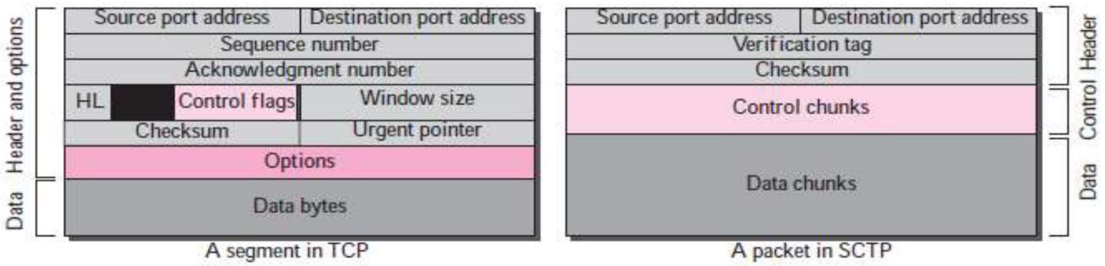


Fig: Packet, data chunks, and streams

SCTP Features



Comparison between a TCP segment and an SCTP packet

TCP segment	SCTP packet
Control information is part of the header	Control information is included in the control chunks
Data is treated as one entity	Carry several data chunks, each can belong to a different stream
Options section exist separately	Options are handled by defining new chunk types
Mandatory part of header is 20 bytes	General header is only 12 bytes
Checksum is 16 bits	Checksum is 32 bits
Combination of IP and port addresses define a connection	Verification tag is an association identifier
Includes one sequence number in the header	Includes several different data chunks
Some segments carry control information	Control chunks never use a TSN, IS, or SSN number, they are used for data chunks only

SCTP Features

➢ **Acknowledgment Number**

- SCTP acknowledgment numbers are chunk-oriented refer to TSN
- Control information is carried by control chunks, which do not need a TSN
- Control chunks are acknowledged by another control chunk of the appropriate type

➢ **Flow Control**

- SCTP implements flow control to avoid overwhelming the receiver

➢ **Error Control**

- TSN numbers and acknowledgment numbers are used for error control.

➢ **Congestion Control**

- SCTP implements congestion control to determine how many data chunks can be injected into the network

SCTP Packet Format

- Main Parts are
  - General header
  - Chunks – set of blocks

- Types of chunks
  - Control chunks - controls and maintains the association
  - Data chunks - carries user data

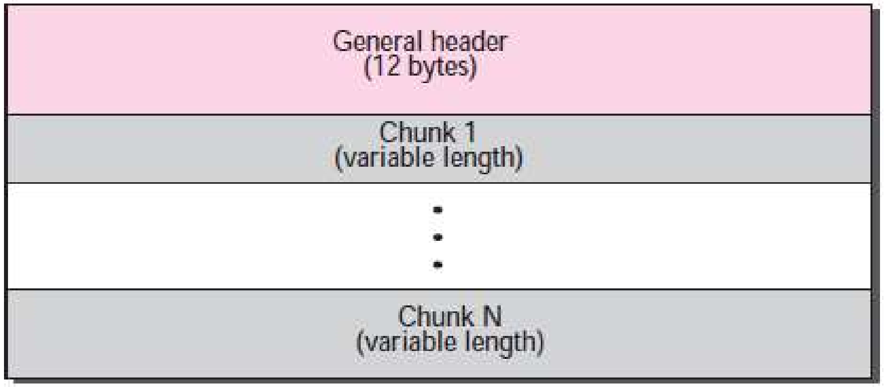
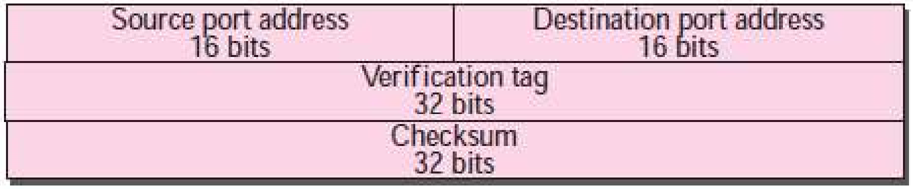


