CS 168 Fall 2024 — Intro to the Internet: Architecture and Protocols — Final Exam

These solutions are in draft form, and likely contain errors.
Solutions last updated: December 9, 2024

Name: ______________________________

Student ID: ______________________________

This exam is 170 minutes long. There are 9 questions, worth 100 points.

In addition, Question 10 is a redemption question from the midterm (see Q10 for more details).

Question:	1	2	3	4	5	6	7	8	9	Total
Points:	18	14	11	13	16	11	5	7	5	100

For questions with **circular bubbles**, you may select only one choice.

- ○ Unselected option (completely unfilled)
- ● Only one selected option (completely filled)
- ✓ Don't do this (it will be graded as incorrect)

For questions with **square checkboxes**, you may select one or more choices.

- ■ You can select
- ■ multiple squares (completely filled)
- ☑ Don't do this (it will be graded as incorrect)

Anything you write outside the answer boxes or you ~~cross out~~ will not be graded. If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we will grade the worst interpretation.

Honor Code: Read the honor code below and sign your name.

I understand that I may not collaborate with anyone else on this exam, or cheat in any way. I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that academic misconduct will be reported to the Center for Student Conduct and may further result in, at minimum, negative points on the exam.

Sign your name: ______________________________

Q1 *Quick Questions* (18 points)

Each subpart is 1 point, unless otherwise specified.

Q1.1 When an error occurs, HTTP requires servers to send the exact correct error code.

○ True [✓ False]

Solution: False. HTTP has no requirements on how the error codes are used. In general, the goal of the error is to elicit the correct behavior from the client, not necessarily to pinpoint the exact error that occurred.

In the next 3 subparts, consider TCP with no congestion control, as implemented in the project.

Q1.2 Both active close and passive close require you to send a FIN packet.

[✓ True] ○ False

Solution: True. In passive close, the other side sends the first FIN, but you eventually have to send a FIN as well when you are done sending data.

In active close, you send the FIN first.

Q1.3 Suppose you set RWND (receiver window, `snd.wnd` in the project) to a very large value, instead of the value reported by the other side. What issue could occur?

○ You may run out of buffer space.

[✓ The other side may run out of buffer space.]

○ The other side will have fewer in-flight packets (from the other side to you).

○ The other side will have more in-flight packets (from the other side to you).

Solution: Remember that your value of RWND determines how much data you can send to the other side, based on the amount of buffer space the other side has left.

If your value of RWND is too large, you will send too much data to the other side, causing the other side to run out of buffer space.

Note that your value of RWND does not affect the data the other side is sending. (You will report how much buffer space you have, for the other side to adjust their window accordingly, but that's the other side's RWND value, not yours.)

Q1.4 Suppose you report a very large value of RWND to the other side. This causes the other side to set its RWND (`snd.wnd`) to a very large value. What issue could occur?

- [✓ You may run out of buffer space.]
- The other side may run out of buffer space.
- You will have fewer in-flight packets (from you to the other side).
- You will have more in-flight packets (from you to the other side).

Solution: If you report a very large value of RWND, you're telling the other side that they can send lots of data to you. This may cause you to run out of buffer space.

As before, note that the value of RWND that you report affects how much data the other side is sending. It does not affect how much data you are sending.

In the next two subparts, our goal is to add a cache to reduce the total amount of bandwidth used on the network (compared to when we had no cache).

Q1.5 A **private** cache only achieves our goal when a single client tries to access a resource multiple times.

[✓ True] False

Solution: True. If the client is accessing a resource for the first time, it won't be in the client's private cache, so no network bandwidth will be saved. On subsequent accesses, the client can retrieve the resource from the cache, saving bandwidth.

Q1.6 A **proxy** cache only achieves our goal when a single client tries to access a resource multiple times.

True [✓ False]

Solution: False. Multiple clients can share a proxy cache. If Client A and Client B are both using the same proxy cache, network bandwidth can be saved if Client A accesses the resource (adding it to the cache), and then Client B accesses the resource (now from the cache instead of the origin server).

Q1.7 In TLS, the TLS header is included as part of the TCP payload.

