Đề ôn tập PE NWC203c Summer 2021- số 5 (sinh viên phải làm từng bước tới kết quả)

1. Three users are sharing a common link of 1 Mb/s. User *A* is downloading large files and is connected to the common link via a slow access link at *x* Mb/s, user *B* is connected via a 100 Mb/s link, but the application she is using requires at most *x* Mb/s, and finally user *C* is connected via a 1 Gb/s link and is downloading a movie that can take up any amount of bandwidth available to it. What is the max-min fair allocation for these three flows at the common link?

Two Cases: 1) If x < 1/3Mb/s then A and B get x Mb/s, C gets (1-2x)Mb/s, otherwise A, B, and C get 1/3 Mb/s

2. If we want to "break-in" to an ongoing TCP connection, and pretend to be one of the end-hosts, we need to send an sequence number that will convince the other end that we are legitimate. For example, imagine that a source host, A, tries to create a TCP connection with host B. When host A sends the SYN message, host C (masquerading as B) sends a SYN+ACK message to host A with the correct sequence number. Even though C didn't see the original SYN message, it gets lucky and guesses the right sequence number to send back so as to fool host A.

(a.) If Host C were to guess the value from among all possible 32-bit sequence numbers, on average how many guesses would it take to get it right?

2^{31}

If Host C already knew 12 of the bits (e.g. because it knew something about the pseudo-random ISN generator), on average how many guesses would it take to get it right?

2^{19}

If Host C wanted to send a spoofed email to A, it would take just one "guessed" packet. Assuming the e-mail body and packet headers were 1500bytes, given a 1Gb/s link (and assuming that C already knew 12 bits of the sequence number), on average how long would it take to send one e-mail with a spoofed source?

*2^{19} * 1500bytes * 8bits/byte * 1/ 10^{9} = 6.3seconds*

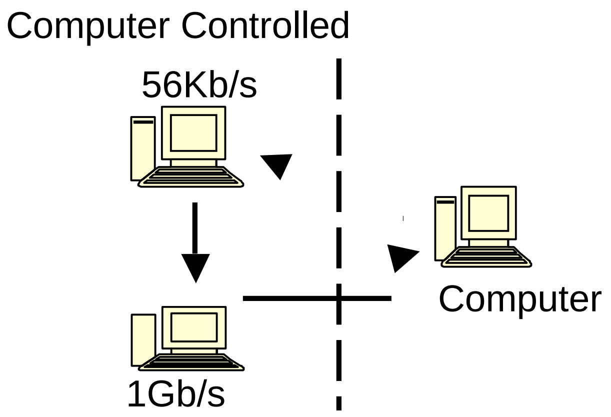
(b.) What simple change could you add to the SMTP protocol to defend against this attack?

Send an application level challenge and wait for a response before sending the message.

(c.) A variant of spoofing is used to protect a high- bandwidth (and thus valuable) machine by forging the source of a lower bandwidth (less valuable) machine also under the attacker's control. The victim's return packets are then forwarded by the low-bandwidth host back to the high-bandwidth host which then can retrieve the correct sequence number.
Assuming the low bandwidth host is connected to a 56Kb/s link and the high bandwidth host has a 1Gb/s connection, how many messages per second can the high bandwidth server send (again use single packet e-mail messages of 1500bytes)?

Acks on the 56k link are the bottleneck in this attack. Since an ACK is generally 64 bytes

*56k/(64*8bits) = 109messages/sec*



(d.) Does your fix for part (d) above also apply to this form of spoofing?

In general, any solution that relied on the attacker not receiving the initial SYN|ACK does not work in this setup.

3. Physicists generate a few terabytes of data per day, which they would like to share with their colleagues in CERN. SLAC and CERN are connected with 10 Gb/s links end-to-end and have a round-trip time of 200 milliseconds. They use standard TCP in transferring the bulk file, but quickly discover they are not able to achieve a sustained 10 Gb/s from end to end. They consult the local networking expert, Alice, who conjectures that the problem is the way in which TCP responds to losses in the network. Let's first find out if Alice's conjecture is true.

(a.) Assuming a simplified model of TCP (ignore Slow-Start and assume TCP is in the AIMD mode) derive an approximate expression for TCP's average window-size as a function of the loss-probability, *p*.

The area under the sawtooth is $\left(\frac{1}{2} \cdot \frac{\hat{W}}{2} \cdot \frac{\hat{W}}{2}\right) + \left(\frac{\hat{W}}{2} \cdot \frac{\hat{W}}{2}\right) = \frac{3}{8}\hat{W}^2$*. The probability that a packet will be dropped is* $\frac{1}{\frac{3}{8}\hat{W}^2}$ *(1 drop per sawtooth).*

Solving for $\hat{W}$*, we see* $\hat{W} = \sqrt{\frac{8}{3p}}$*. The average window is equal to 3/4 of* $\hat{W}$*, which is* $\sqrt{\frac{3}{2p}}$*.*

(b.) Using your answer to part (a), what is the average number of round-trip times between loss events that TCP can tolerate so as to sustain an average window-size of *w* packets. Observe that the time interval between losses needs to increase as the "pipe" size increases.