Fig: SCTP packet format

SCTP Packet Format

- ➢ **General Header**
- Defines the end points of each association to which the packet belongs
- Guarantees for a packet belongs to a particular association
- Preserves the integrity of the contents of the packet
- ➢ There are four fields in the general header
- **Source port address:** 16-bit field defines the port number of the sender process
- **Destination port address:** 16-bit field defines the port number of the receiving process
- **Verification tag:** Number that matches a packet to an association
    - It serves as an identifier for the association
    - Separate verification used for each direction in the association.
- **Checksum:** 32-bit field contains a CRC-32 checksum



Source port address 16 bits | Destination port address 16 bits

Verification tag 32 bits

Checksum 32 bits

Fig: General header

SCTP Packet Format

➢ **Chunks**

- Control information or user data are carried
- First three fields are common to all chunks
    - **Type:** 8-bit field define up to 256 types of chunks(few have been defined, rest are reserved for future use)
    - **Flag:** 8-bit field defines special flags that a particular chunk may need.
    - **Length:** 16-bit field defines the total size of the chunk, in bytes, including the type, flag, and length fields

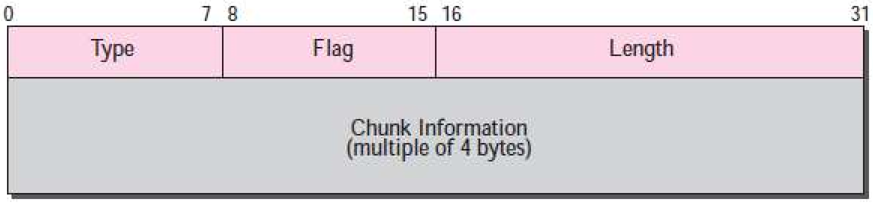
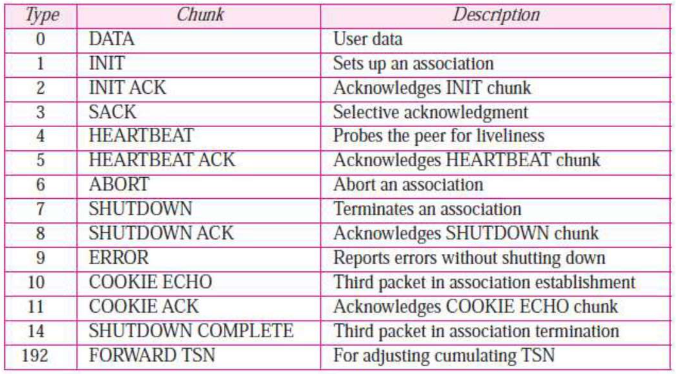


Fig: Common layout of a chunk

Table: Types of Chunks



Type	*Chunk*	*Description*
0	DATA	User data
1	INIT	Sets up an association
2	INIT ACK	Acknowledges INIT chunk
3	SACK	Selective acknowledgment
4	HEARTBEAT	Probes the peer for liveliness
5	HEARTBEAT ACK	Acknowledges HEARTBEAT chunk
6	ABORT	Abort an association
7	SHUTDOWN	Terminates an association
8	SHUTDOWN ACK	Acknowledges SHUTDOWN chunk
9	ERROR	Reports errors without shutting down
10	COOKIE ECHO	Third packet in association establishment
11	COOKIE ACK	Acknowledges COOKIE ECHO chunk
14	SHUTDOWN COMPLETE	Third packet in association termination
192	FORWARD TSN	For adjusting cumulating TSN

- Information field depends on the type of chunk
- SCTP requires the information section to be a multiple of 4 bytes
    - If not, padding bytes (eight 0s) are added at the end of the section

SCTP Packet Format

➢ **Data**

- Carries the user data
- A packet may contain zero or more data chunks
- Common fields
  - Type field has a value of 0
  - Flag field has 5 reserved bits and 3 defined bits
    - U - signals unordered data
    - B - beginning bit of fragmented message
    - E - end bit of fragmented message

