CS244
Advanced Topics in Computer Networks
Midterm Exam – Monday, May 2, 2016
OPEN BOOK, OPEN NOTES, INTERNET OFF

Your Name: Answers

SUNet ID: root @stanford.edu

In accordance with both the letter and the spirit of the Stanford Honor Code, I neither received nor provided any assistance on this exam.

Signature: _____

- The exam has 11 questions totaling 90 points.
- You have 75 minutes to complete them.
- Some questions may be much harder than others.
- All questions require you to justify your answer to receive full credit, even multiple choice questions for which you circle the correct answer(s).
- Keep your answers concise. We will deduct points for a correct answer that also includes incorrect or irrelevant information.

1	/5
2	/5
3	/5
4	/5
5	/5
6	/5
7	/12
8	/15
9	/9
10	/15
11	/9
Total	/90

1. **[5 points]:**

   PIM. If PIM runs to completion in each cell time (i.e. until no more connections can be added to the match), what is the maximum number of iterations required for an $N \times N$ switch?

   A $\log_2(N)$.

   B $N/2$.

   C N.

   D $\log_2(N) + 1$.

Justification:

2. **[5 points]:**

   PIM. The paper authored by Tom Anderson proves a lower bound on the expected fraction of unresolved requests that are resolved in each iteration. What is the bound?

   A 3/4.

   B 1/4.

   C 1/2.

   D None.

Justification:

3. [5 points]:

PIM We would like to find ways to make the PIM algorithm converge faster (i.e. in fewer iterations). The proof of the convergence time for the PIM algorithm starts by assuming that an output grants a requesting input that receives *no* other grants with probability k/n. Imagine that the algorithm is "tuned" to make it twice as likely to pick an input that receives *no* other grants, i.e. $2k/n$. What effect will this have on the convergence time?

A No effect.

B It will make it converge faster, because it expects to resolve at least 7/8 of unresolved requests in each iteration.

C It will make it converge faster, because it expects to resolve at least 3/8 of unresolved requests in each iteration.

D It will make it converge slower, because it expects to resolve at least 1/8 of unresolved requests in each iteration.

Justification:

4. **[5 points]:**

PGPS. Which of the following are true for Packetized Generalized Processor Sharing (PGPS)? Circle the correct option. Note that more than one option may be correct.

A PGPS serves packets in the increasing order of their packet lengths.

B PGPS provides the same average throughput guarantees for each flow as GPS.

C PGPS serves packets in decreasing order of their finishing time, which is computed based on GPS scheduling.

D PGPS can provide packets with an end-to-end delay guarantee across the network, but only if the source is constrained in some way, to prevent it sending in too much traffic.

Justification:

5. **[5 points]:**

Accountability. Which of the following assists in achieving the last goal in Clark's "Design Philosophy of the DARPA Internet Protocols" (The resources used in the internet architecture must be accountable)? Circle all that apply.

A Flow Groups and Tunnel Groups in B4

B Precise RTT measurement in TIMELY

C Leaky Bucket tokens in GPS

D Separation of control and data planes in SDN

Justification:

6. [5 points]:

Throughput. Consider an $N \times N$ input-queued switch carrying packets all of the *same* size. All links in the network operate at rate R. When packets arrive to the switch, it simply forwards the packet to a randomly picked output (i.e. it doesn't consult a lookup table based on the destination address). The output is picked uniformly and at random from all outputs. Which of the following statements are true?

A If each input maintains a single FIFO queue for all arriving packets, then the throughput can be limited to approximately 58% because of head-of-line blocking.

B If each input maintains a single FIFO queue, the input queues are always empty.

C If each input maintains virtual output queues (VOQs), the input queues are always empty.

D If each input maintains virtual output queues (VOQs), then the throughput is 100% for all arrival patterns.

Justification:

7. [12 points]:

Congestion control. Below is a list of congestion protocols.

A TCP Reno

B TIMELY

C QUIC

D TCP Reno with ECN—if the ECN congestion bit is marked in the packet ACK, perform multiplicative decrease with TCP Reno's congestion window.

