COMP3331/93331: Solutions to Mid-term Exam

Question 1 Hodge-Podge (5 marks: 1 mark for each question)

(a) Since the clients are using HTTP1.1 (persistence with pipelining), all objects will be requested and transmitted over a single TCP connection between each client and the server. Thus, each client will have one TCP socket open.

The server will service all 10 clients simultaneously. Thus, the server will have 10 sockets open, 1 socket to communicate with each client. In addition, the server will also have a welcome socket open. Thus, in total, there will be 11 sockets open at the server.

(b) Since the clients are using HTTP1.0 (non-persistent HTTP) with parallel connections, each object will be requested and transmitted over a distinct TCP connection between each client and the server. Thus, each client will have five TCP sockets open.

The server will service all 10 clients simultaneously. Thus, the server will now have 50 sockets open (10 clients x 5 sockets each). In addition, the server will also have a welcome socket open. Thus, in total, there will be 51 sockets open at the server.

(c) A selfish peer could download only from the seed (which has the entire file and thus does not need any chunks) or through the "optimistic unchoking" process when other peers give this peer a chance to download some chunks from them.

(d) DNS and first-person shooter games typically use UDP.

(e) Since TCP uses cumulative acks, the receipt of ACK number 8000 guarantees that everything up to byte # 7999 has been correctly received by the receiver. Thus, the sender can be sure that the first 3 packets (#5000, 6000 and 7000) have been correctly received.

NOTE: In practice, the sequence #s would be 5001, 6001 and 7001 since the SYN packet uses up 1 sequence number. However, we will ignore this here.

Question 2 – The Poisoning of Google (4 marks)

(a) Alice would configure the following entries in her authoritative name server (i.e. in 9.9.9.10)

www.searchzilla.com 9.9.9.9 A long-TTL
www.searchzilla.com ns1.google.com NS long-TTL
ns1.google.com 9.9.9.10 A long-TTL
www.google.com 9.9.9.9 A long-TTL

The first record is the standard A record for searchzilla. This is not specific to the attack.
The second record is a malicious entry, which suggests that the name server for searchzilla is ns1.google.com, which happens to be name server for google.com.
The third record is the hostname to IP address mapping for the google name server. However, note that this is a malicious entry as it incorrectly maps the google name server to the authoritative nameserver for searchzilla (9.9.9.10).
The last record maliciously maps the google.com hostname to the searchzilla web server.

Alice would encourage users on the Internet to visit her website, www.searchzilla.com

When Alice's name server receives a type A DNS query for www.searchzilla.com, it will return the 2nd and 3rd record (from above) in the reply. The 2nd record would be included in the authority section of the reply and the 3rd record will be included in the additional section.

If DNS servers in the Internet blindly cache these records, then future queries for www.google.com will be directed to Alice's authoritative name server, 9.9.9.10 (as a result of the 3rd record). Alice's name server will reply back with the 4th record to such queries. As a result, the user querying for www.google.com will instead be presented with the searchzilla home page. To be precise, the GET request for the google home page will be sent to the searchzilla web server, which will reply back with the searchzilla home page.

(b)

A robust DNS server implementation should be less trustful of results returned by other DNS servers and only cache information that's directly relevant to the queried domain and which is coming from the authoritative name servers. In the above example, since google.com is not a subdomain of searchzilla.com, a correct DNS server implementation should ignore all information related to google.com in the results.

Another option is to use something like DNSSec, which allows the query response to be authenticated.

Question 3 - Oh why so selfish? (3 marks)

In order for all the peers to receive the file, the file must have been uploaded N times by the server and the non-selfish peers. The total number of bits that are uploaded is NF. The total bandwidth for uploading is $u_s + \sum_{i=k+1}^{N} u_i$ because only the server and the non-selfish peer perform upload. A lower bound for the download time is: $\frac{NF}{u_s+\sum_{i=k+1}^{N} u_i}$.

Therefore, we have

$$D_{\text{P2P}}^{\text{selfish users}} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=k+1}^{N} u_i}\right\} \tag{2}$$

Given that $u_s + \sum_{i=1}^{N} u_i > u_s + \sum_{i=k+1}^{N} u_i$, we have $\frac{NF}{u_s+\sum_{i=k+1}^{N} u_i} > \frac{NF}{u_s+\sum_{i=1}^{N} u_i}$, the delay when there are selfish users will be no less than when all users upload.

Remark: The aim of this question is to test whether you understand how the expression $\frac{NF}{u_s+\sum_{i=1}^{N} u_i}$ is derived. If you do, you will be able to adapt the original solution to the new situation suggested in the question.

