CNT5106C Computer Networks
Instructor: Prof. Ahmed Helmy
Homework #2 Solution
On the Transport Layer, Congestion Control and TCP

Q1. (8 points: 4 points each)

I. Consider a reliable data transfer protocol that uses only negative acknowledgements. Suppose the sender sends data only infrequently. Would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

II. Now suppose the sender has a lot of data to send and the end-to-end connection experiences few losses. In this second case, would a NAK-only protocol be preferable to a protocol that uses ACKs? Why?

Ans. 1.

I. In a NAK only protocol, the loss of packet *x* is only detected by the receiver when packet *x+1* is received. That is, the receivers receives *x-1* and then *x+1,* only when *x+1 is* received does the receiver realize that *x* was missed. If there is a long delay between the transmission of x and the transmission of *x+1,* then it will be a long time until *x* can be recovered, under a NAK only protocol.

II. On the other hand, if data is being sent often, then recovery under a NAK-only scheme could happen quickly. Moreover, if errors are infrequent, then NAKs are only occasionally sent (when needed), and ACK are never sent – a significant reduction in feedback in the NAK-only case over the ACK-only case.

Q2. (12 points: 4 points each case) Consider a transport protocol that uses connection-oriented network service. Suppose that the transport protocol uses a credit allocation flow control scheme (such as TCP), and the network protocol uses a sliding-window scheme. What relationship should there exist between the dynamic window of the transport protocol and the fixed window of the network protocol, such that the transport protocol is effective? Consider the following cases: I. There is one-to-one relationship between a transport layer connection and network layer connection. II. A transport layer connection may be split among multiple network layer connections. III. Multiple transport layer connections may be multiplexed in one network layer connection.

Ans 2.

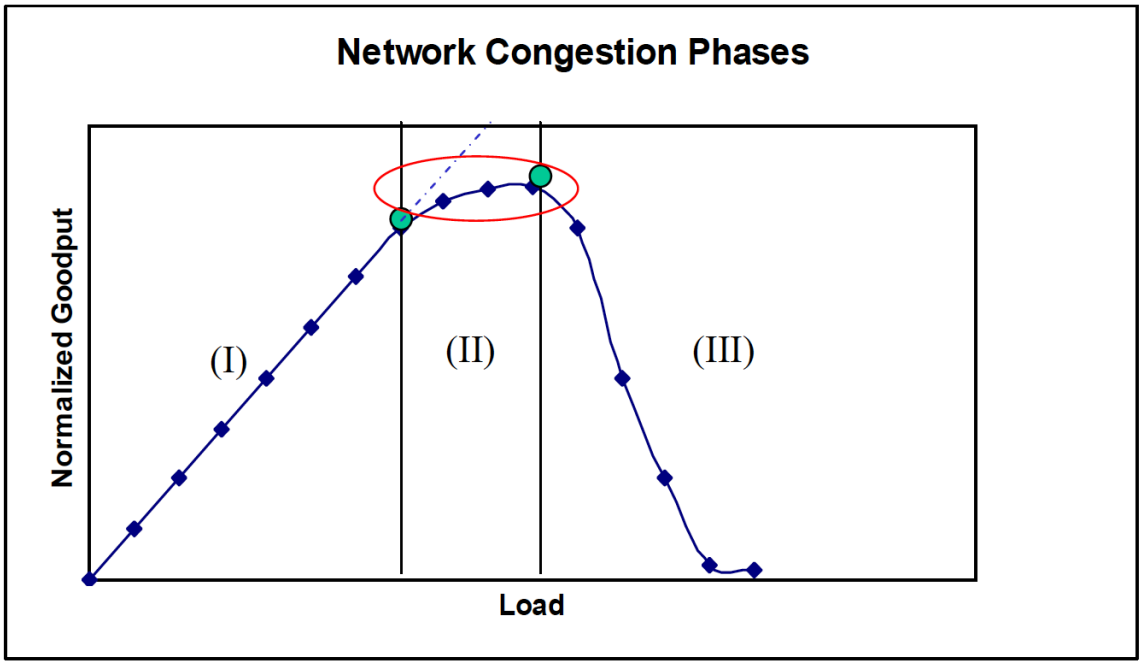
I. If there is one-to-one relationship between network connections and transport connections, then it will do no good to grant credit at the transport level in excess of the window size at the network level. So, in order to be effective, the window size of the transport protocol should be less than that for the network protocol.

II. If one transport connection is split among multiple network connections, then the credit/window of the transport protocol should be less than the sum of the network window sizes.

