ELEC 3500
Tutorial - 5 (Solution)

5-0

$$D_{P2P} \geq \max\left\{ \frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i} \right\} \quad \text{-- (1)}$$

- $\frac{F}{u_s}$: server upload time
- $\frac{F}{d_{min}}$: highest client download time
- $\frac{NF}{u_s + \sum_{i=1}^{N} u_i}$: Peer to Peer share time

Peer to peer architecture is much faster than the centralised client server structure when the total upload capacity of the system is high. The total upload capacity is represented by the denominator of the third term of the above equation.

This happens when the sum of upload rates of individual peer is higher than u_s; the upload rate of the server.

Example! $F = 10\,MB$, $u_s = 20\,Mbps$, $u_1 = u_2 = u_3 = 5\,Mbps$
$N = 3$, $d_{min} = 10\,Mbps$

$$F = 10\,MB = 10 \times 1024 \times 1024 \times 8 = 83886080 \text{ bits}$$

$$\frac{F}{u_s} = \frac{83886080}{20 \times 10^6} = 4.194 \text{ sec.}$$

$$\frac{F}{d_{min}} = \frac{83886080}{10 \times 10^6} = 8.388 \text{ sec}$$

$$\frac{NF}{u_s + \sum u_i} = \frac{3 \times 83886080}{20 \times 10^6 + 15 \times 10^6} = 7.19 \text{ sec.}$$

The minimum distribution time for the client server applicatio

$$D_{cs} \geq \max\left\{ \frac{NF}{u_s}, \frac{F}{d_{min}} \right\} = \max\{3 \times 4.194, 8.38\} \text{ sec}$$

$$= \max\{12.582, 8.38\} \text{ sec} = 12.582 \text{ sec.}$$

Using eq (1) $D_{P2P} \geq \max\{4.194, 8.388, 7.19\}$ sec $= 8.38$ sec.

Hence, $D_{cs} > D_{P2P}$; client server approach takes longer time to deliver the same content.

Now assume $u_1 = u_2 = u_3 = 20$ Mbps. $d_{min} = 50$ Mbps

All other values are same

$$\frac{NF}{u_s + \Sigma u_i} = \frac{3 \times (10 \times 1024 \times 1024 \times 8)}{20 \times 10^6 + 60 \times 10^6}$$

$$= 3.145 \text{ sec.}$$

$$\frac{F}{d_{min}} = \frac{10 \times 1024 \times 1024 \times 8}{50 \times 10^6} = 1.677 \text{ sec}$$

$D_{cs} \geq \max\{12.582, 1.677\}$ sec $= 12.582$ sec.

$D_{P2P} \geq \max\{4.194, 1.677, 3.145\}$ sec $= 4.194$ sec

4.194 ← bottleneck client server

3.145 ← P2P distribution is much faster.

Above calculation assumes that server first download the message then peers distribute the message to others.

Here also, $D_{cs} > D_{P2P}$, in this case P2P delay reduces due to higher speed.

5-1. (a) Since the audio and video contents are combined in the same file the N versions of the content can be delivered from N files.

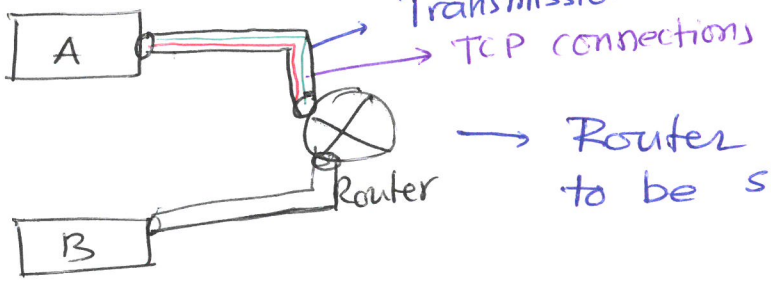
This is a scenario we see in case of Netflix and other content delivery systems. We assume that each versions of the content is coded at a different rate hence, the server will match the delivery content with the available capacity of a network.

(b) If streams are separately stored the we need 2N files.