Question 4 – Sliding Windows (4 marks)

(a)

Go-back-n:

The time to transmit one data packet, L/R = 125 x 8 bits /1Mbps = 1ms

RTT = 2 x one-way propagation delay = 9ms

As discussed in Week 4 lectures slide 57, the utilization of the link (channel) when a sliding window protocol with window size N, is given by,

$U = N (L/R)/(L/R + RTT)$

Achieving a throughput of 0.8Mbps on a 1Mbps link implies that the utilisation should 0.8.

Thus, $0.8 = N \times 1ms /(1ms + 9ms)$. This results in $N = 8$ packets.

Selective repeat:

Over an error-free link, GBN and SR achieve exactly the same performance, so the above answer also applies to SR.

(b)

Go-back-n:

To achieve a window size of 8 with GBN, we will need 9 unique sequence numbers (1 more than the window size). Thus, it is necessary to have a 4-bit sequence number space in GBN.

Selective repeat:

As was discussed in the lecture notes, with SR, the window size must be less than or equal to half the size of the sequence number space. Thus, in order to accommodate a window size of 8, we need at least 16 unique numbers, which implies that SR would require a 4-bit sequence number space.

Question 5 – Switchapalooza (4 marks)

COMP3331 (UNDERGRADUATE VERSION)

Case 1)

What is L (in msec)? **200**

What is T (in Mbps)? **16**

The circuit between A and B as a whole can offer service with end-to-end bandwidth T, and end-to-end delay of L+L/2. Recall that in circuit switching, there is no store-and-forward delay at the router.

Time to transmit the 1MB file is 1 MB/T.

Thus, $1 MB/T + L + L/2 = 0.8$ sec - (1)

The 2nd file is sent back-to-back, thus,

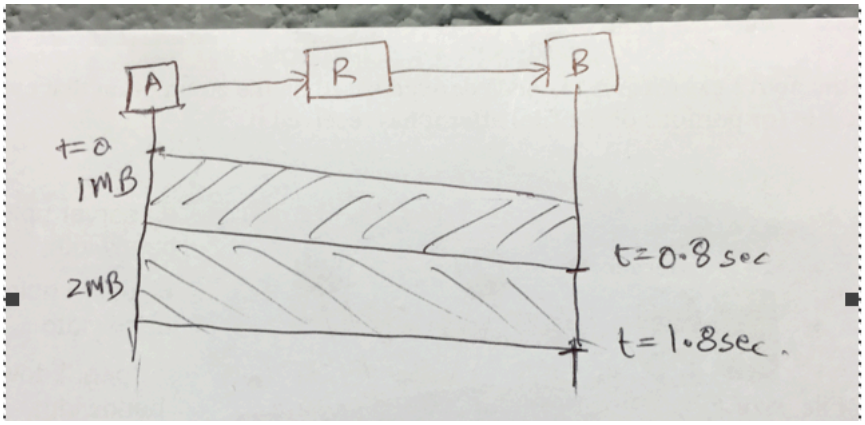
$(1MB + 2 MB) /T + L + L/2 = 1.8$ sec – (2)

Subtracting (1) from (2), $2MB/T = 1$ sec. Thus, $T = 2$ MB/s (= 16 Mbps.)

Substituting $T = 2$ MB/s in (1), 0.5 sec $+ 1.5 L = 0.8$ sec

Thus, L = 0.2 sec = 200 msec.

A timing diagram is included below to illustrate the delay

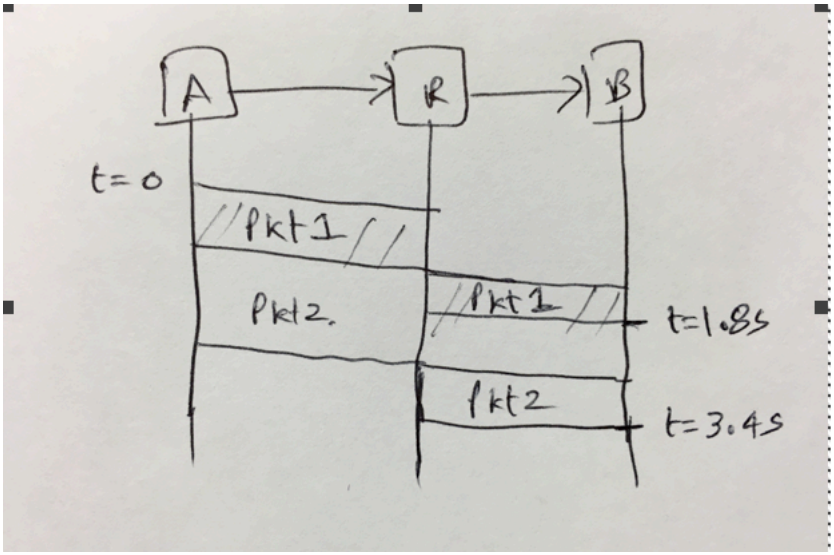


Case 2)

What is L (in msec)? **0.56**

What is T (in Mbps)? **12.5**

Since, the second link is faster than the first link, there is no queuing delay. See timing diagram below.