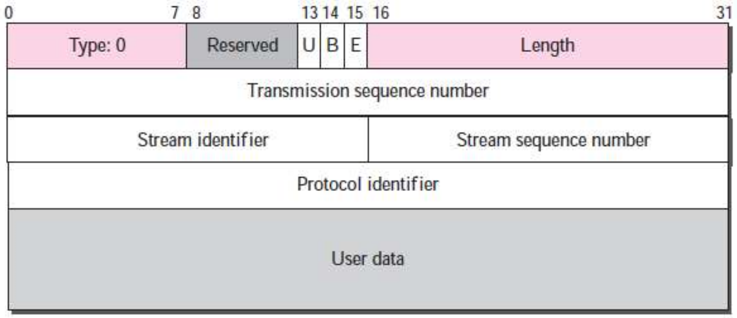


Fig: DATA chunk

- TSN - Sequence number initialized in an INIT chunk for one direction and in the INIT ACK chunk for the opposite direction
- SI - all chunks of same stream in one direction have same stream identifier
- Protocol identifier: 32-bit field used by the application program to define the type of data which is ignored by SCTP
- User data: carries the actual user data
  - No chunk can carry data belonging to more than one message
  - A message can be spread over several data chunks
  - Must have at least one byte of user data, can’t be empty
  - If the data cannot end at a 32-bit boundary, padding must be added

SCTP Packet Format

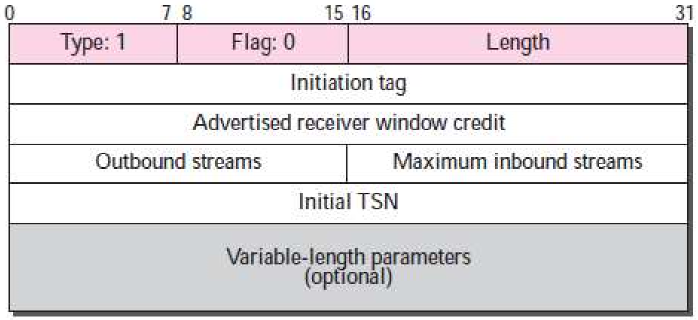


Fig: INIT chunk

➢ ***INIT*** **(Initiation chunk)**

- First chunk sent by an end point to establish an association
- Cannot carry any other control or data chunks
- Verification tag for this packet is 0
- Common fields
  - Type field has a value of 1
  - Flag field is 0
  - Length field value is minimum of 20
- Initiation tag
  - 32-bit field defines the value of the verification tag for packets traveling in the opposite direction
  - Tag is same for all packets traveling in one direction in an association
  - Random number between 0 and $2^{32} - 1$
- Advertised receiver window credit:
  - 32-bit field used in flow control
  - Defines the initial amount of data in bytes that the sender of the INIT chunk can allow
- Outbound stream: 16-bit field defines the number of streams an initiator of the association suggests for streams in outbound direction.
- Maximum inbound stream: 16-bit field defines the maximum number of streams an initiator of the association can support in inbound direction
- Initial TSN: initializes TSN in the outbound direction
- Variable-length parameters: optional parameters to define the IP address of sending end point, multihome, cookie state etc.

SCTP Packet Format

➢ ***INIT ack***(initiation acknowledgment chunk)

- Second chunk sent during association establishment
- Value of the verification tag is the value of the initiation tag of INIT chunk.
- The parameter of type 7 defines the state cookie sent by the sender of this chunk
- Initiation tag field in this chunk initiates the value of the verification tag for future packets traveling from the opposite direction