5-2 Yes, it is possible to open simultaneous multiple TCP connections to a Website. Each connection will use a separate port which will allow parallel connections to download a ~~faster~~ file faster.

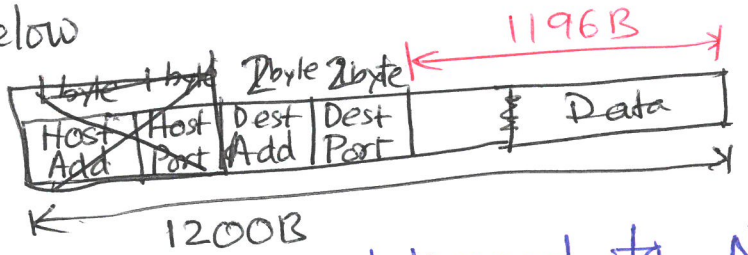
A TCP connection is mapped on a physical connection offered by the physical layer. If too many TCP connections are opened to download a file then it will consume higher amount of physical transmission resources which may cause capacity constraint for other users who are sharing the same physical ~~link~~ path.

Example:



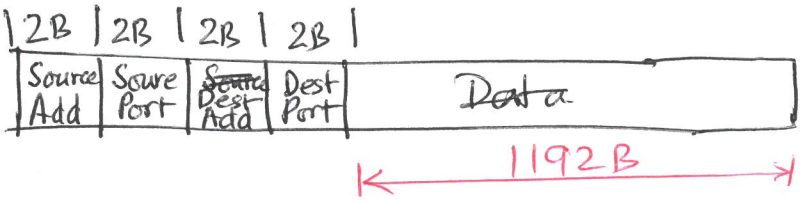
5-3

@ Let's call this protocol as a Simple Transport Protocol (STP). At the sender end, the STP protocol accept a chunk of data from its application. It accepts a maximum data chunk size of 1196 bytes, a destination host address, and destination port no. STP adds a 4 byte header as below



Above block is delivered to Network Layer which is transmitted and received by peer layers. Content is delivered to the peer application by the port number.

(b)



(4)

(c) No the transport layer don't involve ~~the~~ any issues with the transit network functions whether it is core or access network. It is an end-to-end protocol.

5-4 An application developer may not want to use the TCP congestion control technique which throttle the applications sending rate at times of congestion. Also, the TCP header is larger than UDP header. IP telephony, VoIP, videoconferencing ~~app~~ avoid TCP ~~application~~ protocol mainly due to the congestion processes Also, the real time services such as voice and video can afford some losses, so reliability ~~requir~~ requirements from the content delivery point is less restrictive. However, they require lower delay and higher throughput.

5-5 Some voice and video may use ~~combinati~~ TCP protocol because the firewall often blocks UDP traffic. However, in that case the TCP is reconfigured to minimise the effect of congestion control techniques.

5-6 Yes it is possible to include reliability algorithm in the application Layer and use ~~such~~ UDP protocol. However, significant efforts are required at the application Layer.

5-7 Yes, both segments will be directed to the same socket. For each received segment, at the socket interface, the operating system will provide the process with the IP address to determine the origins of individual segments.

5-8 For each

(a) 20 bytes (90 – 109)

(b) Ack number = 90 indicating that it is still waiting for byte with Seq. no. 90.

5-9 Checksum generation: Add all data bytes and then generate 1's complement data.

```
Byte 1 :   0101 0011
Byte 2 : + 0110 0110
         -------------
           1011 1001
Byte 3 :   0111 0100
         -------------
         1 0010 1101
```

Carry bit. + → 1

Fold the carry bit

```
           0010 1110
```

Take 1's complement : 1101 0001

When the data bytes and checksums are received, the data bytes are added and then added with the Checksum data. In case of no error the result should be all all 1.

Addition value of 3 bytes : 0010 1110
1's Complement : 1101 0001
→ 1111 1111 → No Error condition.

Assume that last bit of 3rd byte is corrupted, hence the received byte is 0111 0101

At receiver all 3 bytes are added, ~~and~~ the result is added with the checksum.