III. If multiple transport connections are multiplexed on one network connection their aggregate credit/window should not exceed the network protocol window.

Q3. (14 points: 6 + 8)
I. Describe (with the aid of a graph) the different phases of network load/overload outlining the degrees of congestion with increase of load. Indicate the point of congestion collapse and explain why it occurs. (6 points)
II. Where does TCP operate on that graph? Explain for the various phases of TCP; slow start, congestion avoidance (due to timeout), fast retransmit-fast recovery triggered by duplicate ACKs. (8 points)

Ans 3.
I.



(I) No Congestion
(II) Moderate Congestion
(III) Severe Congestion (Collapse)

- no congestion --> near ideal performance
- overall moderate congestion:
  - severe congestion in some nodes
  - dynamics of the network/routing and overhead of protocol adaptation decreases the network Tput
- severe congestion:
  - loss of packets and increased discards
  - extended delays leading to timeouts
  - both factors trigger re-transmissions
  - leads to chain-reaction bringing the Tput down

II. For TCP: in slow start, the load starts from CongWin=1 (at the beginning of phase I), then ramps up quickly (exponential growth of CongWin) until a loss is experienced (in phase II or beginning of phase III).

After the loss, if a timeout occurs, TCP goes down to CongWin=1 (at the beginning of phase I) then ramps up to roughly half the load the led to the loss (i.e., half way in phase I).
In congestion avoidance CongWin increases linearly, which means the load increases slowly towards the end of phase I and into phase II, until another loss occurs.
In fast retransmit fast recovery (due to duplicate acks), the load is cut in half (half way into phase I), then slow (linear) increase towards phase II (as in congestion avoidance).

Q4. (20 points: 4 points each)
Let's take a closer look at the reasoning behind TCP Reno's window adjustment algorithm. The algorithm is as follows (from the lecture slides):

- Initially we use Slow start:
  CongWin = *CongWin* + 1 with every ack
- When timeout occurs we enter congestion avoidance:
  ssthresh=*CongWin/2, CongWin*=1
  slow start until *ssthresh*, then increase linearly
  CongWin=*CongWin*+1 with every *RTT*, or
  CongWin=*CongWin*+1/*CongWin* for every ack
- Fast recovery: when 3rd dup ack arrives
  ssthresh=*CongWin/2*
  retransmit segment, set *CongWin*=*ssthresh*+3
  for every duplicate ack *CongWin*=*CongWin*+1
  after receiver gets cumulative ack: *CongWin*=*ssthresh*

Answer the following questions based on your understanding about this algorithm.

(a) Why do we set *ssthresh* = *CongWin/2* upon entering congestion avoidance? Why not set *ssthresh*= *CongWin/3* or *ssthresh*=*CongWin*-20? Is there any particular reason, or this is an arbitrary decision?
(b) Why do we set *CongWin* = 1 upon entering congestion avoidance in the basic TCP algorithm (the one WITHOUT fast recovery)?
(c) Why do we set *CongWin* = *ssthresh* + 3 when we receive the 3rd dup ack in the fast recovery algorithm?
(d) In the fast recovery algorithm, for every dup ack *CongWin* = *CongWin* + 1, is this equivalent to exponential increase?
(e) What is the feature in the fast recovery algorithm that makes it possible to set *CongWin*=*ssthresh* instead of *CongWin*=1 when TCP exits fast recovery?

Ans. 4

(a) This is because before entering congestion avoidance we used exponential increase, letting CongWin be doubled every round trip time. Assuming there is a certain "network capacity" we want to probe using TCP sliding window algorithm, the last increase makes CongWin to exceed this capacity and causes packet loss. Since we have no knowledge about for how much we go beyond the capacity, the easy and conservative thing to do is to assume the old CongWin before the last increase was the upper limit and set ssthresh = (current) CongWin/2. By doing so TCP will stop

exponential window growth next time at this ssthresh and use conservative linear increase after it.

(b) The original TCP algorithm ignores dup ACK and does not distinguish between moderate congestion and severe congestion. As a timeout event occurs, it assumes that the network is in congestion and immediate action should be taken to alleviate it, so it takes the extreme measure to reduce CongWin to the smallest possible value, 1.

(c) The "3" in the equation represents the 3 packets that triggered the 3 dup acks. The 3 dup acks indicate that 3 packets are removed from the pipe, and hence it is save to add 3 more packets into the pipe beyond the safe window size, ssthresh.