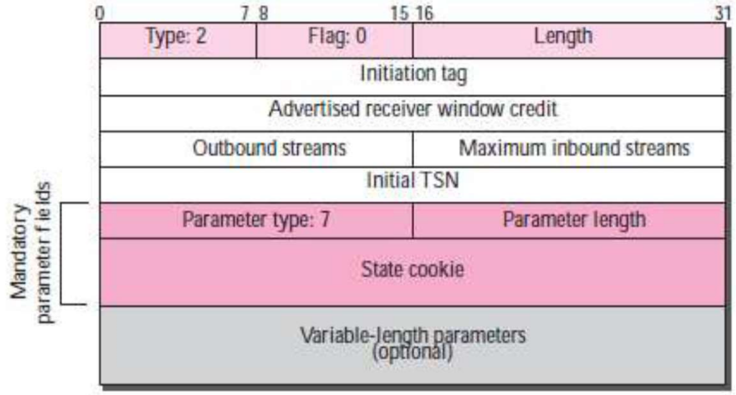


Fig: INIT ACK chunk

SCTP Packet Format

➢ *Cookie echo*

- Third chunk sent during association establishment that carry user data too
- Sent by the end point that receives an INIT ACK chunk
- Chunk of type 10

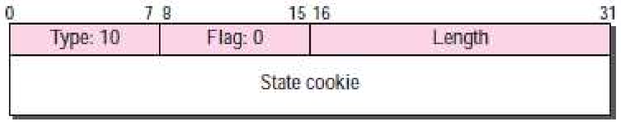


Fig: COOKIE ECHO chunk

➢ *COOKIE ACK*

- fourth and last chunk sent during association establishment with data chunk too
- sent by an end point that receives a COOKIE ECHO chunk
- chunk of type 11

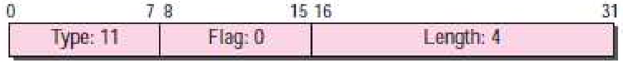
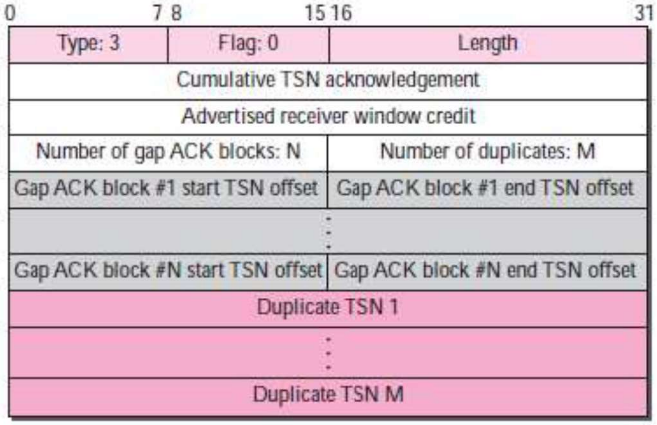


Fig: COOKIE ACK

SCTP Packet Format

➢ ***SACK***(selective ACK chunk)

- Acknowledges the receipt of data packets
- Common fields
  - Type field has 3
  - Flag bits are set to 0s



0
7 8
15 16
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

Fig: SACK chunk

- Cumulative tsn acknowledgment: 32-bit field defines the tsn of the last data chunk received in sequence
- Advertised receiver window credit: 32-bit field that have updated value for the receiver window size
- Number of gap ACK blocks: 16-bit field defines the number of gaps in the data chunk received after the cumulative TSN
- Number of duplicates: 16-bit field defines the number of duplicate chunks following the cumulative TSN
- Gap ACK block start offset: 16-bit field gives the starting TSN relative to the cumulative TSN
- Gap ACK block end offset: 16-bit field gives the ending TSN relative to the cumulative TSN
- Duplicate tsn: 32-bit field gives the tsn of the duplicate chunk.

SCTP Packet Format

➢ *HEARTBEAT and HEARTBEAT ACK*

- First has a type of 4 and the second a type of 5
- Used to periodically probe the condition of an association
- An end point sends a HEARTBEAT chunk, peer responds HEARTBEAT ACK if it is alive
- Parameter fields provide sender-specific information like address and local time
- Same is copied into the HEARTBEAT ACK chunk.

