3 way Handshake

① · Establish a connection

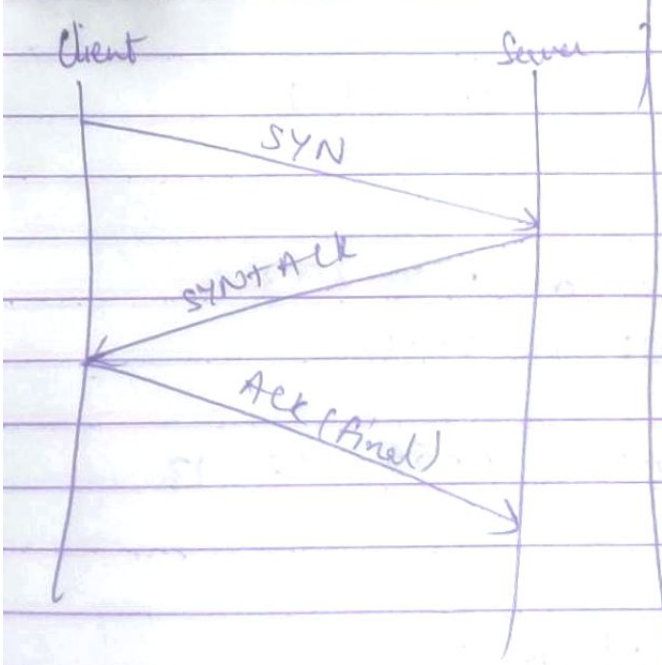
② · 3 steps → ① SYN : Synchronice Sequence No.

(Client sends a request to it sends a segment with SYN which informs the server that the client is likely to start ~~conversatio~~ communication & with what sequence number)

Step 2 → SYN + ACK ⇒ Server responds with SYN ACK signal.

ACK → Signifies response of the segment it received.

Step 3 → ACK : Client receives the acknowledgement & both establish a reliable connection.



4 way Handshake.

① · Terminate

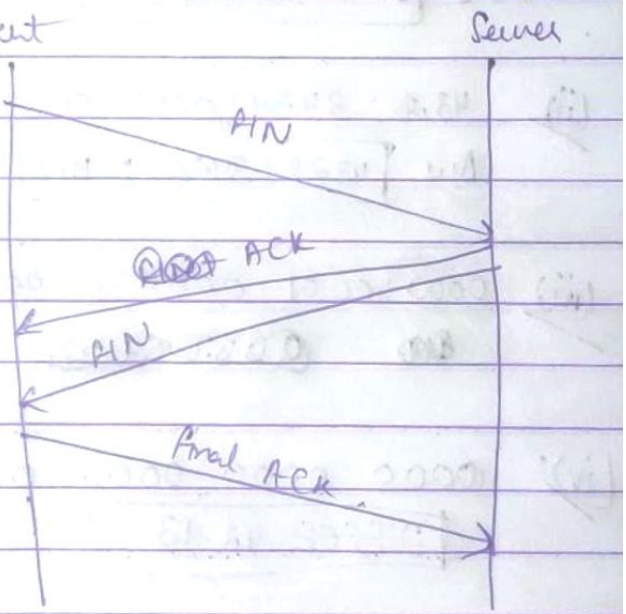
② · 4 steps :

Step ① → Either client or server sends the FIN flag as a request for termination of connection.

Step ② → The receiving side sends an ACK flag as acknowledgement for closing request to other side.

Step ③ → The server will also send a FIN flag as a closing signal to the other side.

Step ④ → Client receives the final FIN flag and sends an ACK flag as final acknowledgement for the suggested connection closing.



→ same func. as that of DHCP.

Limitations

Bootstrap:
1. DHCP N/w servers have much broader use than bootstrap N/w server.
2. It does not provide temporary IP addressing.
3. It only supports manual configuration. Not auto configuration.
4. It is not compatible with DHCP server.
5. Does not support mobile devices.
6. May cause errors because of manual configuration.
7. Lease period of Bootstrap Protocol is 30 days.

Ques: 30: (a): Abbreviating IP address:

1. Remove all the leading zeroes (eg: 000A → A)
2. If there are all zeroes after a number, it can be represented by a single 0. (eg: 0000 → 0)
3. leading zeroes can be replaced by ::

(i) 0000:2213:FFFF:0000:0000:0000:0000:0000

Ans → 0:2213:FFFF::

(ii) 4322:3424:0000:0000:0000:0000:0000:1111

Ans: 4322:3422::1111

(iii) 0000:0001:0000:0000:0000:0000:2000:2002

0:1::2000:2002

(iv) 0000:0000:0000:0000:0000:FFEE:42.13

::FFEE:42.13

(v) (0:0:0:0:0:0:0:1) → ::1 Ans

eg: 0000:0000:FFFF:0000:0000:0000:0000:0000

0:0:FFFF::

We can use :: only once in an IP address.

(b) (i) BCBC:B:BC::4567

BCBC:000B:00BC:0000:0000:0000:0000:4567

(ii) Hexa-decimal to binary: → 8 4 2 1

0 0 0 0 → 0

A → 10
B → 11
C → 12
D → 13

BCBC:000B:00BC:0000:0000:0000:

→ 12 → (8+4+2) → 1100

0000:4567

B → 11 → (8+2+1) → 8 4 2 1

1 0 1 1

0100

(1011 1011:1011 1011:0000 0000 0000 1011:0000 0000 1011 1011:0000 0000
(16×4) zeros: 0100 0101 0110 0111) Ans

16×4 zeros:

(V). zero compressed notation → Continuous zeroes are replaced with ::

leading zeroes can be removed.

Step 1

An SCTP data chunk has the following headers:

Chunk type - 8 bit - First 2 hex digits

Reserved - always 0 - 4 bits - Third hex digit

Flags: I, U, B, E. Each of these takes up 1 bit, 4 bits total - Fourth hex digit If the U flag is set, then the chunk is unordered.

Chunk length - 16 bits, 4 hex digits

TSN - sequence number - 8 hex digits

SI - stream identifier - 4 hex digits

SSN - stream sequence number - 4 hex digits

Payload protocol identifier - 8 hex digits

Data...

The first 2 hex digits are 0, indicating that this is a data chunk.

Step 2

2 of 8

a)

The fourth hex digits is 0, which means the U flag isn't set. Therefore this is an ordered chunk

Step 3

3 of 8

b)

The B flag marks the beginning fragment. The E flag marks the ending fragment. Neither of these flags are set, therefore this is a middle fragment.

Step 4

4 of 8

c)

The second group of 4 digits 0015_{16} identify the chunk length.

Converting the lenght into decimal: $0015_{16} = 21_{10}$.

A chunk length needs to have a size divisible by 4. So, 3 bytes of padding are necessary.

Step 5

5 of 8

d)

TSN is the sequence number of the entire stream, which is found in the second octet 00000005.

Converting the value into decimal: $$00000005_{16} = 5_{10}$.

Step 6

6 of 8

e)

SI is the four digits after the second octet: $0003_{16} = 3_{10}$

Step 7

7 of 8

f)

SSN is the four digits after SI: $000A_{16} = 11_{10}$.