THE UNIVERSITY OF MELBOURNE

Week9 Transport Layer

COMP90007 Internet Technology

Prepared by: Chenyang Lu (Luke)

Your Tutor

Chenyang Lu (Luke)

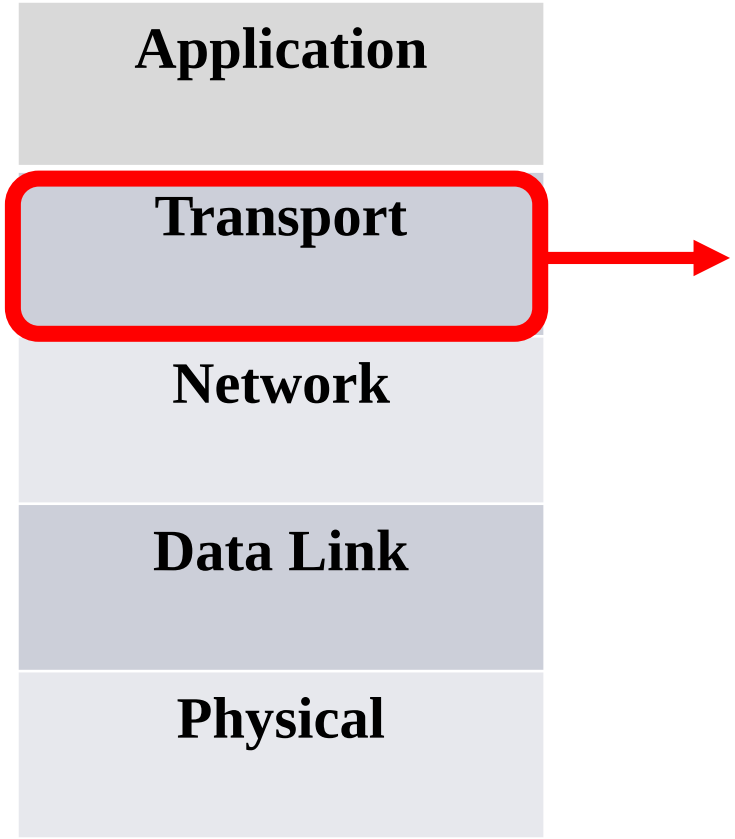
- Email: chenyang.lu@unimelb.edu.au

- Workshop Slides: https://github.com/LuChenyang3842/Internet-technology-teaching-material

Day	Time	Location
Tue	18:15	Bouverie st –B114
Wed	10:00	Elec Engineering -122
Wed	17:15	Bouverie-sr 132

Transport Layer

Transport Layer

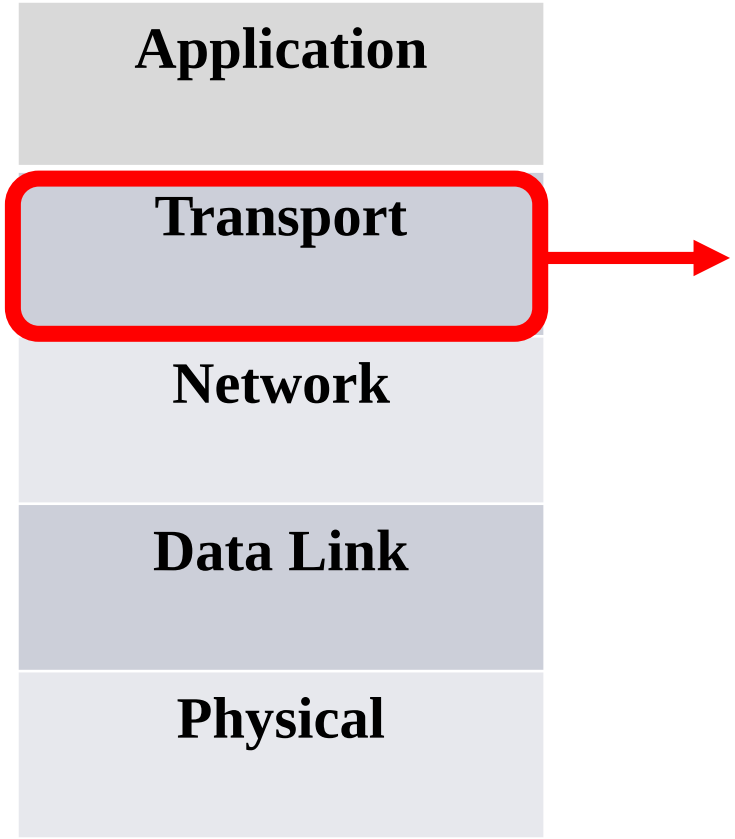


Question:

1. what is the main function of transport layer?

2. What are the two main protocols used in the transport layer?

Transport Layer



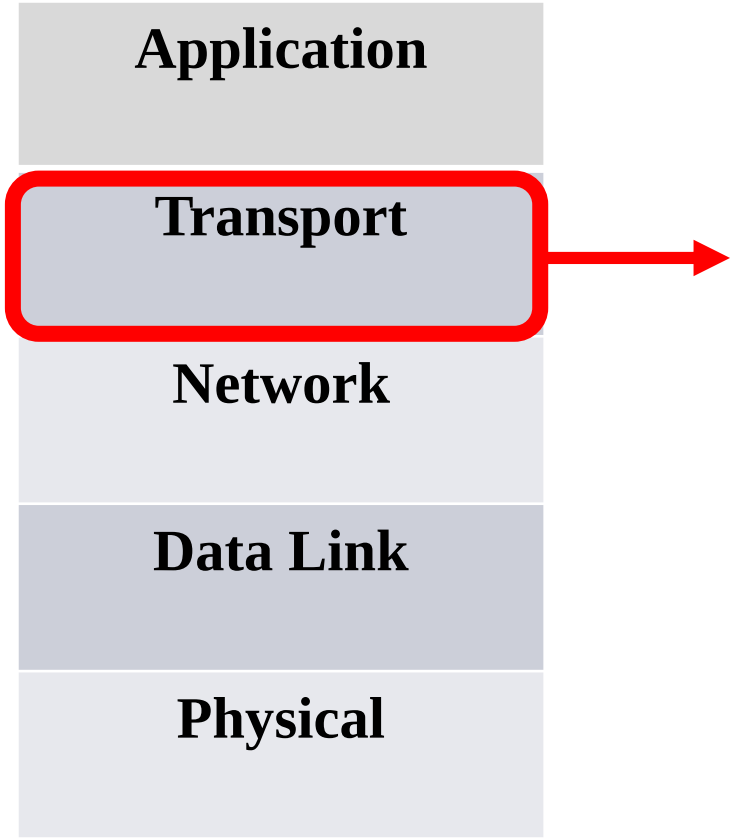
Main function:

- Provide efficient, reliable & cost-effective *data transmission service*
- Independent of physical or data networks

Two Major protocol:

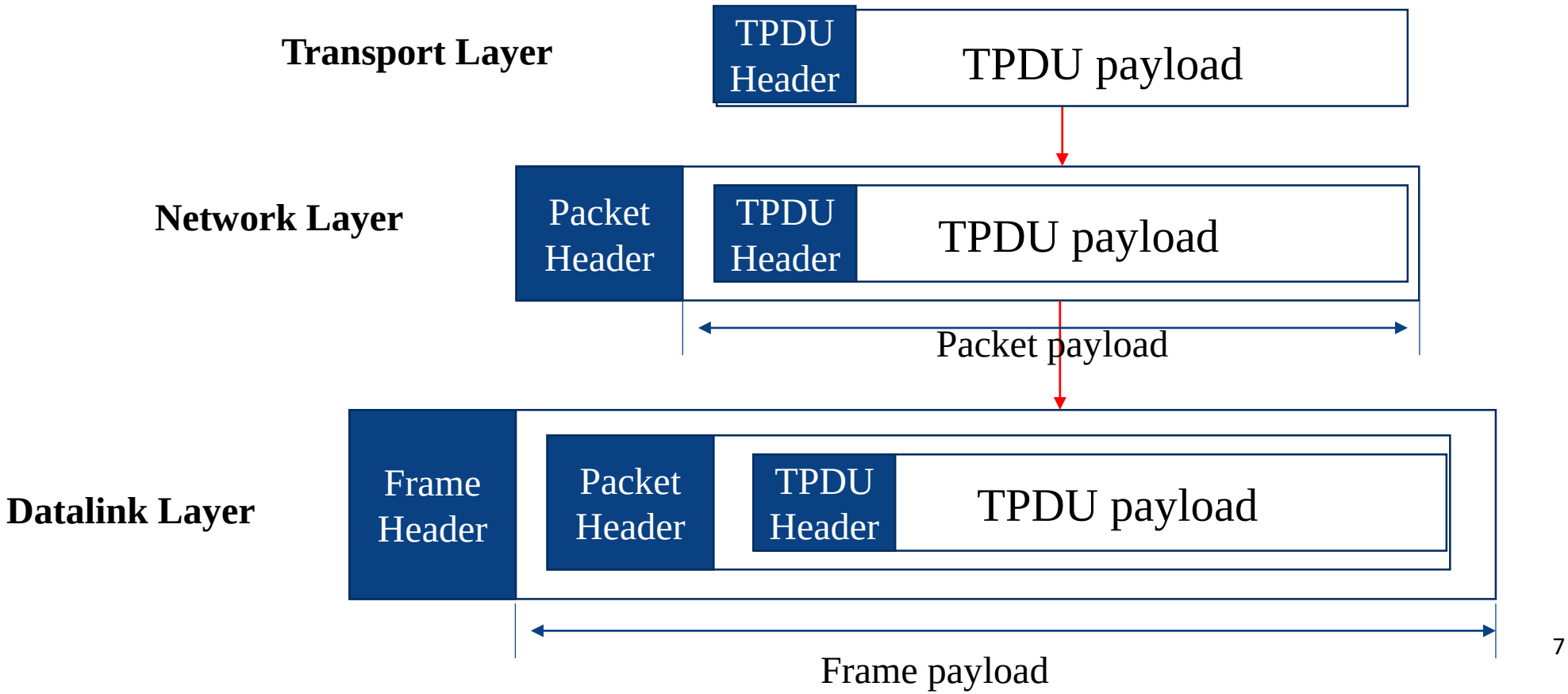
TCP (Connection oriented) and UDP (Connetionless)

Transport Layer

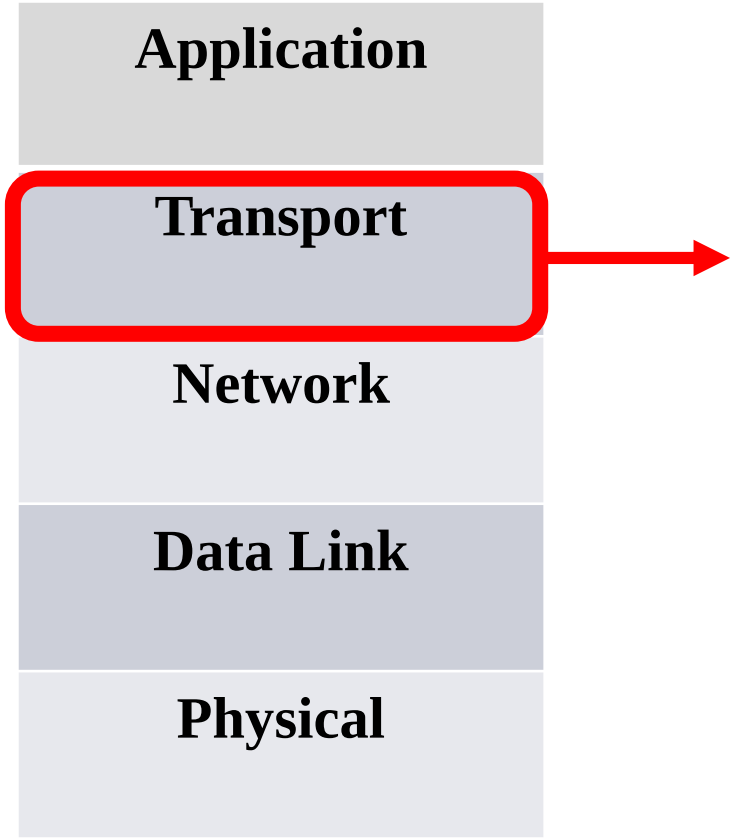


1. Transport Layer Encapsulation
2. Transmission Control Protocol (TCP)
   - Connection-establishment
     - Three-way handshake
   - Connection-release
     - Asymmetric
     - Symmetric
   - TCP segment header
   - TCP based socket
   - Error-control
   - Flow-control
   - Congestion-control
     - ❑ Slow start
     - ❑ Additive increase
3. User Datagram Protocol (UDP)

Encapsulation



Transport Layer

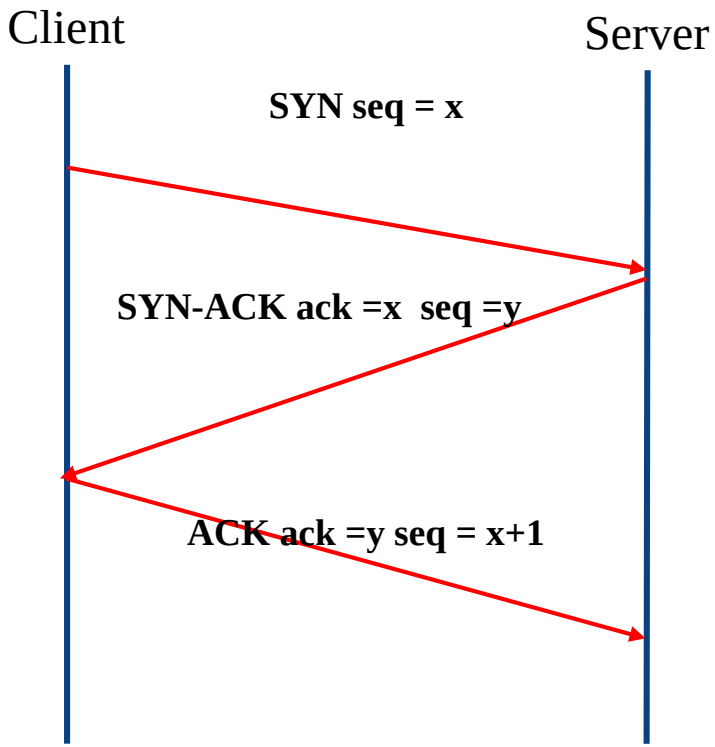


1. Transport Layer Encapsulation
2. Transmission Control Protocol (TCP)
   - Connection-establishment
     - Three-way handshake
   - Connection-release
     - Asymmetric
     - Symmetric
   - TCP segment header
   - TCP based socket
   - Error-control
   - Flow-control
   - Congestion-control
     - ❑ Slow start
     - ❑ Additive increase
3. User Datagram Protocol (UDP)