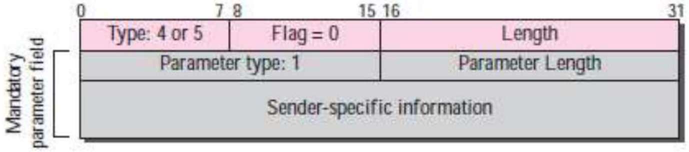


Fig: HEARTBEAT and HEARTBEAT ACK chunks

➢ *SHUTDOWN, SHUTDOWN ACK, and SHUTDOWN COMPLETE*

- Used for closing an association
- Shutdown
    - Type 7 is eight bytes in length
    - Second four bytes define the cumulative TSN
- SHUTDOWN ACK: type 8 is four bytes in length.
- Shutdown complete
    - Type 14 is 4 bytes long
    - T flag is 1 bit flag shows that the sender does not have a TCB table

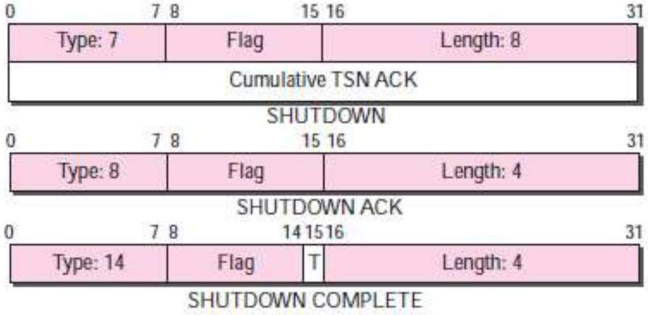


Fig: SHUTDOWN, SHUTDOWN ACK, and SHUTDOWN COMPLETE chunks

SCTP Packet Format

➢ *ERROR*

- Sent when an end point finds some error in a received packet.
- It does not imply the aborting of the association.

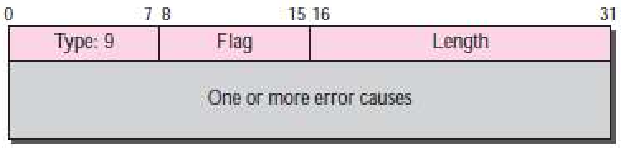


Fig: ERROR chunk

Table: Errors

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

➢ *ABORT*

- Sent when an end point finds a fatal error and needs to abort the association.

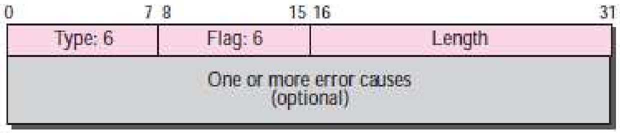


Fig: ABORT chunk

➢ *FORWARD TSN*

- This is a chunk recently added to the standard to inform the receiver to adjust its cumulative TSN

SCTP Client/Server(Association)

➢ **Association Establishment**

- ***Four-way handshake***

  1. First packet has INIT chunk sent by client
     - Verification tag is 0
     - Rwnd is advertised in a SACK chunk
     - Inclusion of a DATA chunk in the third and fourth packets
  2. Second packet has INIT ACK chunk sent by server
     - Verification tag is the initial tag field in the INIT chunk
     - Initiates the tag to be used in the other direction
     - Defines the initial TSN and sets the servers' rwnd
  3. Third packet has COOKIE ECHO chunk sent by client
     - Echoes the cookie sent by the server
     - Data chunks are included in this packet
  4. Fourth packet has COOKIE ACK chunk sent by server
     - Acknowledges the receipt of the COOKIE ECHO chunk
     - Data chunks are included with this packet.