For each of the following, write down the letter(s) of the congestion protocol(s) that has the corresponding feature. For example, if the feature is "packet modifications and window sizes adjusted at the sender," list A, B, C, and D, and justify briefly: "congestion signals trigger the size of the next window of packets sent by the sender."

a. Single-bit congestion signal

Answer:
A,C,D

b. Multi-bit congestion signal

Answer:
B

c. Records some measure of RTT

Answer:
A,B,C,D

d. In the single *lossless* link case, signals congestion only when switch buffer has reached capacity

Answer:
A,C

e. Packets modified at intermediate hops

Answer:
D

f. Implements loss recovery

Answer:
A,C,D

8. [15 points]:

Congestion control.

a. At the time Van Jacobson and Karels added congestion control to TCP, which of these mechanisms did TCP implementations use to detect lost segments? *Please circle T or F. No justification needed for this part.* **(1/2 point each)**

T F A timeout
T F Four acknowledgments asking for the same next byte
T F Selective acknowledgment of segments delivered while prior segments are missing
T F NACKs for segments still missing after later segments have been delivered
T F Explicit congestion notification from the network

Answer:
T, F, F, F, F (The timeout is the only mechanism to detect loss in "standard" TCP.)

b. Which of these mechanisms do current TCP implementations use to detect lost segments? *Please circle T or F. No justification needed for this part.* **(1/2 point each)**

T F A timeout
T F Four acknowledgments asking for the same next byte
T F Selective acknowledgment of segments delivered while prior segments are missing
T F NACKs for segments still missing after later segments have been delivered
T F Explicit congestion notification from the network

Answer:
T, T, T, F, T (Current TCP implementations have everything except NACKs.)

c. If the current loss-detection mechanisms had been in TCP from the start, do you think Van Jacobson and Karels would still have wanted to add congestion control to TCP? Give at least one reason why and why not. **(5 points)**

Answer:

Yes, they probably would have still wanted to add congestion control to TCP.

Reasons why not: *with selective acknowledgments, on a single bottleneck, TCP will almost never retransmit a segment that was going to make it to the receiver anyway. Therefore, "congestion collapse" cannot occur because the total goodput will not drop below the link capacity.*

Reasons why:

- *Congestion collapse can occur on multi-hop topologies, even with selective acknowledgment. (Packets can traverse one hop, displacing others, and then get dropped later. This means some of the capacity of the first link was wasted.)*
- *A TCP that's sending way too fast can cause huge loss rates, which can really hurt other applications on the same network that care about data arriving in a timely manner (a category that includes real-time applications that use UDP, but even plenty of TCP applications like SSH or Web browsers).*
- *Bufferbloat would be even worse if active-queue-management schemes were not able to slow down a TCP flow by dropping or marking a small fraction of segments.*

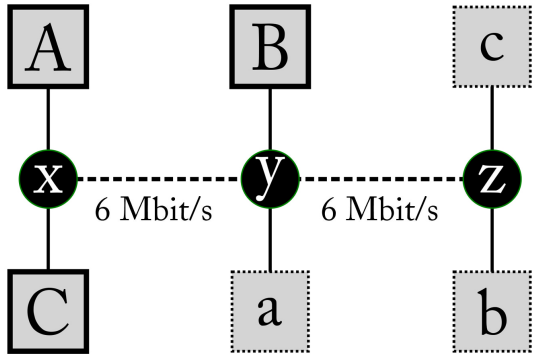
d. What assumptions about the network stack and application does the VJ+K algorithm seem to make? Which of these assumptions would you question today? **(5 points)**

Answer:

- **Assumption:** *when segments go missing, that indicates congestion and the need to reduce the number of packets in-flight at a given time (the congestion window). Worth questioning today, where wireless links might drop a packet because of low signal strength or natural interference.*
- **Assumption:** *cross traffic won't mind if a flow keeps increasing the number of packets in-flight until segments start going missing. The buffers inside the network are small enough that the delay will still be acceptable. Worth questioning today, when in-network queues can grow to be huge before dropping a packet (bufferbloat).*
- **Assumption:** *Congestion-control should be done separately for each byte stream. Even if a host has 10 connections open to 2 different hosts, all 10 connections will have separate congestion control and will have to ramp up from a small initial window on their own. Worth questioning today, when users typically have a single bottleneck and open many flows to the same host (arguing for unified congestion control with multiple streams, a la QUIC).*
- **Assumption:** *A packet is either lost or it isn't. If it's lost, retransmit it* and *reduce the congestion window. Worth questioning today, when the consequences of a false positive may be worse for one decision than the other. Perhaps hosts should reduce the congestion window on a hint of loss, but be prepared to revise their conclusions on new evidence.*
- **Assumption:** *An in-order byte stream is the primary abstraction of importance. Once an application writes a byte, that byte is irrevocable (even if it hasn't even been sent yet and the application wants to change its mind). Worth questioning today, when websites routinely use many streams in parallel, and some applications may decide that old data is not even worth sending if it hasn't made it there yet.*

9. [9 points]:

Flow-rate fairness. In the network below, sender A sends a flow to receiver a, sender B sends a flow to receiver b, and sender C sends a flow to receiver c. The link rate between x and y is 6 megabits per second, as is the link rate between y and z.



a. **(3 points)** What is the *max-min fair* allocation of throughputs to the three flows?