Connection Establishment

➢ Connection oriented (TCP)

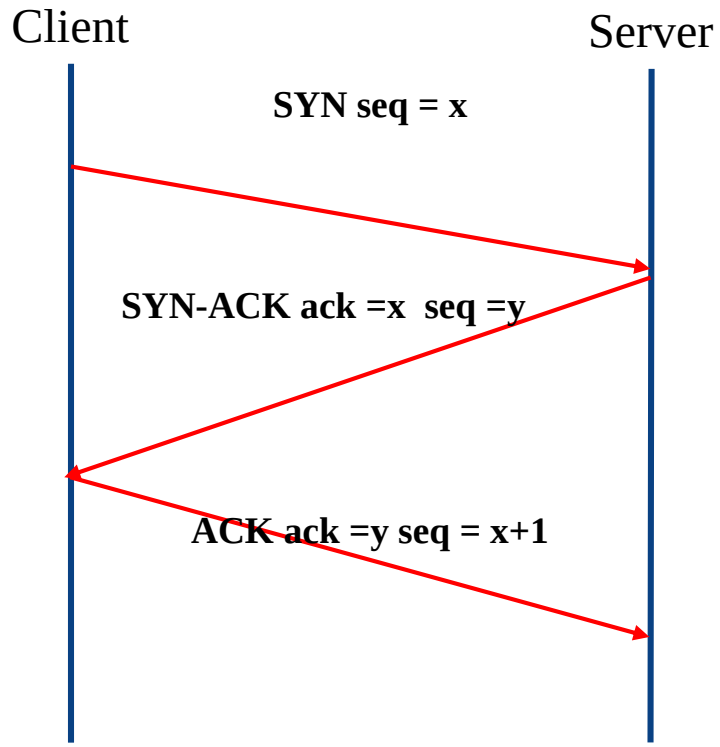
- Start with connection request
- Connection established by **Three-way handshake**



1. Client and server both have a timer, if **time-out** and no ack back for a certain data unit, send again to avoid lost of data unit
2. Use **sequence number** to avoid duplication and data out of order

Looks like easy and clean! But it is not, we need to do more thought experiment to fully understand three-way handshake

Connection Establishment



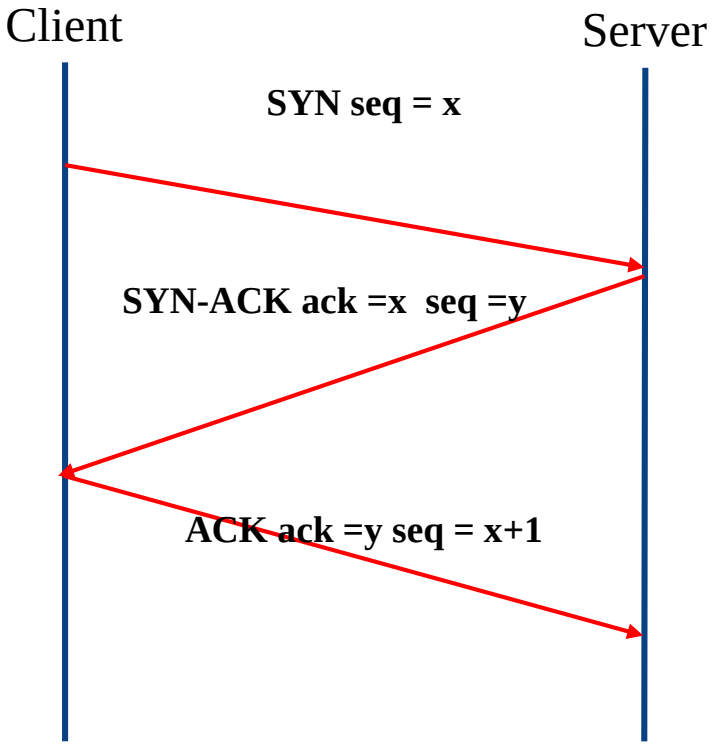
Question:

For the connections between two hosts, Is the initial sequence number always the same?? Say, always start from 0?

NO

- We should either store infinitely increasing sequence number or find another mechanism to refresh the sequence number safely!

Connection Establishment --- (Maximum packet life time and sequence number refresh)



- The hosts **can not infinitely** maintain a continuously growing sequence number,
- we need a mechanism to refresh the sequence number table and discard the old sequence number.
- To achieve this, we set **packet lifetime** to kill off old aged packet. So we can safely refresh the “sequence number table” stored in the hosts after a certain time interval
- We should refresh the "sequence number table" stored in hosts only after the "**maximum packet lifetim**e" that we choose.
- The maximum packet lifetime should be **large enough** to ensure **that not only the packet but also its acknowledgements** have vanished

Refer to textbook 513

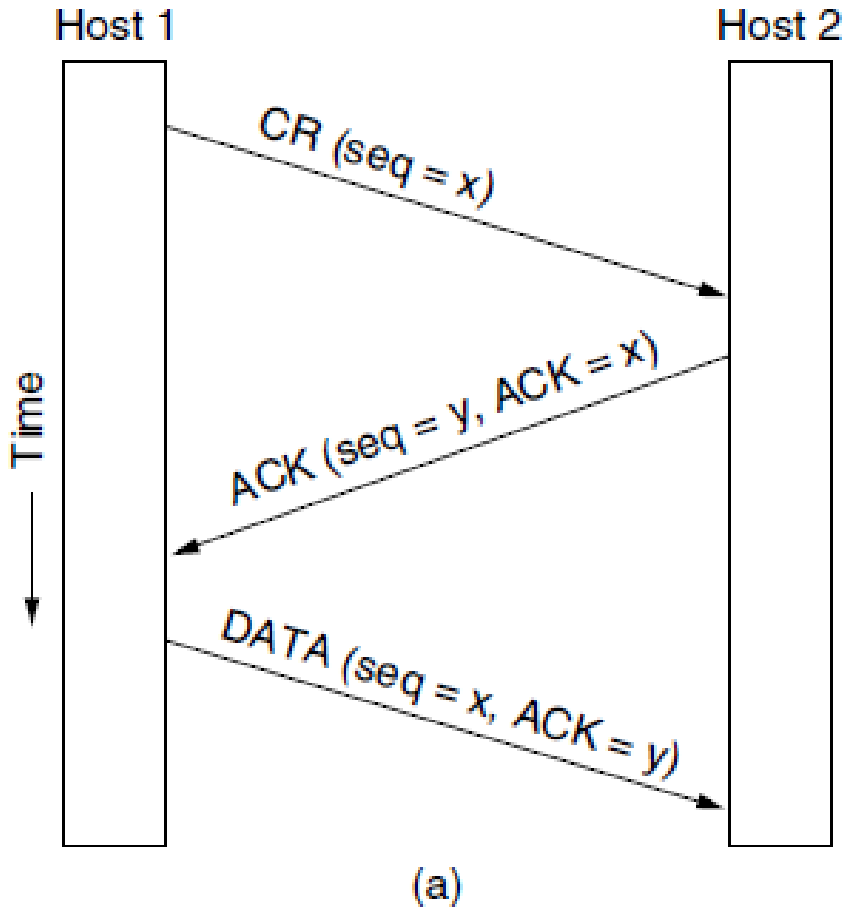
Connection Establishment --- (Maximum packet life time and sequence number refresh)

Packet lifetime: Time to live of packet, information contained in packet

Maximum packet lifetime: The maximum packet lifetime that host believes. Information contained in host.

