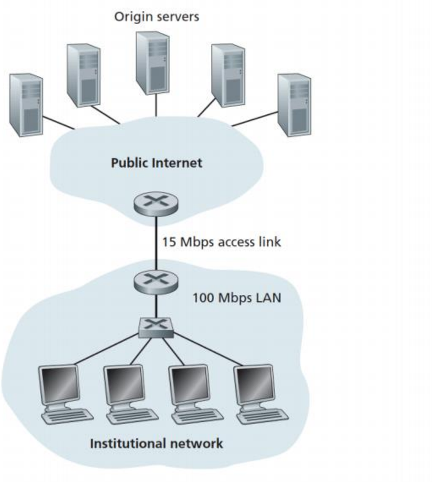
CS 655: Introduction to Computer Networks

Fall 2020

Homework 2

Ziqi Tan
U 88387934

1. (15 pts) Consider the figure below, for which there is an institutional network connected to the Internet. Suppose that the average object size is 850,000 bits and that the average request rate from the institution's browsers to the origin servers is 16 requests per second. Also suppose that the amount of time it takes from when the router on the Internet side of the access link forwards an HTTP request until it receives the response is three seconds on average. Model the total average response time as the sum of the average access delay (that is, the delay from Internet router to institution router) and the average Internet delay. For the average access delay, use Δ/(1 – Δb), where Δ is the average time required to send an object over the access link and b is the arrival rate of objects to the access link.



Bottleneck between an institutional network and the Internet

a. Find the total average response time.
   i. Internet delay is 3 seconds.
   ii. Δ = 850, 000 bits / 15 Mbps, b = 16 per seconds
       Δ/(1 – Δb) = 0.607 seconds
       The total average response time is 3 + 0.607 = 3.607 seconds

b. Now suppose a cache is installed in the institutional LAN. Suppose the miss rate is 0.4. Find the total response time.
   The total response time is 0.4 x 3.607 + (1-0.4) x 0 = 1.44 seconds

2. (15 pts) Consider a short, 10-meter link, over which a sender can transmit at a rate of 150 bits/sec in both directions. Suppose that packets containing data are 100,000 bits long, and packets containing only control (e.g., ACK or handshaking) are 200 bits long. Assume that N parallel connections each get 1/N of the link bandwidth. Now consider the HTTP protocol and suppose that each downloaded object is 100 Kbits long, and that the initial downloaded object contains 10 referenced objects from the same sender. Would parallel downloads via parallel instances of non-persistent HTTP make sense in this case? Now consider persistent HTTP. Do you expect significant gains over the non-persistent case? Justify and explain your answer.

Non-persistent HTTP: each connection gets 1/10 of the link bandwidth.
For each request, it takes (200 bits x 3 + 100, 000 bits) / (150 bps / 10) = 6706 seconds.

Persistent HTTP: (200 bits x 3 + 100, 000 bits x 10) / 150 bps = 6671 seconds.

Persistent HTTP takes 35 seconds less than non-persistent HTTP. In this case, it has 0.5% gains over non-persistent case. It is not a significant gain.

3. (10 pts) Consider distributing a file of F = 15 Gbits to N peers. The server has an upload rate of u_s = 30 Mbps, and each peer has a download rate of d_i = 2 Mbps and an upload rate of u. For N = 10, 100, and 1,000 and u = 300 Kbps, 700 Kbps, and 2 Mbps, prepare a chart giving the minimum distribution time for each of the combinations of N and u for both client-server distribution and P2P distribution.

Client-Server:

Minimum distribution time (seconds)	N = 10	N = 100	N = 1000
u = 300 kbps	7500	50000	500000
u = 700 kbps	7500	50000	500000
u = 2 Mbps	7500	50000	500000

P2P:

Minimum distribution time (seconds)	N = 10	N = 100	N = 1000
u = 300 kbps	7500	25000	4.5e+04
u = 700 kbps	7500	15000	2.1e+04
u = 2 Mbps	7500	7500	7500

4. (10 pts) Suppose Bob joins a BitTorrent torrent, but he does not want to upload any data to any other peers (so called free-riding).
   a. Bob claims that he can receive a complete copy of the file that is shared by the swarm. Is Bob's claim possible? Why or why not?
   b. Bob further claims that he can further make his "free-riding" more efficient by using a collection of multiple computers (with distinct IP addresses) in the computer lab in his department. How can he do that?