[✓ True] False

Solution: TLS lives at a higher layer than TCP, so the TCP payload includes the TLS header.

(As an analogy, TCP lives at a higher layer than IP, so the IP payload includes the TCP header.

Q1.8 Consider an application provider using a CDN. The provider uses application-level mapping to redirect clients to CDN servers.

The provider can always identify a single CDN server that is closest to the client, and redirect the client to that closest CDN server.

○ True [✓ ● False]

Solution: False. "Close" can be measured in several different ways, such as geographic distance, cost in the network topology, latency, etc., and sometimes the measurements are not perfect or can change over time.

Additionally, information about geographic distance and network topology may not be known to the application provider.

Q1.9 In host networking for datacenter servers, basic operations like checksum verification and fragmentation have been offloaded from software to hardware.

[✓ ● True] ○ False

Solution: True. This idea of offloading basic tasks from software to hardware is the key idea behind host networking.

Q1.10 In host networking, offloading is done for improved performance, not improved correctness.

[✓ ● True] ○ False

Solution: True. The operations we're doing are the same, but they are much faster in hardware than in software.

Q1.11 In RDMA, when a job is scheduled and a WQE (Work Queue Element) is created, the NIC (Network Interface Card) must stop what it is doing and process the job immediately.

○ True [✓ ● False]

Solution: False. The NIC has a queue of pending jobs, and does not need to process a queued job immediately.

Q1.12 In RDMA, when a job is completed and a CQE (Completion Queue Element) is created, the operating system must stop what it is doing and process the job immediately.

○ True [✓ False]

Solution: False. The completion queue holds notifications about completed jobs, and the completion notification does not need to processed immediately.

Q1.13 With SDN, individual routers no longer need the **data** plane.

○ True [✓ False]

Solution: False. With SDN, individual routers still need the data plane in order to forward packets.

Q1.14 With SDN, individual routers no longer need the **control** plane.

[✓ True] ○ False

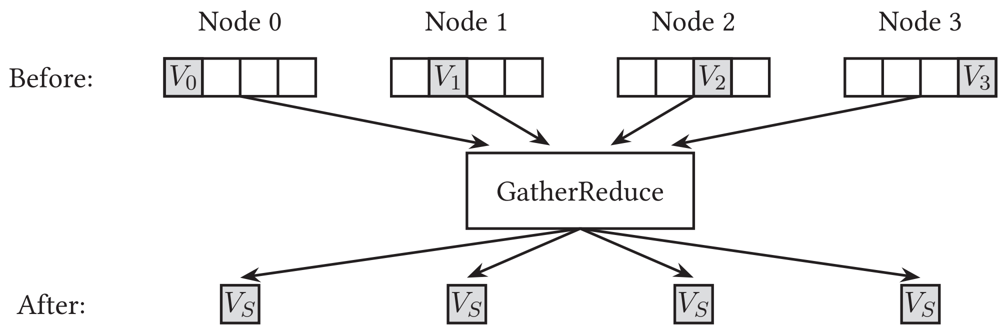
Solution: True. With SDN, individual routers no longer need the control plane to run routing algorithms. They can just be programmed directly by the SDN controller.

Q1.15 In a cellular network, two devices can have the same IMEI (device ID).

○ True [✓ False]

Solution: False. The IMEI is unique and burned into each device, so two devices can't share an IMEI.

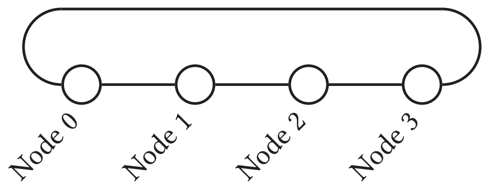
In the next two subparts, consider a new collective operation called GatherReduce. Using the representation from lecture, the GatherReduce operation is depicted as follows:



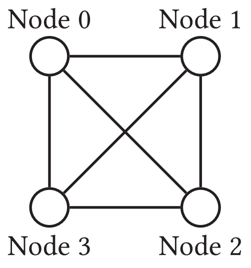
GatherReduce should compute $\boxed{V_S} = \boxed{V_0} + \boxed{V_1} + \boxed{V_2} + \boxed{V_3}$ and send this sum to each node.