Three protocol scenarios using Three-way handshake

- **Under Normal Operation,** connection established through 3-way handshake:
  1. Host 1 send Connection request to Host2 with initial sequence number x
  2. Host 2 response with ACK (confirming x), and declare its own initial sequence number y
  3. Last, Host 1 acknowledges host 2's choice of an initial sequence number.

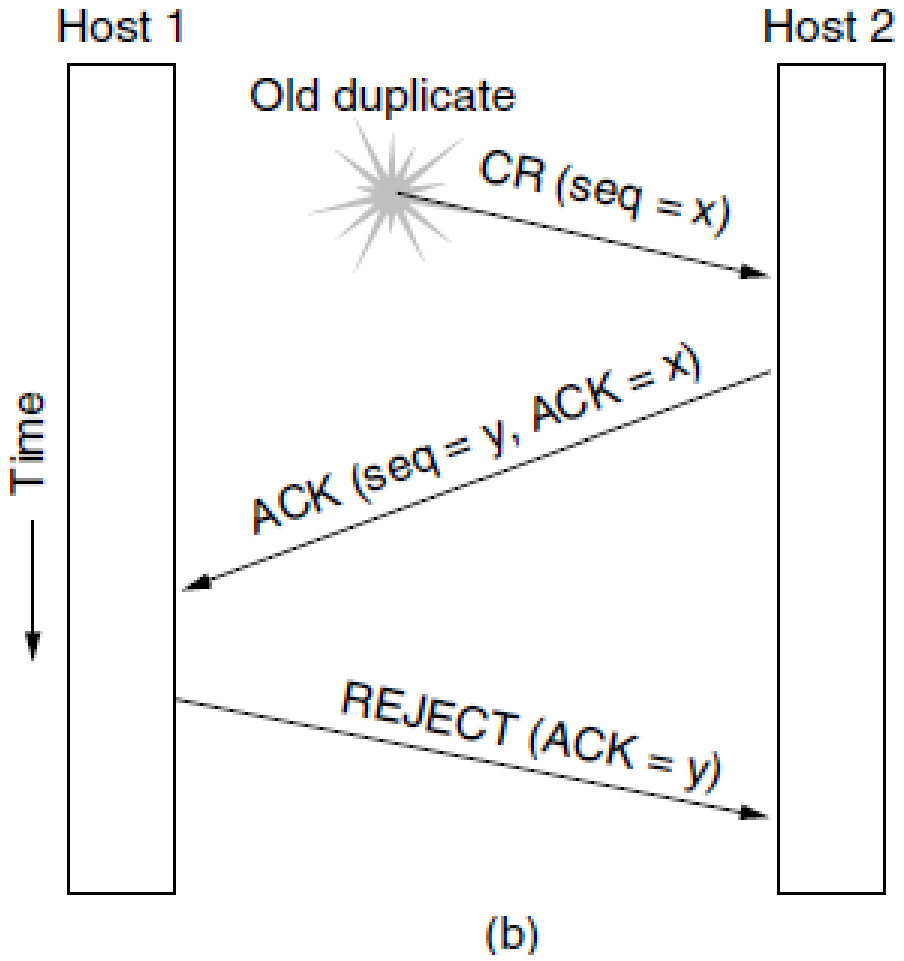


(a)

Three protocol scenarios using Three-way handshake

- **when delay happens:**
  1. The first segment is a delayed duplicate CONNECTION REQUEST from an old connection. This segment arrives at host 2 without host1's knowledge.
  2. Host2 React to this segment by sending host1 an ack segment. Asking for verification that host 1 was indeed trying to set up a new connection.
  3. Host1 reject host 2's attempt to establish a connection.

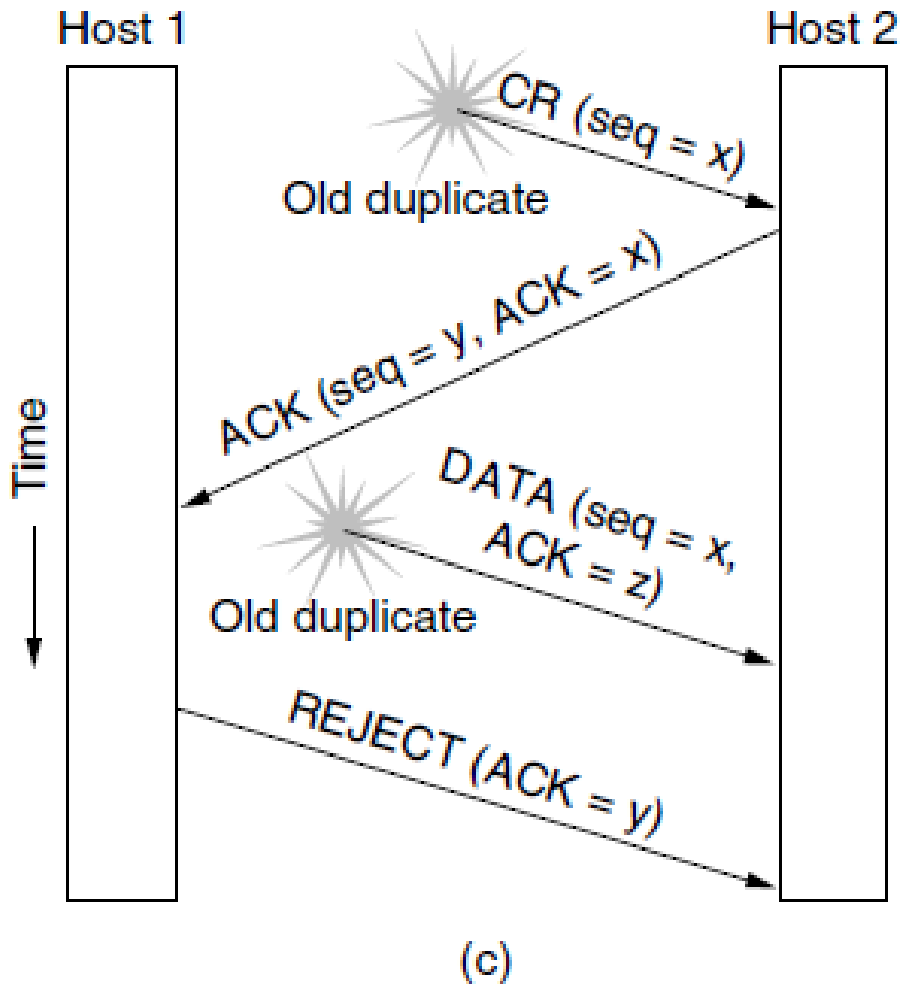
In this way, a delayed duplicate does no damage.