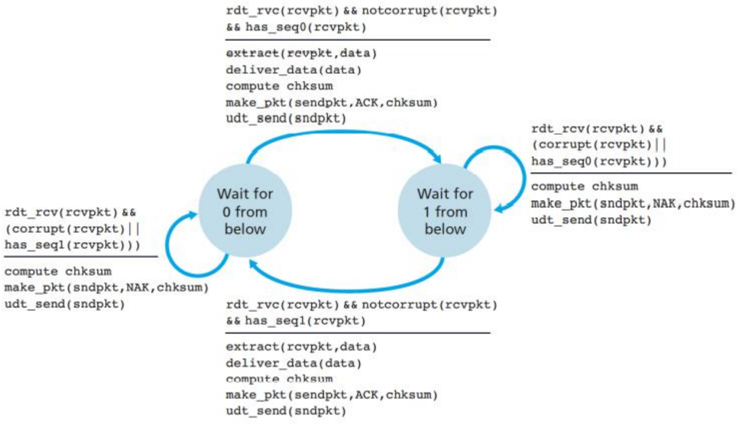
   Answer:

   a. It is possible that all chunks of the file have been distributed on different peers. Thus, Bob can receive a complete copy of this file. According to the trading algorithm, as long as there are enough peers, Bob can always become optimistically-unchocked and receive data from other peers.
   b. If the claim in a is correct, the claim in b is also correct. Bob can let each computer ask different chunks of the file through optimistically-unchocking and then combine these chunks.

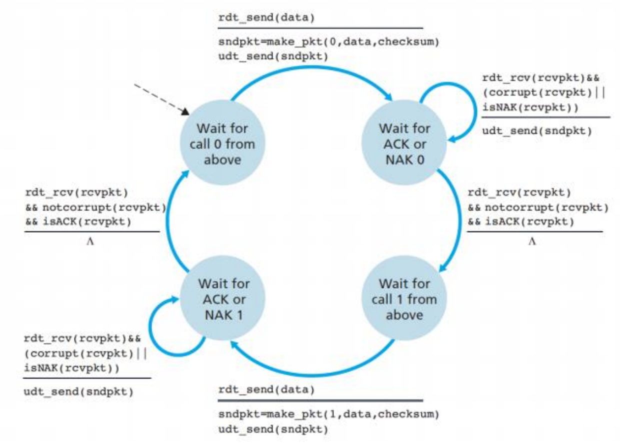
5. (10 pts) Show that the (incorrect) receiver, shown in figure below, when operating with the sender shown, can lead the sender and receiver to enter into a deadlock state, where each is waiting for an event that will never occur.

   Consider the following scenario where a deadlock will happen:

   1. Sender state is in "wait for call 0 from above".
   2. Receiver state is in "wait for 0 from below".
   3. The sender sends a packet with sequence number 0 and then the state of the sender will become "wait for ACK or NAK 0".
   4. The receiver gets the uncorrupted packet 0 and then send the ACK to the sender. The state of the receiver will become wait for 1 from below.
   5. **Now, the ACK packet is corrupted. The sender retransmits packet 0 again.**
   6. Unfortunately, the receiver is waiting for packet 1. Thus, it will send NAK back to the sender.
   7. **The sender then again retransmits the packet 0.** The deadlock happens.



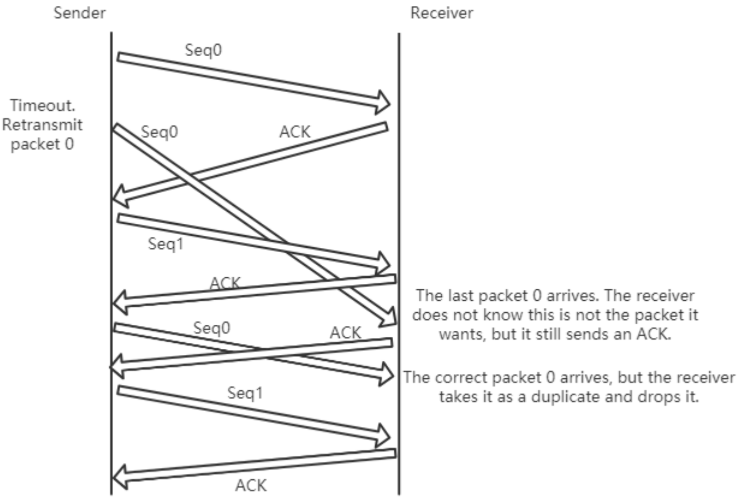
An incorrect receiver



Sender

6. (10 pts) Consider the alternating-bit (stop-and-wait) protocol. Draw a diagram showing that if the network connection between the sender and receiver can reorder messages (that is, that two messages propagating in the medium between the sender and receiver can be reordered), then the alternating-bit protocol will not work correctly (make sure you clearly identify the sense in which it will not work correctly). Your diagram should have the sender on the left and the receiver on the right, with the time axis running down the page, showing data (D) and acknowledgment (A) message exchange. Make sure you indicate the sequence number associated with any data or acknowledgment segment.