Byte 1 + Byte 2 : 1011 1001
Received byte 3 : 0111 0101

1 0010 1110
(1 → 1)
0010 1111

Checksum byte: 1101 0001

1 0000 0000 → indicates error in the data.

1's complement is used because if the sum contains a zero, the receiver knows ~~there~~ there is an error or errors.

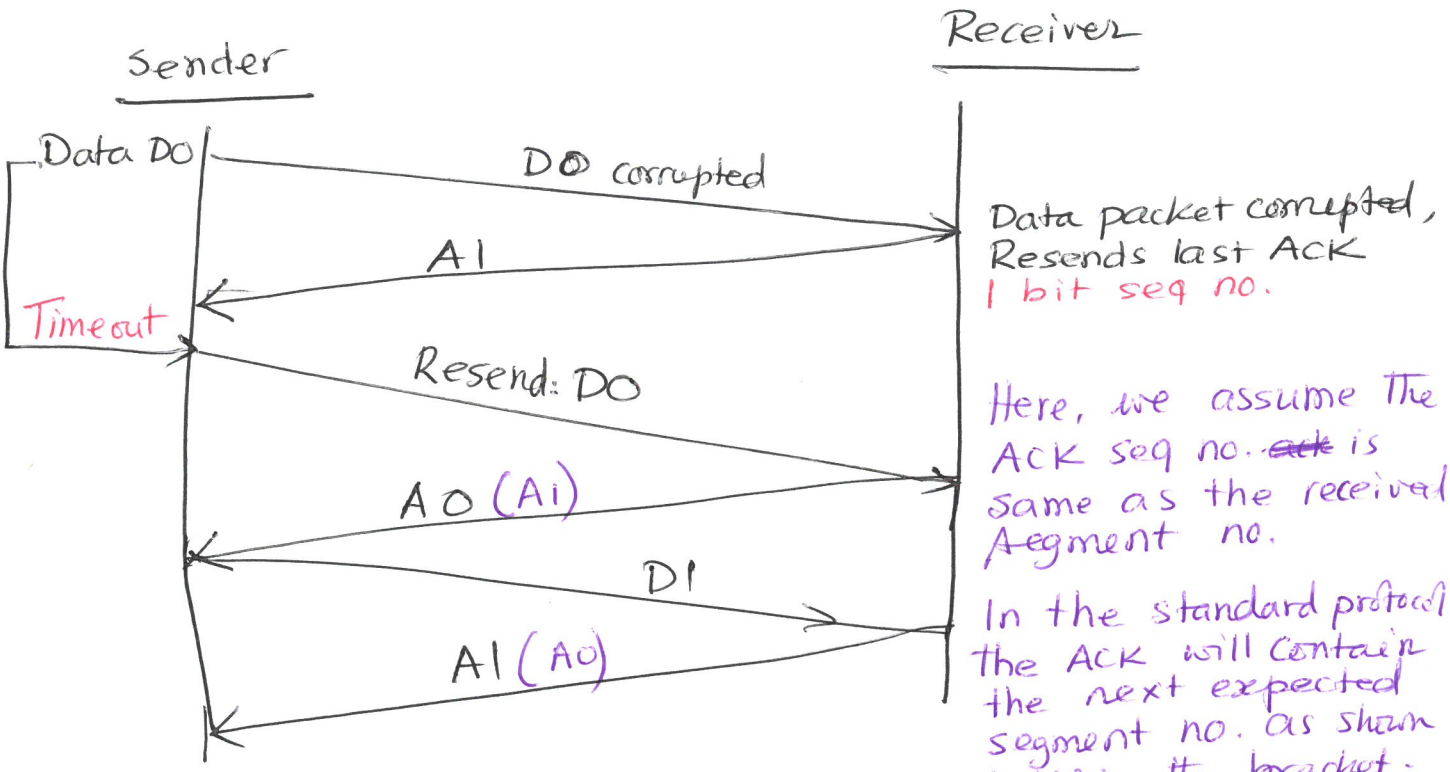
This technique will detect all one-bit errors. Two bit errors can be undetected.