Three protocol scenarios using Three-way handshake

- **The worst case**, Both delayed CONNECTION REQUESTI and an ACK are floating around in the network:

1. As in the previous example, host 2 gets a delayed connection request and replies to it.
2. At this point, it is crucial to realize that host2 has proposed using y as the initial sequence number for host 2 to host1 traffic, knowing full that no segments containing sequence number y or ack to y are still in in existence (Thanks for correctly choosing maximum packet lifetime!)
3. When the second delayed segment arrives at host 2, the fact that z has been acknowledged rather than y, tells host2 that this, too, is an old duplicate!
4. Still, nothing goes wrong!



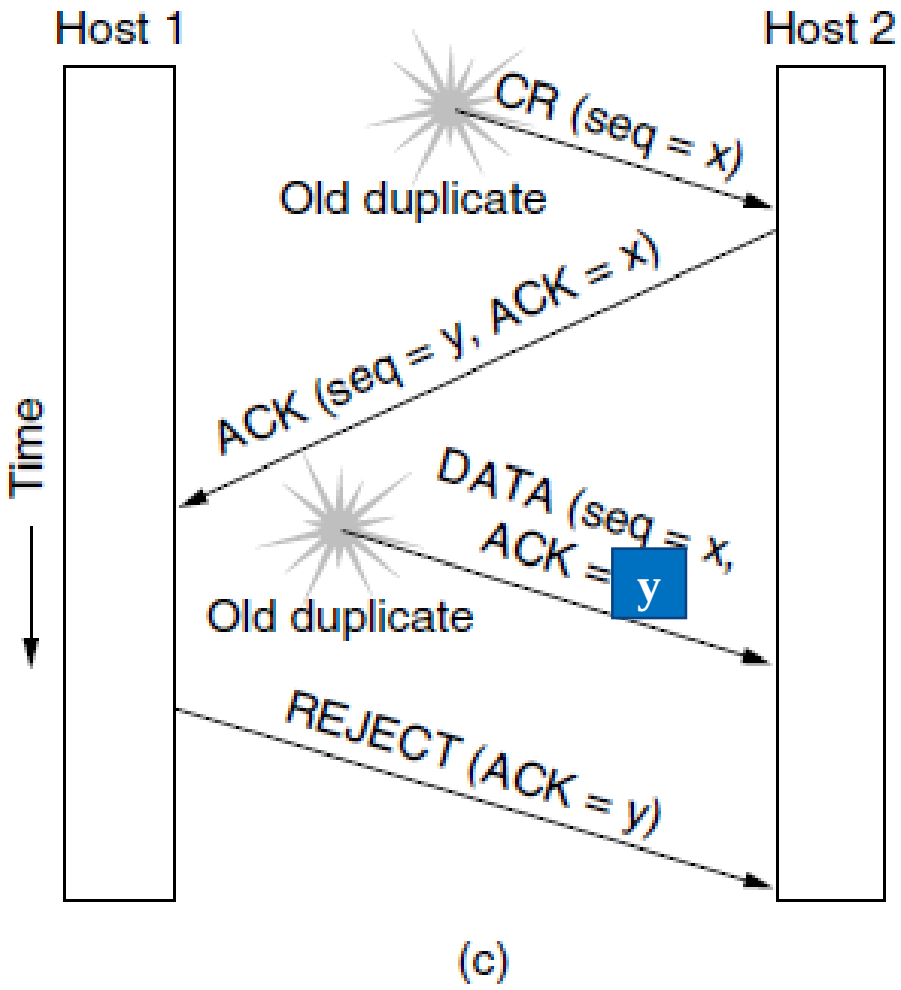
(c)

Question 1 (Connection Establishment)

In determining **maximum packet lifetime**, we have to be careful and pick a **large enough** period to ensure **that not only the packet but also its acknowledgements** have vanished. Discuss why this is needed.

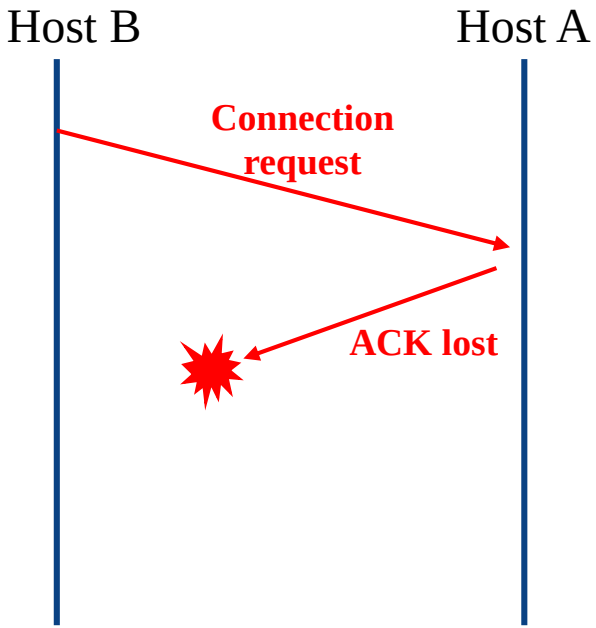
Recall the worst case, Both delayed CONNECTION REQUESTI and ACK are floating around in the network:

1. As in the previous example, host 2 gets a delayed connection request and replies to it.
2. At this point, it is crucial to realize that host2 has proposed using y as the initial sequence number for host 2 to host1 traffic, If we wrongly choose **maximum packet lifetime**, host2 thinks that no segments containing sequence number y or acknowledgements to y are still in existence
3. Unfortunately, due to the wrong choice of maximum packet lifetime, the old duplicate ack to y are actually still floating around and arrive at host2 before REJECT segment sent from host1 arrive host2.
4. Disaster happens!



Question 2 (Connection Establishment)

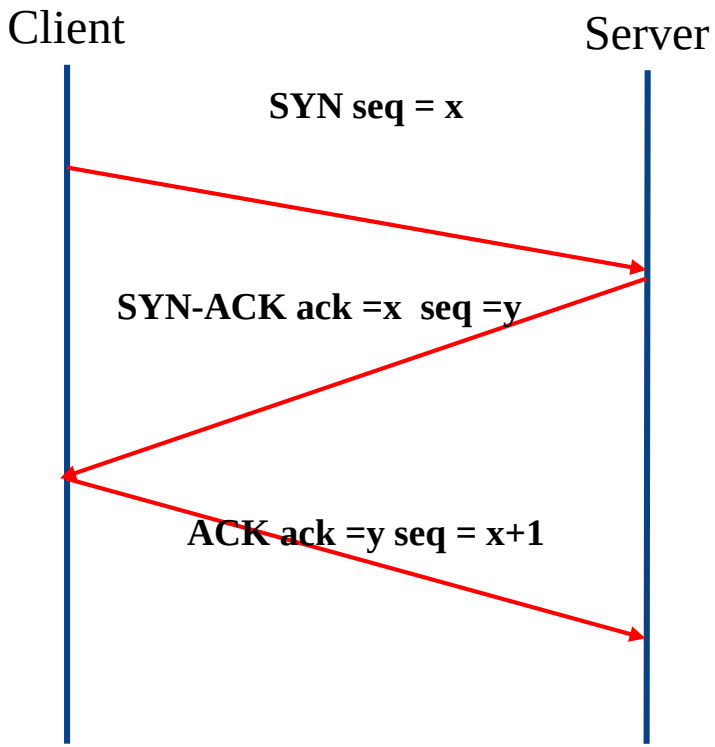
Imagine that a **two-way handshake**, rather than a three-way handshake were used to set up connections. In other words, the third message was not required. Are deadlocks now possible? Give an example or show that none exist.