Note: Each individual square, i.e. $\boxed{V_S}$, $\boxed{V_0}$, $\boxed{V_1}$, $\boxed{V_2}$, $\boxed{V_3}$, represents a vector of size D bytes.

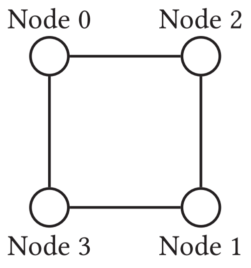
We have three topologies to choose from:



Topology A



Topology B



Topology C

We want to implement this collective, while satisfying the following efficiency constraint:
For each link X–Y, at most D bytes are sent from X to Y on this link, and at most D bytes are sent from Y to X on this link.

Q1.16 (2 points) Which topologies would allow us to implement the GatherReduce collective, while satisfying the efficiency constraint above? Select all that apply.

■ Topology A ■ Topology B ■ Topology C □ None

Solution:

All three topologies achieve the efficiency constraint, because they all have a ring path, i.e. a path from Node 0, to Node 1, to Node 2, to Node 3, back to Node 1.

With this ring path, we can implement GatherReduce while satisfying the efficiency constraint, like this:

Node 0 sends Vector 0 to Node 1.

Node 1 computes Vector 0 + Vector 1, and sends this sum to Node 2.

Node 2 takes the sum it received and computes (Vector 0 + Vector 1) + Vector 2, and sends this sum to Node 3.

Node 3 takes the sum it received and computes (Vector 0 + Vector 1 + Vector 2) + Vector 3. This is the final sum we want.

Node 3 takes the final sum and sends to Node 0. At this point, Node 0 realizes that the packet has made its way around the entire ring, so the computation is done.

At this point, Node 0 can send the final sum in the opposite direction around the ring.

Node 0 sends the final sum vector back to Node 3.

Then, Node 3 sends the final sum vector back to Node 2.

Finally, Node 2 sends the final sum vector back to Node 1.

We satisfied the efficiency constraint because a single sum-in-progress vector was sent in the forward direction (0-to-1, then 1-to-2, then 2-to-3, then 3-to-0). Then, a single final sum vector was sent in the backward direction (0-to-3, then 3-to-2, then 2-to-1, then 1-to-0). Each link is only used twice, once in each direction, which satisfies our constraint. Also, each message we send is a single vector of D bytes, which satisfies the constraint.

Q1.17 Out of the topologies you chose in the previous subpart (i.e. the topologies satisfying the constraint), which topology allows GatherReduce to complete in the fastest time?

Assume all links are identical (i.e. same bandwidth, same propagation delay).

○ Topology A [✓ Topology B] ○ Topology C

○ Multiple topologies achieve the same completion time.

Solution: Topology B can finish the operation in a single time step.

To finish the operation in one time step, each node floods its vector to all other nodes. Specifically:

- Node 0 sends its vector using the 0-to-1, and 0-to-2, and 0-to-3 links.
- Node 1 sends its vector using the 1-to-0, and 1-to-2, and 1-to-3 links.
- Node 2 sends its vector using the 2-to-0, and 2-to-1, and 2-to-3 links.
- Node 3 sends its vector using the 3-to-0, and 3-to-1, and 3-to-2 links.

At this point, everybody knows all four vectors, so everybody can independently compute the final sum vector, and no further sending of messages is required.

Note that every link was used twice, once in each direction. Also, note that every message sent was a single vector of D bytes. Therefore, this strategy satisfies the constraint.

Also, note that this one-time-step strategy cannot be achieved on Topology A or Topology C, because they are not fully-meshed. For example, in Topology A, it takes three time steps to send a packet from 0 to 3, and in Topology C, it takes two time steps to send a packet from 0 to 1.

Q2 *TCP Congestion Control* **(14 points)**

Consider two hosts communicating using the TCP congestion control algorithm seen in class. For the entire question, assume all TCP values are measured in packets (not bytes), unless otherwise specified.