Answer:
The channel can reorder the message. Since we use the stop and wait protocol, it is impossible that the receiver wants packet 0, but receive packet 1. Thus, bad things would happen when the receiver get two packets with the same sequence number. For concreteness, let's see an example.



The last packet 0 arrives. The receiver does not know this is not the packet it wants, but it still sends an ACK.

The correct packet 0 arrives, but the receiver takes it as a duplicate and drops it.

7. (15 pts) Consider a scenario in which Host A wants to simultaneously send packets to Hosts B and C. A is connected to B and C via a broadcast channel—a packet sent by A is carried by the channel to both B and C. Suppose that the broadcast channel connecting A, B, and C can independently lose and corrupt packets (and so, for example, a packet sent from A might be correctly received by B, but not by C). Design a stop-and-wait-like error-control protocol for reliably transferring packets from A to B and C, such that A will not get new data from the upper layer until it knows that both B and C have correctly received the current packet. Give FSM descriptions of A and C. (Hint: The FSM for B should be essentially the same as for C.) Also, give a description of the packet format(s) used.

Answer:

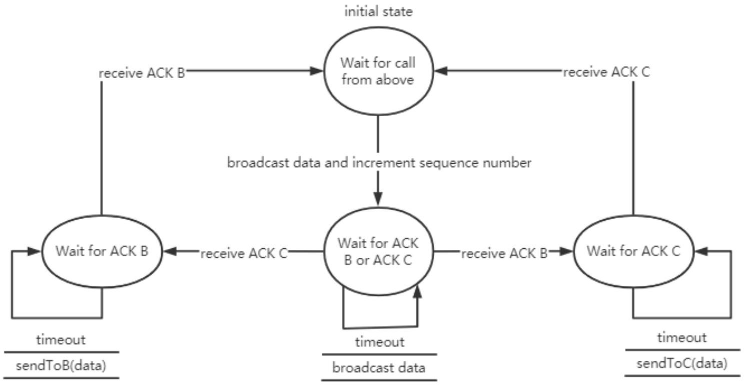
Description of packet format:

Sender to receiver packet format:

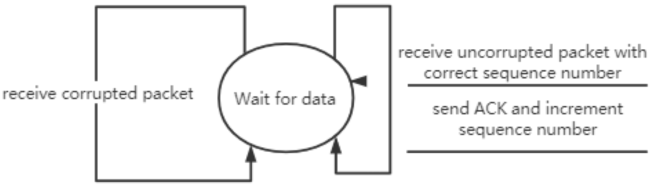
Sequence number	data

Receiver to sender packet format:

Ack sequence number	Host: B or C



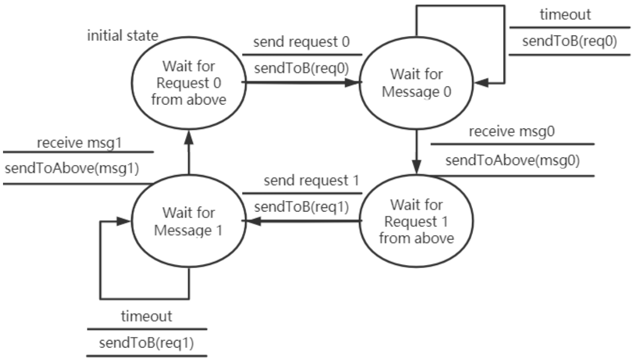
FSM for A


receive corrupted packet
Wait for data
receive uncorrupted packet with correct sequence number
send ACK and increment sequence number

FSM for B

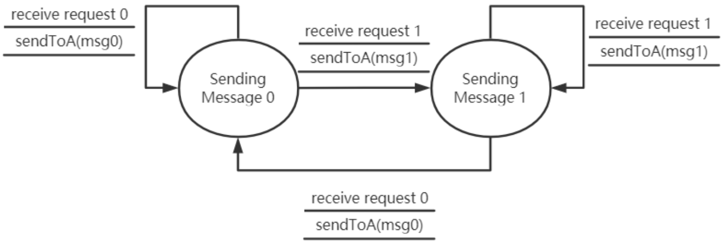
8. (15 pts) Suppose we have two network entities, A and B. B has a supply of data messages that will be sent to A according to the following conventions. When A gets a request from the layer above to get the next data (D) message from B, A must send a request (R) message to B on the A-to-B channel. Only when B receives an R message can it send a data (D) message back to A on the B-to-A channel. A should deliver exactly one copy of each D message to the layer above. R messages can be lost (but not corrupted) in the A-to-B channel; D messages, once sent, are always delivered correctly. The delay along both channels is unknown and variable. Design (give an FSM description of) a protocol that incorporates the appropriate mechanisms to compensate for the loss-prone A-to-B channel and implements message passing to the layer above at entity A, as discussed above. Use only those mechanisms that are absolutely necessary.