Since the average window, w_a, is equal to 3/4 of $\hat{W}$, then we can see that $\hat{W}$ is equal to 4/3 of w_a. $\frac{\hat{W} \times RTT}{2}$ *is the time interval between loss events.*

Therefore, we need $\frac{2}{3} w_a \times RTTs$ *between loss events.*

(c.) Let's see what it takes to fill the pipe (i.e. achieve 10Gb/s). To achieve 10Gb/s, we need an average window size of $10Gb/s \times 200ms \div 1500bytes$ (we are assuming the packets are 1500bytes long). How low a loss-rate do we need for this to happen ?

The average window size is
$\frac{10 \times 10^9 \times 200 \times 10^{-3}}{1500 \times 8} = \frac{5 \times 10^5}{3} = 166666.67$ *packets. Using the equation from part (a), we can see that* $\sqrt{\frac{3}{2p}} = 166666.67$ *. Solving for p, we see that we need a $5.4x10^{-11}$ loss rate.*

(d.) Alice recommends Alice-TCP: an alternative to AIMD – and proposes that TCP be modified to use MIMD instead (Multiplicative Increase Multiplicative Decrease). She believes it will increase the throughput for bulk transfers. In particular, she proposes that when an acknowledgement is correctly received, Alice-TCP will increase the window size by additive constant *a* (*i.e.* $w \rightarrow w+a$, so that it increases by a *multiplicative* amount when the whole window has been acknowledged); when there is a loss, the window size is multiplied by factor (1-*b*), where *b*<1. Assuming that exactly one packet is lost each time the window size reaches $\hat{W}$, show that the average window size is given by: $w = \left(\frac{2-b}{2}\right)\hat{W}$.

The average window per cycle is equal to the sum of one half of the smallest window size and largest window size. The largest window equals ^W and the smallest window is equal the drop factor multiplied by ^W. Therefore,

$$w = \frac{1}{2}\left((1-b)\hat{W} + \hat{W}\right) = \left(\frac{(1-b)\hat{W} + \hat{W}}{2}\right) = \left(\frac{2-b}{2}\right)\hat{W}.$$

(e.) Alice argues that Alice-TCP is better. She says that if we want to sustain a particular average window size, the number of round-trip times between losses is independent of the pipe size. Is she correct? Explain your answer.

Show that for MIMD TCP, the following holds true:

$$pw_a = \frac{a}{b}\left(1 - \frac{b}{2}\right) =: p$$

So, the number of round-trips times between losses for a sustainable average window size, w_a, does not increase with the pipe size!

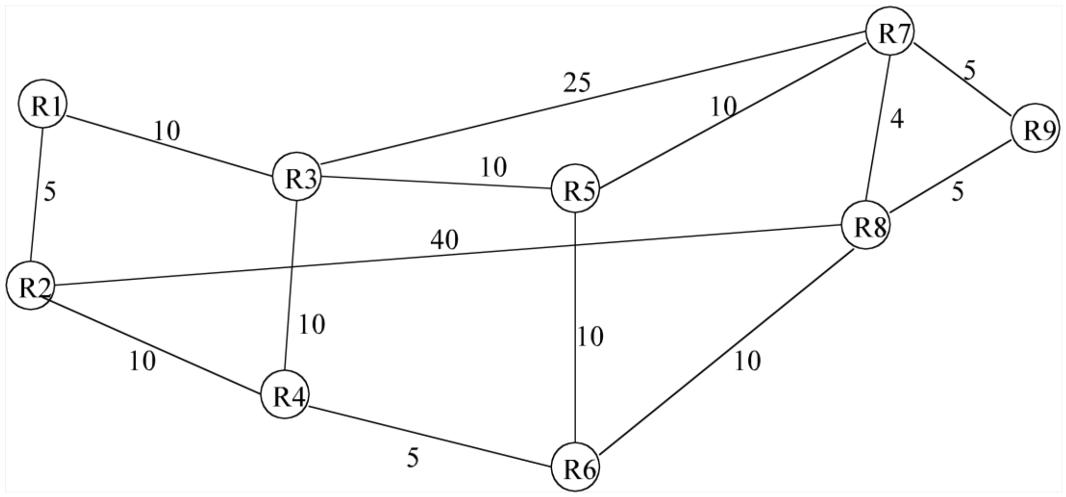
One way to do this would be to write a differential equation for rate change and equate it to 0 for the equilibrium.

$$\dot{w}_i = \frac{a w_i(t)}{T_i} - \frac{2b}{2-b} x_i(t) p_i(t) w_i(t)$$

where $x_i(t) := w_i(t)/T_i$. *Now equate equilibrium rate change* $\dot{x}_i(t)$ *to zero.*

An alternative way to do it would be to follow the procedure like Question 12 of the 2004 final.

4. In the following topology, assume all the links are bidirectional and the cost is the same for both directions.



(a.) Use the algorithm to find the lowest cost paths to R1 for all the nodes. For each step before convergence, show the current lowest cost to reach R1 and the next hop router to get there for each node.