Reminders:

- The hosts are using fast retransmit and fast recovery.
- The sender always has new data to send, and RWND is very large.
- The algorithm moves from Slow Start to Congestion Avoidance when CWND $>$ SSTHRESH.
- During Congestion Avoidance, for each new acked packet, CWND $\leftarrow$ CWND $+ \dfrac{1}{\text{CWND}}$.

Q2.1 (1 point) Immediately after the TCP handshake, what mode of TCP is active?

[✓ Slow Start] ○ Congestion Avoidance ○ Fast Recovery

Solution: By definition, the congestion control algorithm from class starts in Slow Start mode.

Q2.2 (1 point) Immediately after the TCP handshake, CWND = 1 packet.

After the 3-way TCP handshake, 3 distinct packets have been sent and 3 distinct, in-order acks have been received. At this point, what is the new value of CWND?

○ 1 ○ 2 ○ 3 [✓ 4] ○ 8 ○ 16

Solution: In slow start mode, our congestion window will double with every RTT by adding one to our CWND value with every new ack we receive. Therefore, our new CWND value is $1 + 3 = 4$.

The following subparts are independent of the earlier subparts.

Some time later, you are in Congestion Avoidance mode, with the following settings:

- CWND = 12, SSTHRESH = 16
- All packets up to, and including packet #60, have been sent and acked.
- Packets #61 through #72, inclusive, have been sent, but not acked.
- All packets #73 and later have not been sent.

Q2.3 (1 point) For this subpart only, assume there is infinite bandwidth available in the network.

Suppose you receive distinct, in-order acks for packets #61, #62, #63, etc., and you send packets #73, #74, #75, etc. This continues for a long time, with no timeouts or duplicate acks.

As you continue to receive acks and send out more packets indefinitely, what happens?

○ As soon as CWND exceeds SSTHRESH, you switch to Slow Start mode.

○ CWND reaches SSTHRESH = 16, and stops increasing after that.

[✓ CWND increases indefinitely, with no limit, but SSTHRESH doesn't change.]

○ As soon as CWND exceeds SSTHRESH, both CWND and SSTHRESH increase indefinitely.

Solution: The only way to leave congestion avoidance mode is receiving 3 duplicate acks or a timeout. If neither of those occur, then CWND would continue increasing indefinitely.

Note that SSTHRESH only changes when we encounter a loss (detected via duplicate ack or timeout). Since neither of those occur, SSTHRESH never changes.

Now, suppose that only packet #66 is dropped in transit. All other packets are successfully sent, and you receive distinct, in-order acks for all other packets, with no timeouts.

Q2.4 (1 point) In this scenario, TCP would eventually switch out of Congestion Avoidance mode, and into which mode?

○ Slow Start [✓ Fast Recovery]

Solution: You would receive 3 duplicate acks, causing you to transition into Fast Recovery mode.

Since there are no timeouts, we won't transition into Slow Start next.

Q2.5 (2 points) What is the approximate value of CWND the instant **before** TCP switches out of Congestion Avoidance mode? (Hint: You don't need to do any complicated arithmetic here.)

○ 1　[✓ $12\frac{5}{12}$]　○ 16　○ 18　○ 384

○ $12\frac{4}{12}$　○ $12\frac{6}{12}$　○ 17　○ 192　○ 768

Solution: In Congestion Avoidance mode, every new ack increases CWND by roughly 1/CWND.

Since CWND is currently 12, each new ack will increase CWND by roughly 1/12.

You get new, distinct acks for #61, #62, #63, #64, and #65. These 5 packets increase CWND by 5/12 to a total of $12\frac{5}{12}$.

Note that the ack you receive for #65 says ack(66), since that's the next packet the receiver expects to receive. Then, the next ack you receive, for packet #67, will also say ack(66), which is the first duplicate ack. Duplicate acks do not change CWND. The next few acks will also say ack(66), so they won't change CWND either.

TODO: Do students know that dupacks do not change CWND? We verified that they don't in the RFC, but wanted to check if students know.