Deadlock are possible

1. B send CR to A.
2. A send ACK to B. Now, A thinks the connection is established.
3. However, ACK from A to B lost.
4. B is not sure:
   - If A is ready
   - what sequence number will be used during the transmission
   - Not sure if the A have received the connection request
5. Under this situation, B thinks the connection haven't been established and ignore the data transmitted from A but only waits for ACK from A
6. Under this situation, A repeatedly sends same data to B.
7. Dead lock happens
8. **Time out** can resolve this problem

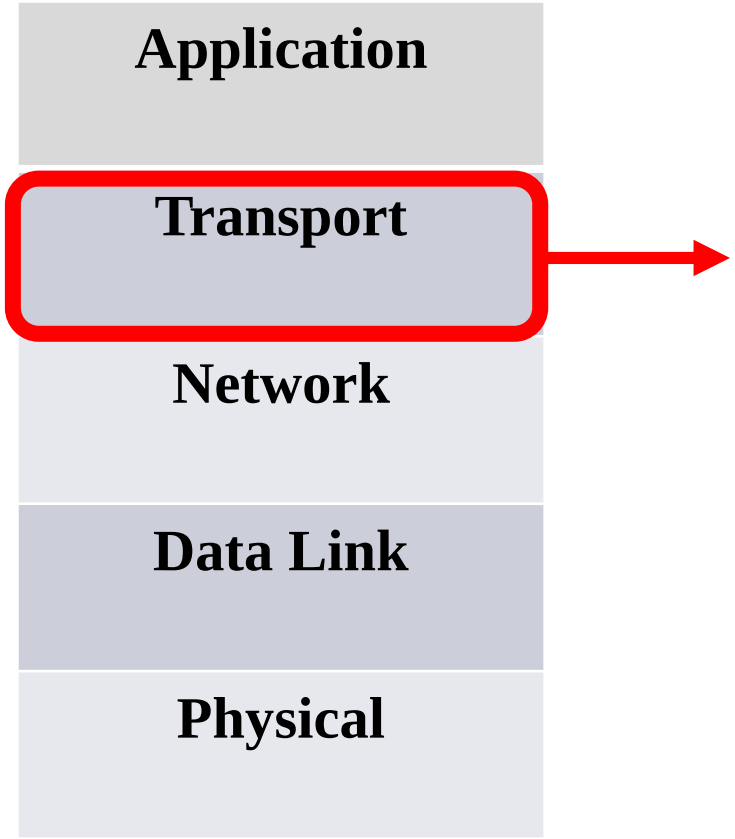
Summary for Connection Establishment



Three-way handshake is easy to understand but not easy to fully understand, ask yourself:

1. Why we need three-way instead of two-way or one-way?
2. Why we need sequence number?
3. Why we need time-out?
4. When should we refresh the sequence number table?
5. How do we choose the maximum packet lifetime?

Transport Layer



1. Transport Layer Encapsulation
2. Transmission Control Protocol (TCP)
   - Connection-establishment
     - Three-way handshake
   - Connection-release
     - Asymmetric
     - Symmetric
   - TCP segment header
   - TCP based socket
   - Error-control
   - Flow-control
   - Congestion-control
     - ❑ Slow start
     - ❑ Additive increase
3. User Datagram Protocol (UDP)

Connection Release

➢ Asymmetric

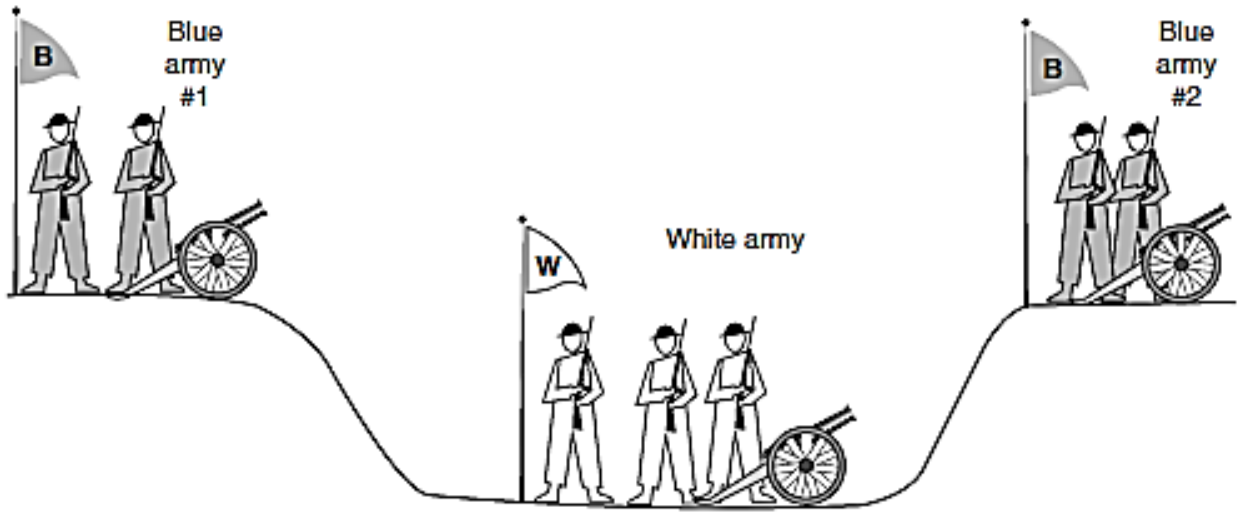
- One side can issue the primitive (**DISCONNECT TPDU**)
- Transmission ends at both side
- Lost data

➢ Symmetric

- Both can issue the primitive (**DISCONNECT TPDU**)
- Transmission only ends at one side
- It is possible for **one side** to **close its TCP connection (it will not send however keep receiving until other end** of the connection **to be closed**...
- The remaining TCP connection is a uni-direction connection

Question 4

What is the 2-army problem? Where does it occur in networking? Provide an example.



Only two blue army attack together, they can win, two blue army can only communicate through unreliable channel. Does there a protocol exist that allows the blue armies win?

A unsolvable problem in computer science!

Example: **Connection release**. To see the relevance of the two-army problem to releasing connections, rather than to military affairs, just substitute "disconnect" for "attack." If neither side is prepared to disconnect until it is convinced that the other side is prepared to disconnect too, the disconnection will never happen

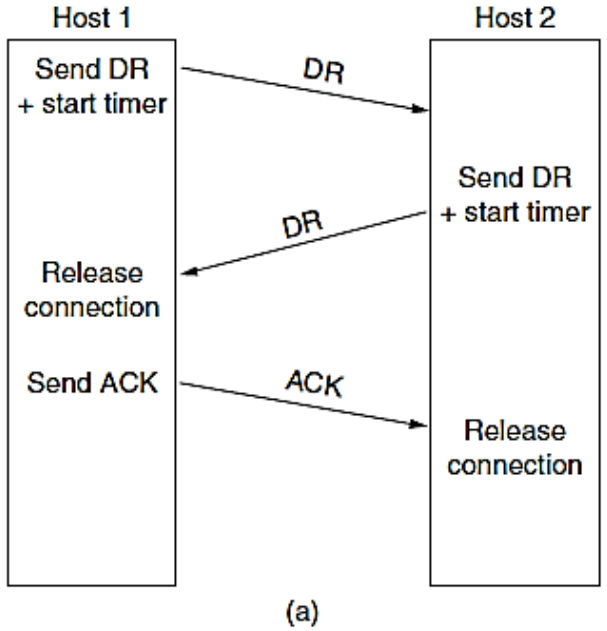
Question 3 (Connection Release)

Does the 3-way handshake-based **connection release** protocol create a flawless disconnection?