	$A \to a$	$B \to b$	$C \to c$
throughput (Mbit/s)			

Answer:

throughput (Mbit/s)	*3*	*3*	*3*

b. **(3 points)** What is the *max-utilization* allocation of throughputs to the three flows? (This allocation maximizes the total of the throughputs of the three flows.)

	$A \to a$	$B \to b$	$C \to c$
throughput (Mbit/s)			

Answer:

throughput (Mbit/s)	*6*	*6*	*0*

c. **(3 points)** What is the *proportionally fair* allocation of throughputs to the three flows? (This allocation maximizes the sum of the log of each flow's throughput.)

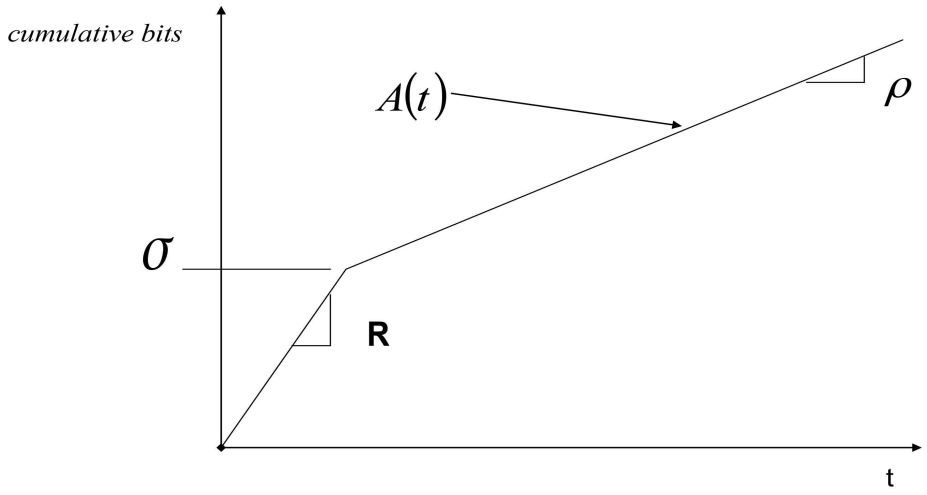
	$A \to a$	$B \to b$	$C \to c$
throughput (Mbit/s)			

Answer:

throughput (Mbit/s)	*4*	*4*	*2*

10. [15 points]:

Leaky-bucket constrained traffic. In this question we'll assume that flows consist of a stream of bits rather than packets. In the PGPS paper 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 shows how the traffic is constrained. The flow passes through multiple routers, all of which use bit-by-bit generalized processor sharing (GPS) to serve multiple flows.



a. (5 points) If the first router allocates the flow a service rate greater than or equal to some constant 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).

Answer:

$Maximumbuffersize = \sigma - \frac{\sigma\rho}{R}$ $Delay = \frac{\sigma - \frac{\sigma\rho}{R}}{\rho} = \sigma/\rho - \sigma/R$

b. **(5 points)** What happens to your answer above when $R = \rho$? Explain your answer.

Answer:

No buffer is needed.

c. **(5 points)** Why do think it is customary to assume that a burst can leave a leaky bucket regulator at infinite rate?

Answer:

R is typically much greater than r, so the answer changes very little when using R.

11. [9 points]:

TIMELY. TCP achieves max-min fair share rates per connection. That is, no flow could have a higher rate without decreasing the rate of an equal or smaller rate flow. B4 also uses max-min fairness for Traffic Engineering in assigning rates to flow groups.

(a) **(4 points)** Does TIMELY also achieve max-min fairness? Explain your reasoning.

Answer:

Yes. TIMELY is performing AIMD similar to TCP, just with a different congestion signal and multiplier. The multiplicative decrease gives greater penalty to flows with higher rates, who decrease their sending rate more aggressively than flows with lower rates.

(b) **(3 points)** Does bufferbloat occur to a similar degree in networks that use TIMELY compared to networks that use vanilla TCP? Explain your reasoning.

Answer:

No. The packet drop as a congestion signal results in bufferbloat. TIMELY strives to keep the queue occupancy low, since a standing queue results in increased RTT, which is the congestion signal for TIMELY.

(c) **(2 points)** What are the benefits of using RTT over a packet drop as a congestion signal?

Answer:

Multi-bit signal, reacts to congestion faster than a packet drop.