Q2.6 (2 points) What is the value of CWND the instant **after** TCP switches out of Congestion Avoidance mode, and into a different mode? Round your answer to the nearest integer.

○ 1　○ 6　○ 8　[✓ 9]　○ 11　○ 12

Solution: After the third duplicate ack, CWND is cut in half from $12\frac{5}{12}$ to $6\frac{5}{24}$.

However, since we're entering fast-recovery mode, we'll temporarily give credit for the 3 acked packets (corresponding to the 3 duplicate acks), so we'll increase CWND by 3, to $9\frac{5}{24}$.

Rounding this answer to the nearest integer gives a CWND of 9.

Q2.7 (1 point) Which of these ack numbers, if received, would cause you to leave the different mode that you switched to?

- ○ An ack number less than 65
- ○ 66
- ○ 65
- [✓ An ack number greater than 66]

Solution: Receiving a new ack causes you to leave Fast Recovery mode. You're currently stuck in Fast Recovery mode because you received duplicate acks all saying 66, so an ack greater than 66 would allow you to leave Fast Recovery mode.

The following subparts are independent of the earlier subparts.

Some time later, you are in Congestion Avoidance mode, with the following settings:

- CWND = 6, SSTHRESH = 8
- All packets up to, and including packet #202, have been sent and acked.
- Packets #203 through #208, inclusive, have been sent, but not acked.
- All packets #209 and later have not been sent.

At this point, the sender stops receiving any further acks.

Q2.8 (1 point) In this scenario, TCP would eventually switch out of Congestion Avoidance mode, and into which mode?

[✓ Slow Start] ○ Fast Recovery

Solution: Because you stop receiving any acks, you'll eventually time out and move back into Slow Start.

Q2.9 (1 point) What is the value of CWND the instant after TCP switches out of Congestion Avoidance mode, and into a different mode?

[✓ 1] ○ 2 ○ 3 ○ 4 ○ 6 ○ 8

Solution: After a timeout, CWND is always reset to 1 packet.

Q2.10 (1 point) What is the packet sent as a result of switching to a different mode?

○ #202 [✓ #203] ○ #208 ○ #209

Solution: The first unacked packet is #203, so the timeout causes us to resend that.

The following subparts are independent of the earlier subparts.

Some time later, the TCP connection has the following settings:

- CWND $= 4$, SSTHRESH $= 8$
- All packets up to, and including packet #411, have been sent and acked.
- Packets #412 through #415, inclusive, have been sent, but not acked.
- All packets #416 and later have not been sent.

For the next two subparts, you can assume CWND is rounded to the nearest integer when deciding which packets to send out.

The next two subparts are independent (i.e. Q2.12 is not continuing from Q2.11).

Q2.11 (1 point) If we are in **Congestion Avoidance** mode, and we receive an ack for packet #412, which packet(s) do we send out as a result? Select all that apply.

[✓ #416] ☐ #417 ☐ #418 ☐ #419 ☐ None

Solution: When you receive a new ack for packet #412, CWND increases to $4 + \frac{1}{4}$, which rounds to 4.

We can have 4 packets in transit. Packets #413, #414, and #415 are still in transit. We can additionally send #416 to keep 4 packets in transit.

Q2.12 (1 point) If we are in **Slow Start** mode, and we receive an ack for packet #412, which packet(s) do we send out as a result? Select all that apply.

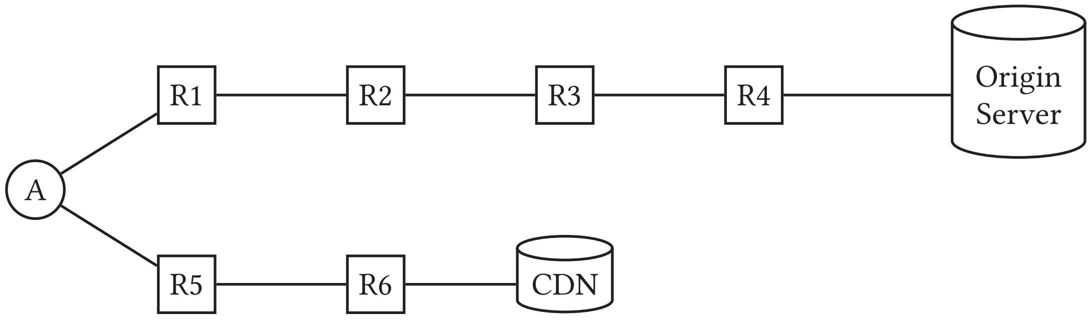
[✓ #416] [✓ #417] ☐ #418 ☐ #419 ☐ None

Solution: When you receive a new ack for packet #412, CWND increases to $4 + 1 = 5$.