5-10

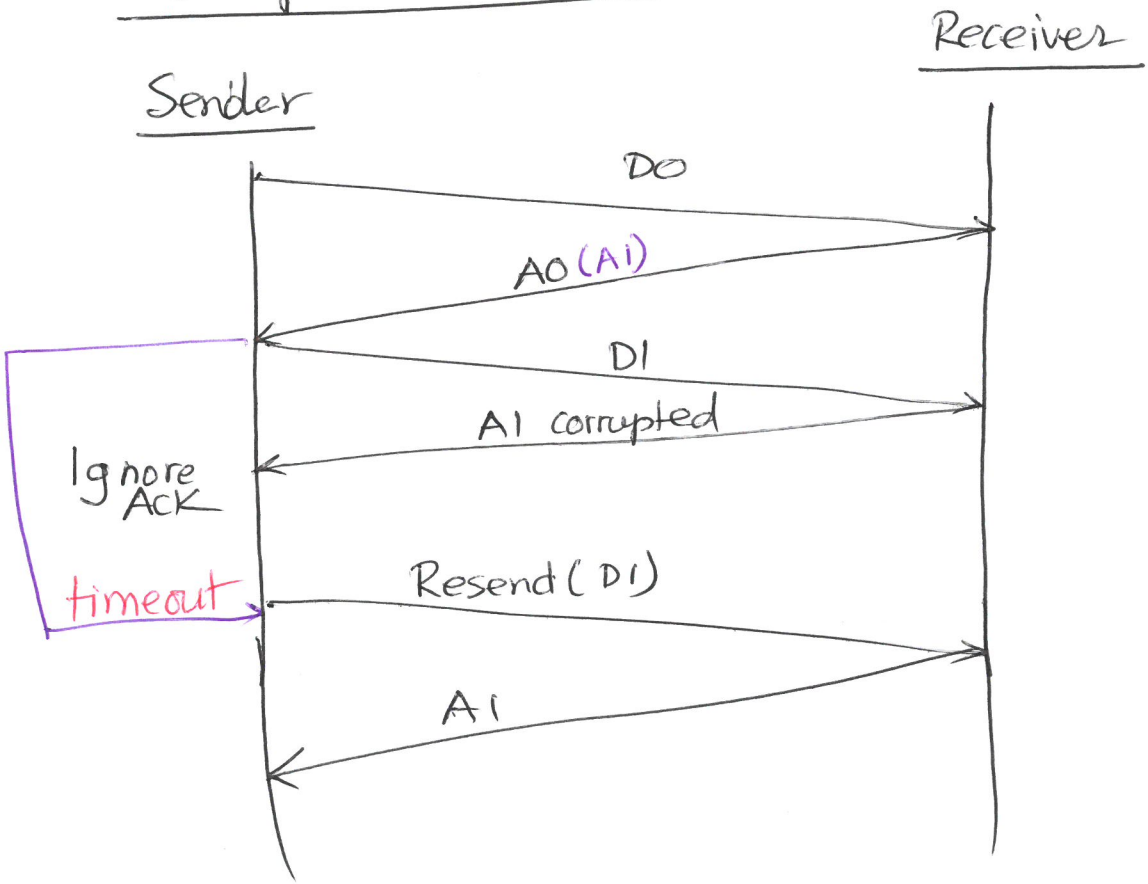
Checksum : 0101 1101 1111 0010
sum of data : 0011 0010 0000 1101
+
1000 1111 1111 1111

The receiver consider the received Segment has error/errors. The UDP protocol will inform the application layer.

5-11 Suppose that the protocol is in a steady state, operating for sometime. The sender (TCP) is in wait state, waiting for data from the application layer. The data corruption scenario is shown below.



Corrupted ACK scenario



5-12 In this case a timer is added whose value is greater than the known the round-trip delay. Timeout is added to wait for ACK/NAK. If a timeout occurs, the most recently transmitted packet is retransmitted. The rtd 2.1 receiver works in following way:

* Suppose the timeout is triggered by a lost data packet. From the receiver's point of view if the retransmitted data packet is received due to the timeout then the receiver will accept that data as a new data because it hasn't seen the data before.
* Suppose that an ACK is lost. The ~~receiver~~ sender will eventually ~~treat~~ retransmit due to timeout.

—x—