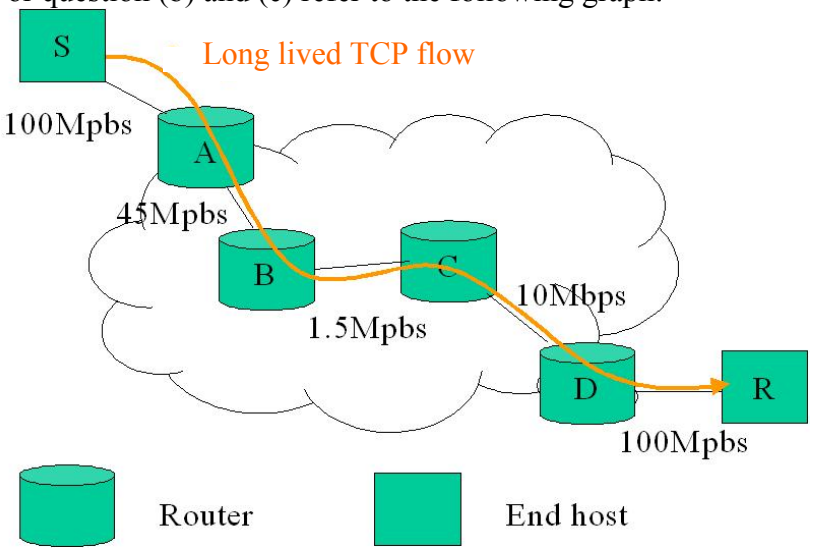
(d) No, since for receiving dup acks the lower end of congestion window does not increase. So CongWin = CongWin + 1 in this case allows the sender to add at most one packet into the pipe only. Note that the previous reduction in CongWin (CongWin = ssthresh + 3 as we studied in (c)) may keep the sender from sending new packets if the number of outstanding packet (packets that are sent, but not ack-ed so far) is more than current CongWin. The idea of this CongWin updating equation is to keep the total number of packets in the pipe at a constant value, ssthresh, since for every one packet removed from the pipe (the one generated the dup ack), one is added from the sender (or, if the outstanding packet number is still greater then CongWin, no new packet will be sent, only the CongWin is incremented).

(e) It is the increment of CongWin for dup acks that makes this possible. The goal of setting CongWin=ssthresh+3 upon receiving 3 dup acks and adjusting window size by CongWin=CongWin+1 is to keep ssthresh packets in the pipe. When TCP finally gets out of fast recovery, there are ssthresh packets in the pipe and CongWin=ssthresh, hence it does not lose the self-clocking and can safely continue from there without going back to CongWin=1.Note that this only happens if fast retransmit is able to resolve the packet loss in time so that the timer for the packet we received duplicated acks does not expire. If the timer expires, we still follow the congestion avoidance algorithm and go back to CongWin=1.

Q5. (9 points (a), (b) + 8 points part (c)) Interaction between TCP and other flows:

(a) (4 points) In a scenario where long lived TCP flows share several links with UDP flows, explain what happens in cases of congestion? [i.e., who gets more share of the bandwidth, and who experiences more packet loss, etc.]

For question (b) and (c) refer to the following graph:



(b) (5 points) If someone were to inject UDP traffic into the network to attempt to harm the long lived TCP flows the most using the least amount of traffic/bytes, where in the network would they inject such UDP traffic? Please specify the location (at which router) and the direction (toward S or D) you would inject the UDP flow and clarify.

(c) (8 points) Consider a scenario in which the network is protected against excessive UDP flows (using UDP filtering gateways/firewalls) and TCP Syn flood attacks (So if you attack by generating a lot of new TCP flows you will also get caught). If someone were to inject traffic into the network to attempt to harm the long lived TCP flows the most, where in the network would they inject such traffic? Note that now you are limited to using small number of 'short' TCP flows. Clarify your strategy as compared to (b) above, would you attack the same link(s)? Please specify the location (at which router) and the direction (toward S or D) you would inject the short TCP flow and clarify.

[Hint: note that short TCP flows spend most of their lifetime in the slow start phase, and they need to get acks to open their window size (i.e., they cannot increase their rate arbitrarily)]

Ans. 5

(a) (4 points) IF the UDP and TCP flows share a bottleneck link then they would compete for bandwidth on that link. When congestion happens on the bottleneck link TCP will backoff (i.e., cut/adjust its window size) and UDP will not (because it doesn't have congestion control). So UDP will dominate the bandwidth while TCP will get less share of bandwidth. UDP will also experience the most losses since it is more aggressive.

(b) (5 points) The UDP traffic should be injected at the bottleneck link, since it takes the least amount of traffic to saturate that link. Also since TCP data packets are much larger than ACK packets, the attack traffic should follow the same direction as the data traffic. (i.e. Inject the UDP flow at router B, toward R).