The first packet arrives at node B at 1.8ms.

Thus, (1KB/T) + L + (1KB/2T) + L/2 = 1.8 ms – (1)

The second packet will arrive at node B at 3.4 seconds.

Thus, (1KB/T) + (2KB/T) + L + (2KB/2T) + L/2 = 3.4 – (2)

Subtracting (1) from (2),

2KB/T + 1KB/2T = 1.6

Thus, T = 2.5 KB/1.6 = 12.5 Mbps

Substituting for T in (1),
1.5KB/12.5Mbps + 1.5L = 1.8ms

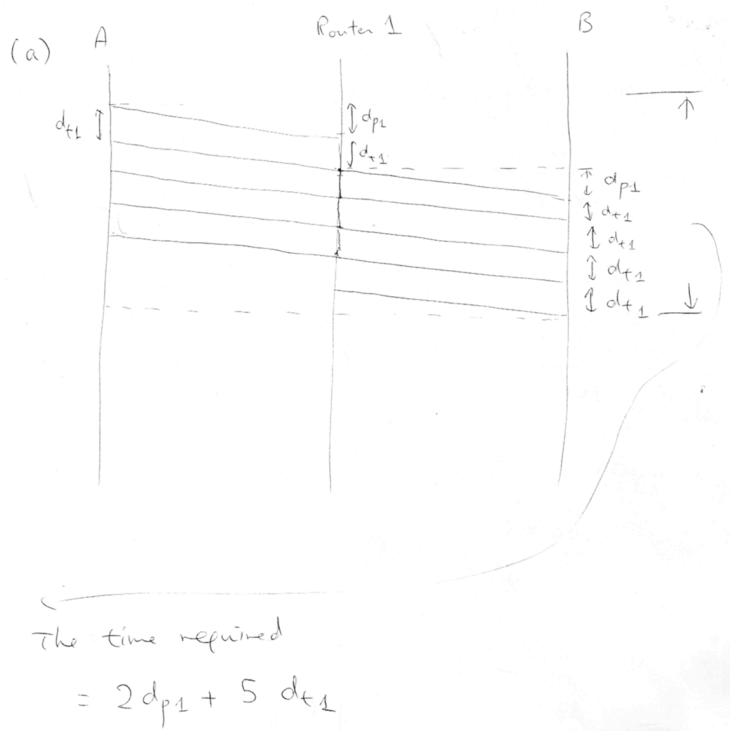
Thus, L = 0.56ms

COMP9331 (POSTGRADUATE VERSION)

(a) Let us begin with a number of basic calculations:

- Propagation delay in link type 1 (d_{p1}) = 2000 x 10^3/2 x 10^8 = 0.01 s
- Transmission delay in link type 1 (d_{t1}) = 8 x 1024/100 x 10^3 = 0.08 s (approx. 1024 by 1000)
- Propagation delay in link type 2 (d_{p2}) = 4000 x 10^3/2 x 10^8 = 0.02 s
- Transmission delay in link type 1 (d_{t2}) = 8 x 1024/50 x 10^3 = 0.16 s (approx. 1024 by 1000)

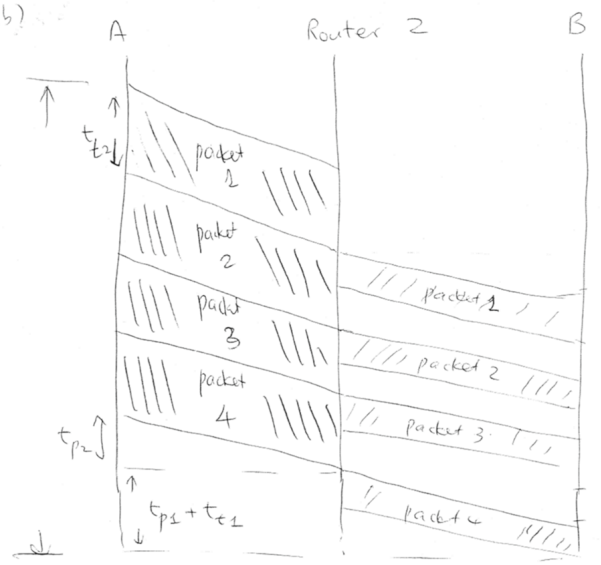
From the timing diagram below, the time required = $2d_{p1} + 5d_{t1}$ = 2(0.01) + 5(0.08) = 0.42 s.



(b)

From the timing diagram below, the time required = $d_{p1} + d_{t1} + d_{p2} + 4d_{t2}$ = 0.01 + 0.08 + 0.02 + 4(0.16) = 0.75 s.

(b)



The time required

$$= d_{p1} + t_{t1} + d_{p2} + 4\, t_{t2}$$