We can have 5 packets in transit. Packets #413, #414, and #415 are still in transit. We can additionally send #416 and #417 for a total of 5 packets in transit.

Q3 *TCP Performance* **(11 points)**

Consider accessing an application using a CDN server, as shown below:



Unless otherwise specified, assume that:

- Each link has the same propagation delay. For example, the delay from A to the origin server is 5/3 times higher than the delay from A to the CDN server.
- Each link uses the standard MTU of 1500 bytes.
- Each link has a packet drop rate of 0.

In the next three subparts, consider the following two scenarios for downloading the same large file:

- **Scenario 1:** Host A downloads the entire file from the origin server.
- **Scenario 2:** Host A downloads the entire file from the CDN server.

Let S_1 be the throughput for Scenario 1, and S_2 be the throughput for Scenario 2.

According to the TCP throughput equation (on the reference card) and the assumptions in each subpart, select the equation that correctly relates the throughput of S_1 and S_2.

Q3.1 (2 points) For this subpart only: The A-to-R1 and A-to-R5 links each have a drop rate of 0.25, and all other links have a drop rate of 0. For example, a connection from A to the CDN has an overall packet drop rate of 0.25.

○ $S_1 = \frac{5}{3} \cdot S_2$ ○ $S_1 = 2 \cdot S_2$ ○ $S_2 = \sqrt{\frac{5}{3}} \cdot S_1$

○ $S_1 = \sqrt{\frac{5}{3}} \cdot S_2$ [✓ $S_2 = \frac{5}{3} \cdot S_1$] ○ $S_1 = S_2$

Solution: Scenario II (CDN server) is faster by a factor of 5/3.

In the TCP throughput equation, MSS is the same in both scenarios (using the same default MTU in both scenarios). Also, the packet loss rate is the same (0.25) in both scenarios.

The only term that varies is the RTT. In Scenario I (origin server), the RTT is 10 links (5 each way), so the TCP throughput equation results in a bunch of constants times $1/10$.

In Scenario II (CDN server), the RTT is 6 links (3 each way), so the TCP throughput equation results in a bunch of constants times $1/6$.

$1/6 > 1/10$, so Scenario II (CDN server) has the higher throughput. To see how much higher, we can divide the Scenario II throughput by the Scenario I throughput. All constant terms cancel, so we get:

$$\frac{\text{Scenario II Throughput}}{\text{Scenario I Throughput}} = \frac{1/6}{1/10} = \frac{10}{6} = \frac{5}{3}$$

Q3.2 (2 points) For this subpart only: The A-to-R1 and A-to-R5 links each have a drop rate of 0.25, and all other links have a drop rate of 0.

Also, for this subpart only: the MTU for all links between A and the origin server is increased to 3000 bytes. All other links have an MTU of 1500 bytes.

Assume the header size is negligible, i.e. you can use the MTU as an approximation for the MSS.

- ○ $S_1 = \frac{3}{2} \cdot S_2$
- [✓ $S_1 = \frac{6}{5} \cdot S_2$]
- ○ $S_2 = \frac{3}{2} \cdot S_1$
- ○ $S_1 = 2 \cdot S_2$
- ○ $S_2 = \frac{6}{5} \cdot S_1$
- ○ $S_1 = S_2$

Solution:

Scenario I (origin server) is faster by a factor of $\frac{6}{5}$.

In the TCP throughput equation, MSS is now 3000 in Scenario I, and 1500 in Scenario I. The packet loss rate is the same (0.25) in both scenarios. As in the previous subpart, the RTT is different in the two scenarios.