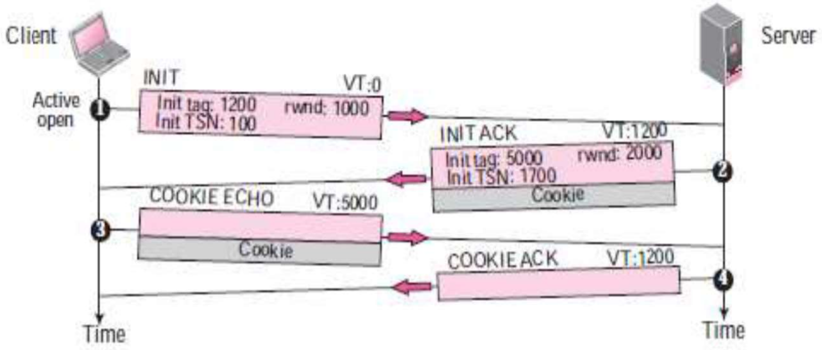


Fig: Four-way handshaking

SCTP Client/Server(Association)

➢ ***Number of Packets Exchanged***

- Number of packets exchanged is four(3 for TCP)
- Allows the exchange of data in the third and fourth packets, so efficient

➢ ***Verification tag***

- It is a common value carried in all packets traveling in one direction in an association
- Blind attacker cannot inject a random packet into an association
- A packet from an old association cannot show up in an incarnation

➢ ***Cookie***

- Cookie is sent with the second packet to the address received in the first packet
- If the sender of the first packet is an attacker, the server never receives the third packet
- If the sender of the first packet is an honest client, it receives the second packet, with the cookie

SCTP Client/Server(Association)

➢ **Data transfer**

- Purpose of an association is to transfer data between two ends
- Once association is established, bidirectional data transfer can take place
- SCTP supports piggybacking
- Each message coming from the process is treated as one unit and inserted into a DATA chunk
- Each DATA chunk formed by a message or a fragment has one TSN and acknowledged by SACK chunks

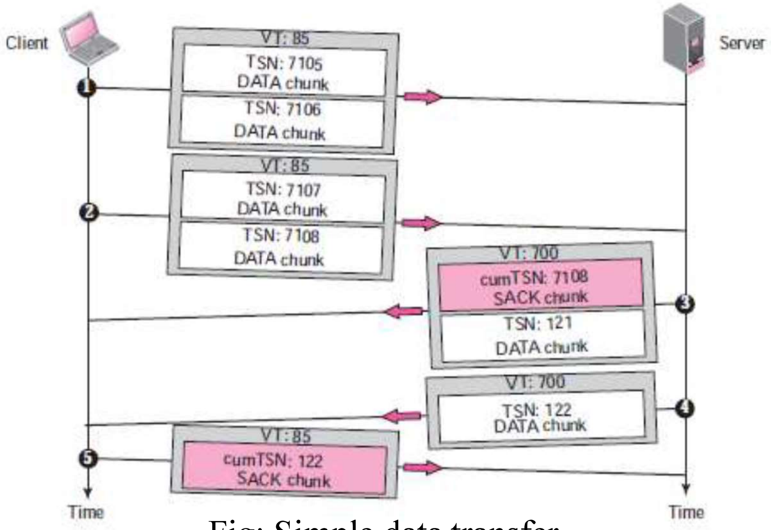


Fig: Simple data transfer

SCTP Client/Server(Association)

➢ ***Multihoming Data Transfer***

- Allows both ends to define multiple IP addresses for communication
- One address is **primary address,** rest are alternative addresses
- The primary address is defined during association establishment
- Primary address of the destination is used by default for data transfer, if it is not available, one of the alternative addresses is used
- SACK is sent to the address from which the corresponding SCTP packet originated

➢ ***Multistream delivery***

- TSN numbers are used to handle data transfer whereas delivery of the data chunks are controlled by SIs and SSNs
- Two types of data delivery in each stream
  - Ordered: SSNs define the order of data chunks in the stream
  - Unordered: U flag is set, it delivers the message carrying the chunk to the destination application without waiting for the other messages

SCTP Client/Server(Association)

➢ ***Fragmentation***

- SCTP preserves the boundaries of the message when creating DATA chunk from a message
- If the total size exceeds the MTU, the message needs to be fragmented
- Steps for fragmentation
  - Message is broken into smaller fragments to meet the size requirement
  - DATA chunk header is added to each fragment that carries a different TSN
  - All header chunks carry the same SI, SSN, payload protocol identifier and U flag
  - B and E are assigned as