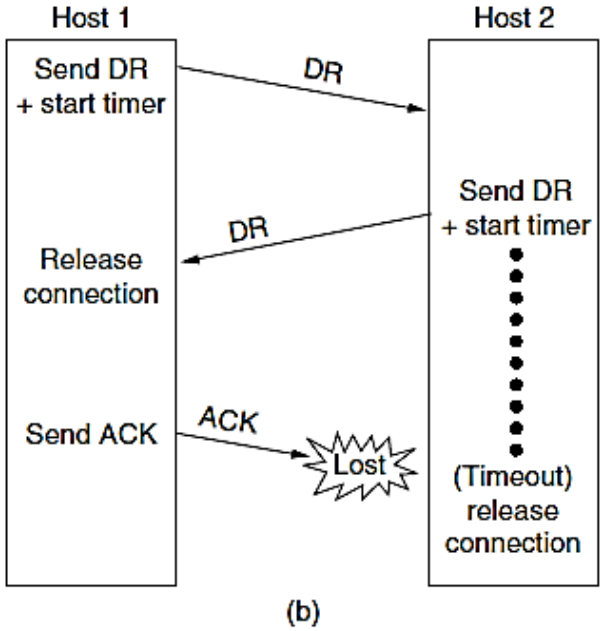
Look at case B, If the timeout triggers while there is
data lingering in the network then the data will be lost
as connection will be terminated early (Force to disconnect). Case D as well, data lost.

They can ensure disconnection but not flawless disconnection.