Router	Hop	Dist	Hop	Dist	Hop	Dist	Hop	Dist	Hop	Dist
R_2	-	∞	*R_1*	*5*						

R_3	-	∞	*R_1*	*10*						
R_4	-	∞			*R_2*	*15*				
R_5	-	∞			*R_3*	*20*				
R_6	-	∞					*R_4*	*20*		
R_7	-	∞			*R_3*	*35*	*R_5*	*30*		
R_8	-	∞			*R_2*	*45*	*R_6*	*39*	*R_6*	*30*
R_9	-	∞					*R_7*	*40*	*R_7*	*35*

(b.) After the minimum cost spanning tree is formed, the link between R4 and R6 fails. For each step before re-convergence, and for each node, show the current lowest cost to reach R1 and the next hop router to get there. Assume split-horizon with poison reverse. Why do we have to assume it ?

Router	Hop	Dist	Hop	Dist	Hop	Dist	Hop	Dist	Hop	Dist
R_2	*R_1*	*5*								
R_3	*R_1*	*10*								
R_4	*R_2*	*15*								
R_5	*R_3*	*20*								
R_6	*R_4*	*20*	-	∞	*R_5*	*30*				
R_7	*R_5*	*30*								
R_8	*R_6*	*30*			-	∞	*R_7*	*34*		
R_9	*R_7*	*35*								

5. A router buffers arriving packets in separate queues – one per flow. Assume that the queues are all empty at time 0, and that the packets of each flow have a fixed size. Assume that flow *A*'s traffic is confined to a leaky-bucket constraint with $\sigma = 5$ packets, and $\rho = 10$ packets per second. The queues are all served using a WFQ scheduler, so as to meet an end-to-end delay guarantee.

(a.) What is the maximum backlog (in packets) at the queue corresponding to flow *A*?

5 packets

Packets of flow B follow a modified leaky-bucket model: The number of arrivals until time t is given by $a(0,t)!\ 5+\sqrt{t}$.

(b.) What is the maximum backlog at the queue corresponding to flow *B*?

5 packets

Packets of user C enter the router periodically. All we know is that in each 10 seconds, no more than 1000 packets arrive.

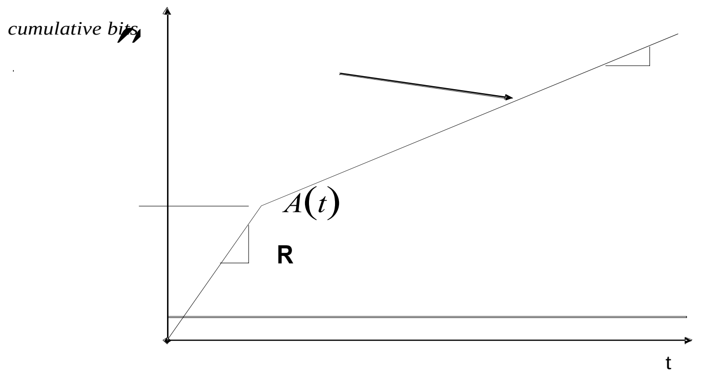
(c.) Can this traffic be modeled as leaky-bucket traffic? If so, what are the corresponding σ and ρ parameters? If not, explain why.

Yes. $\sigma=1000$ *and* $\rho=100$

(d.) At a given time, assume packet *p* of flow *A* is scheduled to leave the router before packet *q* of flow *B*. Is it possible that a future arriving packet changes the departure order of these packets? Why?

When any two packets have been scheduled, they have been placed in order of increasing Finishing Time (as determined by the WFQ scheduler). They then leave in order of Finishing Time. Therefore, once scheduled, their departure order is fixed.

6. In this question we'll assume that flowsconsist of a stream of bits rather than packets. In class, we saw how a leaky bucket regulator can constrain a flow so that the number of bits sent in an interval of duration
τ is given by: $A(t, t+\tau) \leq \sigma + \rho\tau$. If you think about it, this leaky bucket would allow a burst of up to σ bits to depart from the source in zero time (i.e. at an infinite rate)! This is clearly not possible; so in practice, the source is constrained by the data rate of its outgoing link. We'll assume that the flow is constrained by a leaky bucket at the source, and that a burst can depart no faster than *R* bits/second (the data rate of the link to which the source is connected). The figure below shows how the traffic is constrained. The flow passes through ten routers, all of which use bit-by-bit weighted fair queuing to serve



(a.) If the first router allocates the flow a service rate greater than or equal to c (where $\rho \le c$), write down expressions for the maximum delay of a bit in the router, and the size of the buffer needed to prevent overflows. (Write your answers in terms of σ, ρ and R, but not c).

$$\textit{Maximum buffer size} = ! - \frac{!''}{\mathrm{R}}, \textit{Delay} = \frac{! - \frac{!''}{\mathrm{R}}}{''} = \frac{!}{''} \# \frac{!}{R}$$

(b.) What happens to your answer in (a) when $R = \rho$? Explain your answer.

No buffer is needed

(c.) Why do you think it is customary to assume that a burst can leave a leaky bucket regulator at infinite rate?

R is usually much greater than r□so the answer changes very little when using R.