    A. First fragment: 10

    B. Middle fragments: 00

    C. Last fragment: 01
- Fragments are reassembled at the destination

SCTP Client/Server(Association)

➢ **Association Termination (Graceful termination)**

- Either client or server involved in exchanging data can close the connection
- SCTP does not allow a "halfclosed" association, i.e. if one end closes the association, the other end must stop sending new data
- If not, the data in the queue are sent and the association is closed
- Association termination uses three packets
    - SHUTDOWN
    - SHUTDOWN ACK
    - SHUTDOWN COMPLETE

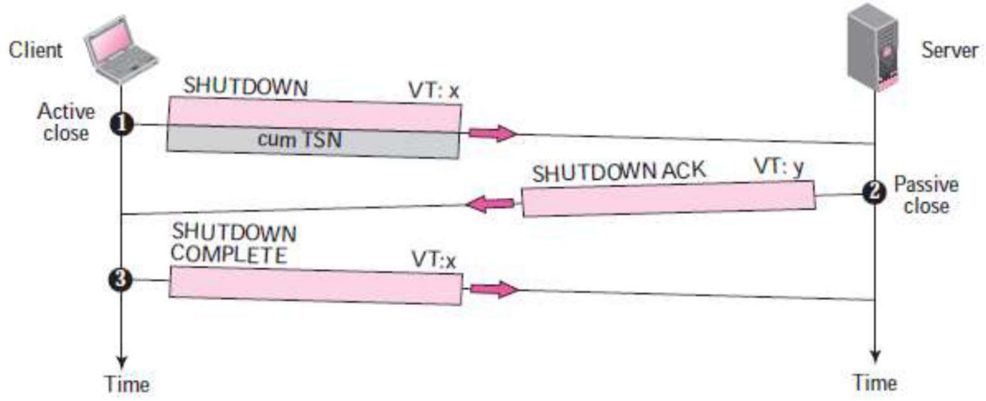


Fig: Association termination

SCTP Client/Server(Association)

➢ **Association abortion**

- Association in SCTP can be aborted based requested by the process at either end or by SCTP
- A process may wish to abort the association if the process receives wrong data from the other end, going into an infinite loop etc.
- Server may wish to abort since it has received an INIT chunk with wrong parameters, requested resources are not available after receiving the cookie, the operating system needs to shut down etc.
- For abortion process either end can send an abort chunk to abort the association

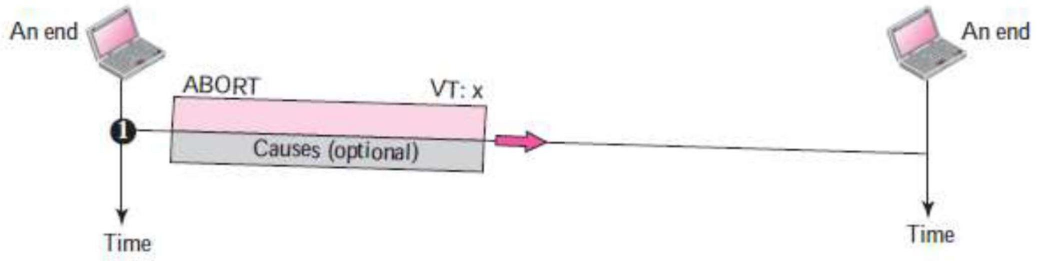


Fig: Association abortion

References

1. Behrouz A. Forouzan, “TCP IP Protocol Suite ” 4th edition, 2010, McGraw-HillISBN: 0073376043 **(Ref 1 in syllabus)**
2. Douglas E. Comer, Internetworking with TCP/IP, Principles, protocols, and architecture,Vol 1 5th Edition,2006 ISBN: 0131876716, ISBN: 978-0131876712 **(Ref 2 in syllabus)**
3. https://www.tutorialspoint.com/remote-procedure-call-rpc
4. https://www.slideshare.net/sunitasahu101/rpc-remote-procedure-call
5. https://aticleworld.com/socket-programming-in-c-using-tcpip