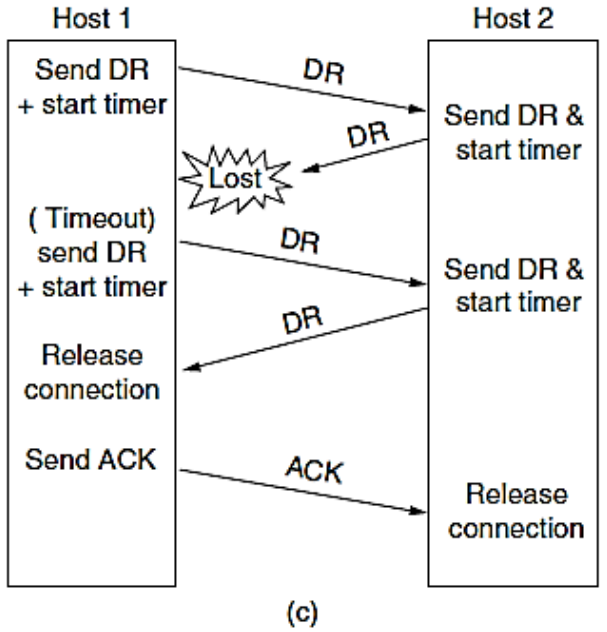
Read through textbook 520



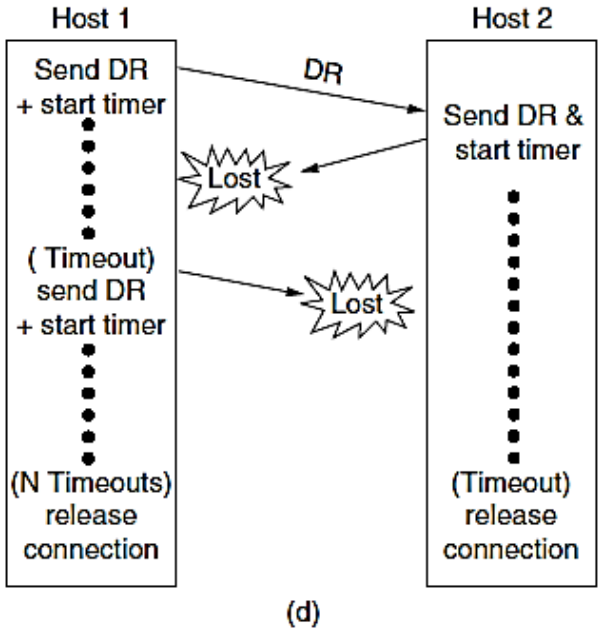
A: Normal case



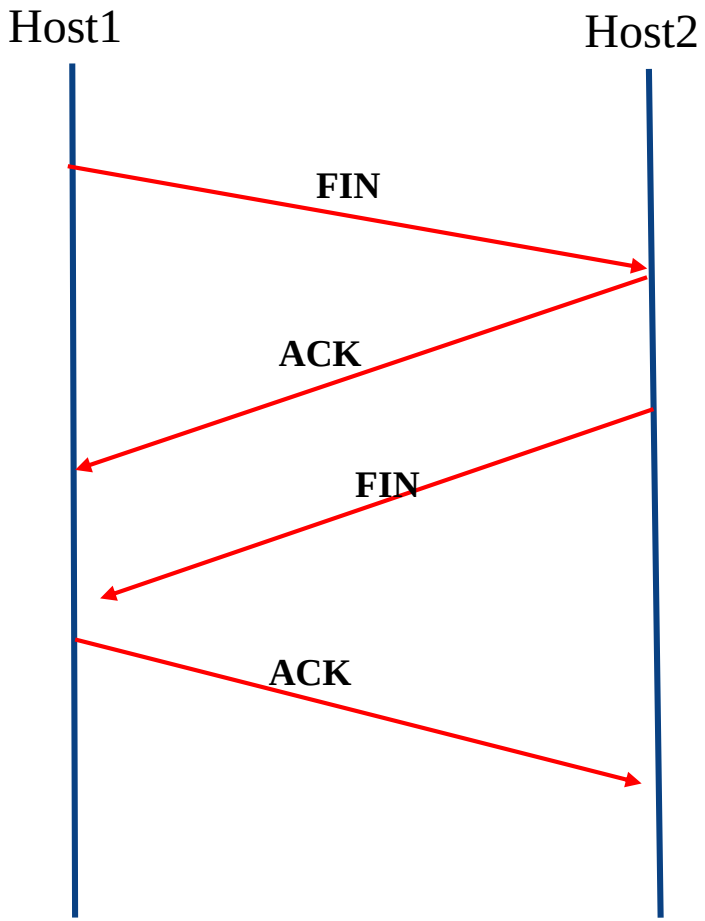
B: Final ACK lost



C: response lost



D: response list and subsequent DRs Lost



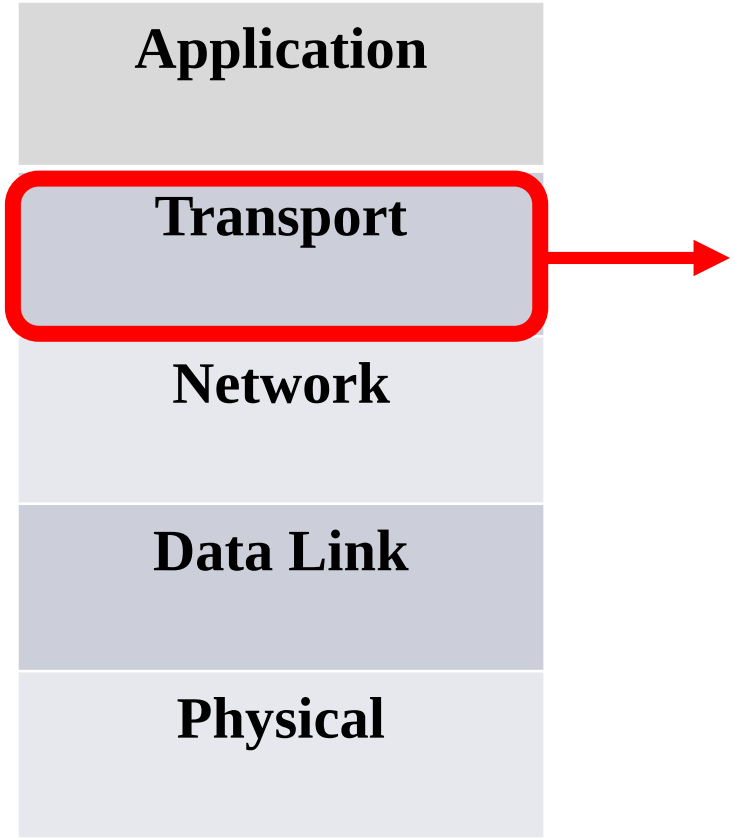
Note: In real life, TCP release use four-way handshake as shown in the left.

The First ACK and second FIN sometimes can be put together and reduce to three-way handshake

Again, it is not a perfect protocol due to two-army problem, but the perfect solution is theoretically impossible, and in practice, the problem rarely arise.

Refer to textbook p562

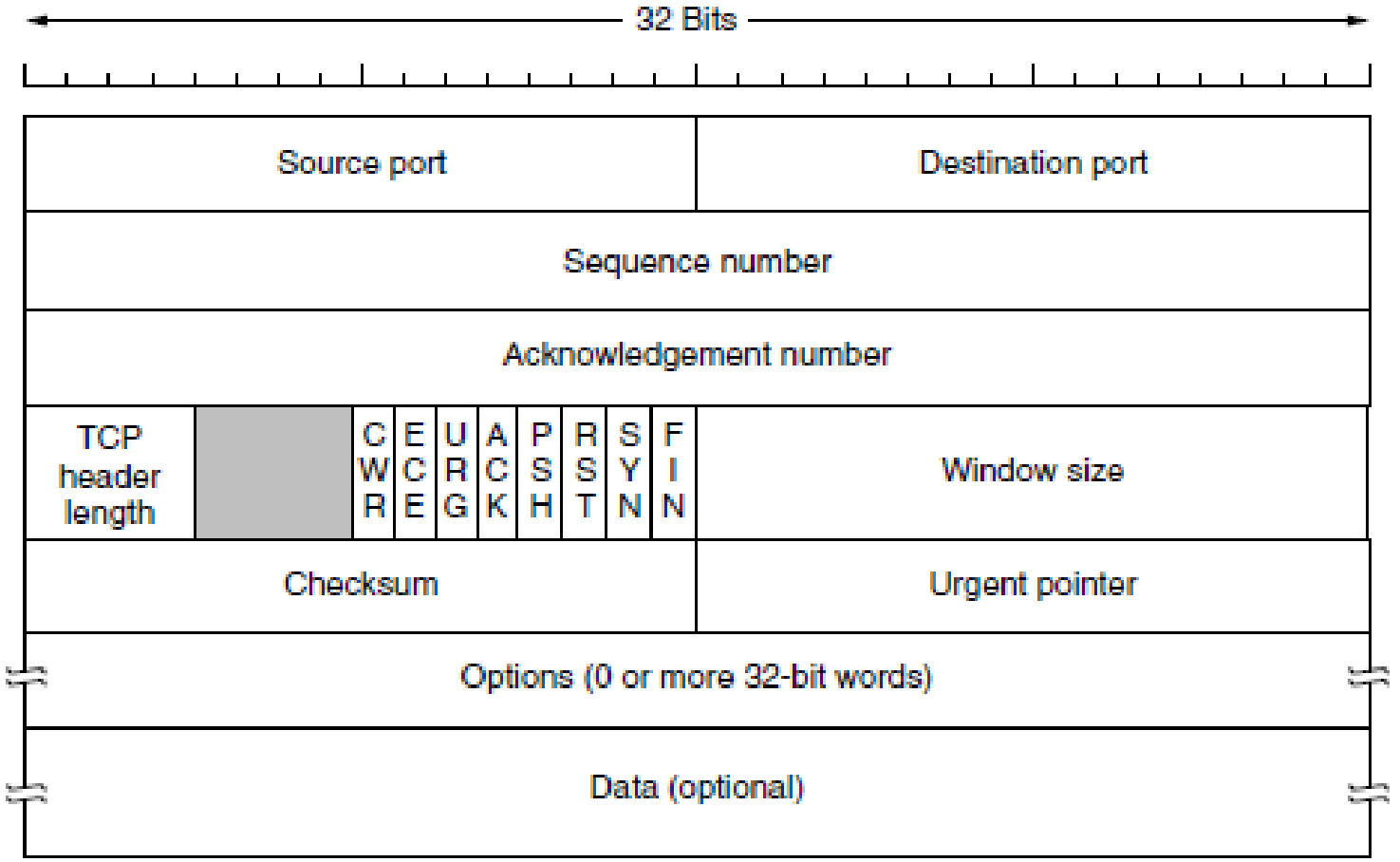
Transport Layer



1. Transport Layer Encapsulation
2. Transmission Control Protocol (TCP)
   - Connection-establishment
     - Three-way handshake
   - Connection-release
     - Asymmetric
     - Symmetric
   - TCP segment header
   - TCP based socket
   - Error-control
   - Flow-control
   - Congestion-control
     - ❑ Slow start
     - ❑ Additive increase
3. User Datagram Protocol (UDP)

Question 5 (TCP segment header)

What information is sent with the TCP Segment header, explain each field briefly?



Socket connection

A socket is an endpoint in a (bidirectional) communication. TCP based socket is widely used in interconnection.

An example of socket programming in java:

```
public class MyServer {


    public static void main(String[] args) {
        try {
            ServerSocket ss = new ServerSocket( port: 6666);
            Socket s = ss.accept();//establishes connection
            DataInputStream dis = new DataInputStream(s.getInputStream());
            String str = (String) dis.readUTF();
            System.out.println("message= " + str);
            ss.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

```
public class MyClient {
    public static void main(String[] args) {
        try{
            Socket s=new Socket( host: "localhost", port: 6666);
            DataOutputStream dout=new DataOutputStream(s.getOutputStream());
            dout.writeUTF( str: "Hello Server");
            dout.flush();
            dout.close();
            s.close();
        }catch(Exception e){System.out.println(e);}
    }
}
```

Question 6

Describe with a simple flowchart how a single socket-based client-server communication works.