(c) (8 points) If injected at the bottleneck link then the short TCP flows will not have enough time/capacity to open up their window size, because they will also experience congestion, loss of packets (and hence lack of acks) and backoff like the long TCP flows. To be more effective the short TCP flows may be injected in non-bottleneck links so that the short TCP flow can expand its window size and generate more packets to squeeze packets of the long lived TCP flow out of the link. In the graph if link BC is already congested, it is not effective to inject the TCP flow before link BC toward the receiver since the short-lived TCP will not be able to expand its window size and a few packets can do only little damage. In this case it is better to inject the short TCP flow at C toward R. The sufficient bandwidth on link CD allows the short TCP to open up its congestion window and send more packets to compete with the victim long-lived flow at link CD, causing more packet drop.
An alternative option would be introducing the short TCP flow at router A and letting it exit the network at router B. As long as the short TCP flow does not traverse the bottleneck link BC, it gets the necessary free capacity to expand its window size, and eventually cause congestion and packet drops of both long-lived and short TCP flows at the attacked link.
[note: an argument can be made that it depends on the excess capacity in the bottleneck link... if it's enough for the short TCP flows to open their window size, then the attack would still be effective even if the short TCP flow attacks the bottleneck link...]

Q6. (12 points) Based on your understanding of explicit and implicit congestion signaling discuss why TCP's performance in general degrades over wireless links? If the network provides explicit congestion signaling discuss modifications you would propose to TCP such that its performance improves over wireless networks.
Discuss possible drawbacks and improvement of your initial solution above considering both wireless and wired links.

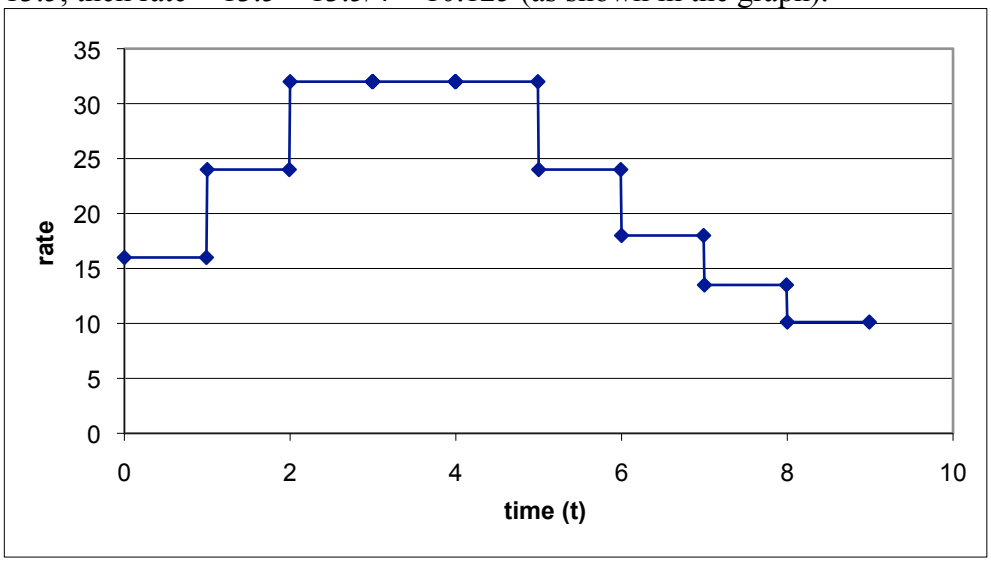
Ans 6.
. TCP cannot differentiate between losses due to congestion and losses due to bit error. A loss would lead TCP to cut down its window size (thinking it would help alleviate congestion in the network) where in fact there may not be congestion. So the end result is performance degradation of TCP.
Explicit Congestion Notification (ECN) provides TCP with an indication of congestion, so when a packet is lost and there is no notification of congestion it is likely to be due to BER on the wireless links.
TCP can be modified to take advantage of this explicit notification in the following way. The receiver side would have to be modified to include the ECN bit in the acks. The sender side would check the ECN field in the acks. So long as ECN bits are included in the acks, TCP could acts as the basic TCP (i.e., cut back window size with loss of

packets). If, on the other hand, loss occurs while ECN bits are not observed in the acks, then no window cut is performed by the sender.
Another modification could be to have the sender cut back its window size (say by half) when it observes ECN bits in the acks.
(8 points for reasonable and correct suggestions in the above part)