Answer:



FSM for A

1. Initially, A "wait for request 0 from above". In this state, A does not receive any message from B. Once send the request 0, the state transits to "wait for message 0".
2. Then, A starts a timer and wait for message 0. In this state, A ignores message 1. If timeout happens, re-send the request 0. If A receives message 0, A transits to wait for request 1 from above.
3. In this state, A does not receive any message from B. Once send the request 1, the state transits to "wait for message 1".
4. Then, A starts a timer and wait for message 1. In this state, A ignores message 0. If timeout happens, re-send the request 1. If A receives message 1, A transits to wait for request 0 from above.



FSM for B

1. Initially, B is in state "sending message 0". If B receives request 0, send message 0 back to A. If B receives request 1, send message 1 back to A and then transits to "sending message 1".
2. If B receives request 1, send message 1 back to A. If B receives request 0, send message 0 back to A and then transits to "sending message 0".

9. (50 pts) This exercise relates to the GENI lab "Two queues or not two queues? The benefit of statistical multiplexing" (https://witestlab.poly.edu/blog/two-queues-or-nottwo-queues-the-benefit-of-statistical-multiplexing). You are asked to compare the M/M/1 results to the M/D/1 results where "D" stands for deterministic (i.e., constant) service times. To produce constant service times, you should use the –c option when you run the traffic generation sender, for example:

   ITGSend -a receiver-1 -E 110 -c 512 -T UDP -t 600000

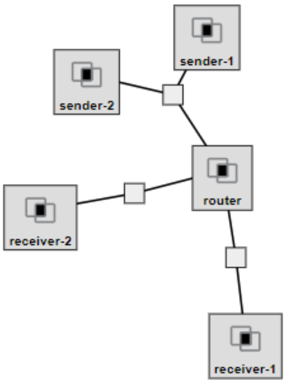
   generates a Poisson traffic stream to receiver-1 with a mean packet rate of 110 packets per second, and constant packet sizes of 512 B. You should also compare your experimental results to those obtained analytically for the M/D/1 queuing system, where the average packet delay is given by:

$$\frac{1}{2\mu} * \frac{2-\rho}{1-\rho}$$

   Submit the CDF of delay results for the percentile values (0.25, 0.5, 0.75, 1) produced by the R script, one for M/M/1 and another for M/D/1, along with your observations and brief analysis of how the two systems compare and why. You only need to include the CDF data and not the plots. In your analysis, you may want to try different parameters, for example, packet rate of 220 packets per second and average packet size of 256. (Hint: each link/network supports a maximum packet size, referred to as MTU. Any packet larger than MTU is either dropped or fragmented.)

Topology:

Sender-1: ziqi1756@pc3.instageni.clemson.edu -p 31413
Sender-2: ziqi1756@pc3.instageni.clemson.edu -p 31414
Router: ziqi1756@pc3.instageni.clemson.edu -p 31412
Receiver-1: ziqi1756@pc3.instageni.clemson.edu -p 31410
Receiver-2: ziqi1756@pc3.instageni.clemson.edu -p 31411



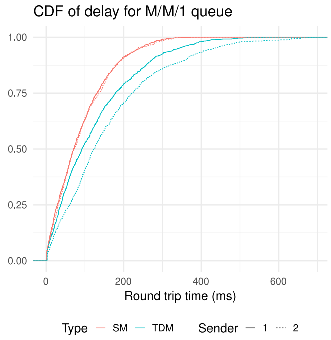
Interface to sender-1	interface-0	10.10.1.1
Interface to sender-2	interface-1	10.10.1.2
Interface to router	interface-2	10.10.1.3

Interface to router	interface-3	10.10.2.1
Interface to receiver-1	interface-4	10.10.2.2

Interface to router	interface-5	10.10.3.1
Interface to receiver-2	interface-6	10.10.3.2

Results:

A mean packet rate of 110 packets per second, and constant packet sizes of 512 B.



	25%	50%	75%	100%
$SM	29.975	71.050	134.000	374.000
$TDM	44.85	110.00	201.00	691.00

Theoretical value:

SM:

λ = 110 packets / second * 2

μ = 1e6 / (8*512) ≈ 244.14 packets/second

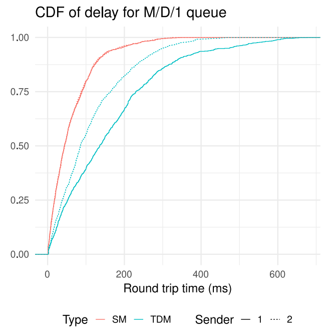
$$T_{SM} = \frac{1}{\mu - \lambda} = 41.425\ ms$$

TDM:

λ = 110 packets / second

μ = 500e3 / (8*512) ≈ 122.07 packets/second

$$T_{TDM} = \frac{m}{\mu - \lambda} = \frac{1}{122.07 - 110} ms = 82.850\ ms$$



	25%	50%	75%	100%
$SM	17.800	44.200	88.925	345.000
$TDM	44.45	106.00	201.25	677.00

Theoretical value:

SM:

λ = 110 packets / second * 2

μ = 1e6 / (8*512) ≈ 244.14 packets/second

$$\rho_{SM} = \frac{\lambda}{\mu} = \frac{220}{244.14} = 0.90$$

$$T_{SM} = \frac{1}{2\mu} * \frac{2-\rho}{1-\rho} = 22.528\ ms$$

TDM:

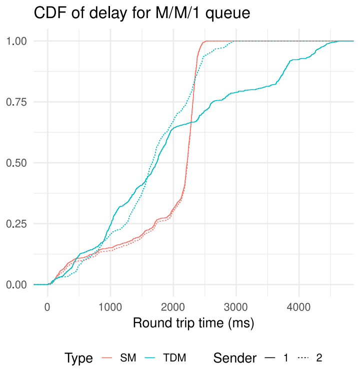
λ = 110 packets / second

μ = 500e3 / (8*512) ≈ 122.07 packets/second

$$\rho_{TDM} = \frac{\lambda}{\mu} = \frac{110}{122.07} = 0.90$$

$$T_{TDM} = \frac{1}{2\mu} * \frac{2-\rho}{1-\rho} = 45.056\, ms$$

A mean packet rate of 220 packets per second, and constant packet sizes of 256 B.



	25%	50%	75%	100%
$SM	1718.0	2222.0	2310.5	2529.0
$TDM	1050	1689	2331	4638

Theoretical value:

SM:

λ = 220 packets / second * 2

μ = 1e6 / (8*256) ≈ 455.28 packets/second

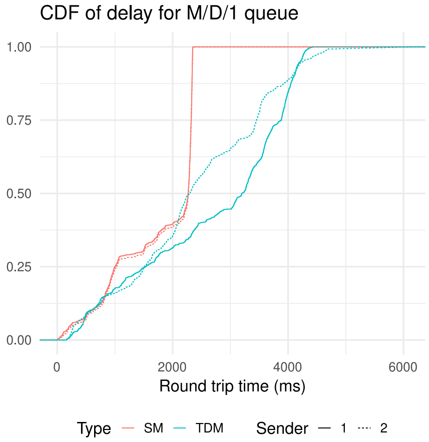
$$T_{SM} = \frac{1}{\mu - \lambda} = 65.445\ ms$$

TDM:

λ = 220 packets / second

μ = 500e3 / (8*256) ≈ 244.14 packets/second

$$T_{TDM} = \frac{m}{\mu - \lambda} = \frac{1}{244.14 - 220} ms = 41.425\ ms$$



	25%	50%	75%	100%
$SM	1016	2270	2322	2352
$TDM	1518.0	2601.5	3602.0	6076.0

Theoretical value:

SM:

λ = 220 packets / second * 2

μ = 1e6 / (8*256) ≈ 488.28 packets/second

$$\rho_{SM} = \frac{\lambda}{\mu} = \frac{440}{488.28} = 0.9$$

$$T_{SM} = \frac{1}{2\mu} * \frac{2-\rho}{1-\rho} = 11.264\ ms$$

TDM:

λ = 220 packets / second

μ = 500e3 / (8*256) ≈ 244.14 packets/second

$$\rho_{TDM} = \frac{\lambda}{\mu} = \frac{220}{244.14} = 0.9$$

$$T_{TDM} = \frac{1}{2\mu} * \frac{2-\rho}{1-\rho} = 22.528\, ms$$

Analysis:

In the case where the mean packet rate is 110 packets per second and constant packet sizes is 512 B, statistical multiplexing (SM) outperforms time division multiplexing (TDM). As we expect, the T_{SM} is half of T_{TDM}.

In the case where the mean packet rate is 220 packets per second and constant packet sizes is 256 B, SM is not always better than TDM.

The experimental delay is always larger than the theoretical delay. Especially in the second case, the experimental delay is much larger.