Drawbacks should consider confusion of the TCP sender because it doesn't know whether the packets are crossing a wireless network or not (if the proposed solution depends on it). The students should also mention the scenarios of partial deployment of the ECN in the network and discuss its consequences… (4 points for a sound argument)

Q7. (14 points) ATM ABR rate-controlled flow, starts with initial cell rate (ICR) of 16k cells/sec. The peak cell rate (PCR)=32k cells/sec, the minimum cell rate (MCR)=1k cells/sec, the maximum burst size (MBS) is 100k cells, and the RM cell rate is 1cell/sec (these RM cells were received by the sender). The rate increase factor (RIF)=1/4, and the rate decrease factor (RDF)=1/4. The no increase (NI) bit was always '0' in the RM cells, while the setting of the congestion indication (CI) bit in the consecutive RM cells was as follows: 0,0,0,0,1,1,1,1. Draw a graph showing the cell rate of the source against time with intervals of 1 sec, starting at time t=0 when the sender started with ICR, up to t=8 secs.

Ans. 7
If CI=0 and NI=0 then the rate increases using the following equation
rate = rate + PCRxRIF [additive increase]
until the rate reaches PCR, then it saturates
PCRxRIF = 32x1/4=8, so we increase in increments of 8 until 32
If CI=1 then the rate decreases using the following equation
rate = rate – rate x RDF
we will have rate = 32 – 8 = 24, then rate = 24 – 6 = 18, then rate = 18 – 18/4 = 13.5, then rate = 13.5 – 13.5/4 = 10.125 (as shown in the graph).



Q8. (8 points) In ATM ABR congestion control the equation to decrease the rate is given by:

$Rate_{new}=Rate_{old}-Rate_{old}*RDF$, where *RDF* is the rate decrease factor,

a. discuss how fast/slow does the sender respond to congestion for the various value of RDF.

b. If the equation was changed to $Rate_{new}=Rate_{old}*\beta$, do you think the response will be better or worse and why.

Ans. 8.

a. for high RDF the response will be fast, potentially causing oscillations. If RDF is low then the response will be slow.

b. The response is very similar to part 'a'. Note that there is no difference in the mechanism if we put β =1-RDF.

We get $Rate_{new}=Rate_{old}-Rate_{old}*RDF= Rate_{old}\ (1-RDF) =Rate_{old}*\beta$

The rate of response will be reversed (as compared to part a above), i.e., when beta is high (close to 1) the response will be slow, but when beta is low (closer to 0) then the response is fast.

Q9. (8 points) Argue for or against this statement (reason using examples as necessary): "Packets are lost only when network failures occur (e.g., a link goes down). But when the network heals (e.g., the failed link comes back up again), packets do not get lost."

Ans. 9

When the network fails (e.g., a link goes down), a number of packets that cross that link may be lost.

When the network heals, packets/flows may cross relatively shorter paths to get to the destination. Shorter paths have a delay-bandwidth product less than longer paths, and hence can hold fewer bytes in the pipe (i.e., fewer segments can be in flight). Having a TCP connection with a high CongWin go over a shorter path may also cause packet loss, since the shorter paths will get congested, and buffers may overflow causing multiple (sometimes severe) packet losses.

Q10. (10 points) The delay performance of a particular ATM network is dominated by one critical physical link, which is shared by three flows (referred to in ATM as virtual paths or VPs). Each time slot, the probability of a cell arriving on the first, second, and third VP is 0.1, 0.2, and 0.3, respectively, and cell arrivals are independent in time. Assume that the statistical multiplexer at the head end of every VP can store an arbitrarily large number of cells.

a. Find the allocation of link bandwidth among the three VPs such that the average cell delay is the same for all VPs

b. Repeat part a if the probability of a cell arriving on the first, second, and third VP is 0.05, 0.05, and 0.5, respectively.

Ans. 10
Using the fluid flow model

a. For the delay to be the same

$$\frac{P_1}{f_1} = \frac{P_2}{f_2} = \frac{P_3}{f_3}$$

We have

$$\left\{ \begin{array}{l} \frac{0.1}{f_1} = \frac{0.2}{f_2} = \frac{0.3}{f_3}, \\ f_1 + f_2 + f_3 = 1. \end{array} \right\}$$

Finally,

$$f_1 = 1/6; f_2 = 2/6; f_3 = 3/6$$

b.

$$\left\{ \begin{array}{l} \frac{0.05}{f_1} = \frac{0.05}{f_2} = \frac{0.5}{f_3}, \\ f_1 + f_2 + f_3 = 1. \end{array} \right\}$$

Finally,

$$f_1 = 1/12; f_2 = 1/12; f_3 = 10/12$$