In Scenario I (origin server), the RTT is 10 links (5 each way), so the TCP throughput equation results in a bunch of constants times $1/10$, times 3000. ($1/10 \times 3000 = 300$.)

In Scenario II (CDN server), the RTT is 6 links (3 each way), so the TCP throughput equation results in a bunch of constants times $1/6$, times 1500. ($1/6 \times 1500 = 250$.)

$300 > 250$, so Scenario I (origin server) has the higher throughput. To see how much higher, we can divide the Scenario I throughput by the Scenario II throughput. All constant terms cancel, so we get:

$$\frac{\text{Scenario I Throughput}}{\text{Scenario II Throughput}} = \frac{300}{250} = \frac{6}{5}$$

Q3.3 (2 points) For this subpart only: The A-to-R1 link has a drop rate of 0.04, the A-to-R5 link has a drop rate of 0.25, and all other links have a drop rate of 0.

○ $S_1 = \sqrt{\frac{3}{2}} \cdot S_2$ ○ $S_1 = \frac{8}{3} \cdot S_2$ ○ $S_2 = \frac{8}{3} \cdot S_1$

[✓ $S_1 = \frac{3}{2} \cdot S_2$] ○ $S_2 = \frac{3}{2} \cdot S_1$ ○ $S_1 = S_2$

Solution: Scenario I (origin server) is faster by a factor of 3/2.

Now, in the TCP throughput equation, RTT and p both vary, although MSS still stays the same.

In Scenario I, RTT is 10 links, and p is 0.04, so the TCP throughput equation results in a bunch of constants times $\frac{1}{10\sqrt{0.04}} = \frac{1}{2}$.

In Scenario II, RTT is 6 links, and p is 0.25, so the TCP throughput equation results in a bunch of constants times $\frac{1}{6\sqrt{0.25}} = \frac{1}{3}$.

$1/2 > 1/3$, so Scenario I (origin server) has the higher throughput. To see how much higher, we can divide the Scenario I throughput by the Scenario II throughput. All constant terms cancel, so we get:

$$\frac{\text{Scenario I Throughput}}{\text{Scenario II Throughput}} = \frac{1/2}{1/3} = \frac{3}{2}$$

Q3.4 (2 points) For this subpart only: The A-to-R1 link has a drop rate that you will determine, the A-to-R5 link has a drop rate of 0.25, and all other links have a drop rate of 0.

What should the A-to-R1 link drop rate be, so that the throughput in the two scenarios ends up equaling the same?

○ 0.3 [✓ 0.09] ○ 0.25 ○ 0.5 ○ 0.64 ○ 0.81

Solution: Let p be the A-to-R1 link drop rate. We will solve for p in this question.

In Scenario I, RTT is 10 links, and p is unknown, so the TCP throughput equation results in a bunch of constants times $\frac{1}{10\sqrt{p}}$.

In Scenario II, RTT is 6 links, and p is 0.25, so the TCP throughput equation results in a bunch of constants times $\frac{1}{6\sqrt{0.25}} = \frac{1}{3}$.

We want these two values to be equal, so we set:

$$\begin{aligned}\frac{1}{10\sqrt{p}} &= \frac{1}{3}\\ 10\sqrt{p} &= 3\\ \sqrt{p} &= 0.3\\ p &= 0.09\end{aligned}$$

Q3.5 (2 points) Consider a scenario where, for every user, downloading from the origin server is always faster than downloading from any CDN server.

In this scenario, are there any advantages to deploying the CDN?

If you say Yes, name an advantage. If you say No, explain why not. Answer in 10 words or fewer.

[✓ Yes] ○ No

Solution: Some possible answers:

Redundancy: If a server goes down, the content can be served by a different server.

Load balancing: CDNs reduce load on the origin server.

Reduced bandwidth: Installing CDNs closer to the end users reduces the amount of bandwidth the network needs to support.

Q3.6 (1 point) Suppose there are many CDN servers in the network, and the application provider uses anycast to redirect users to the closest CDN server.

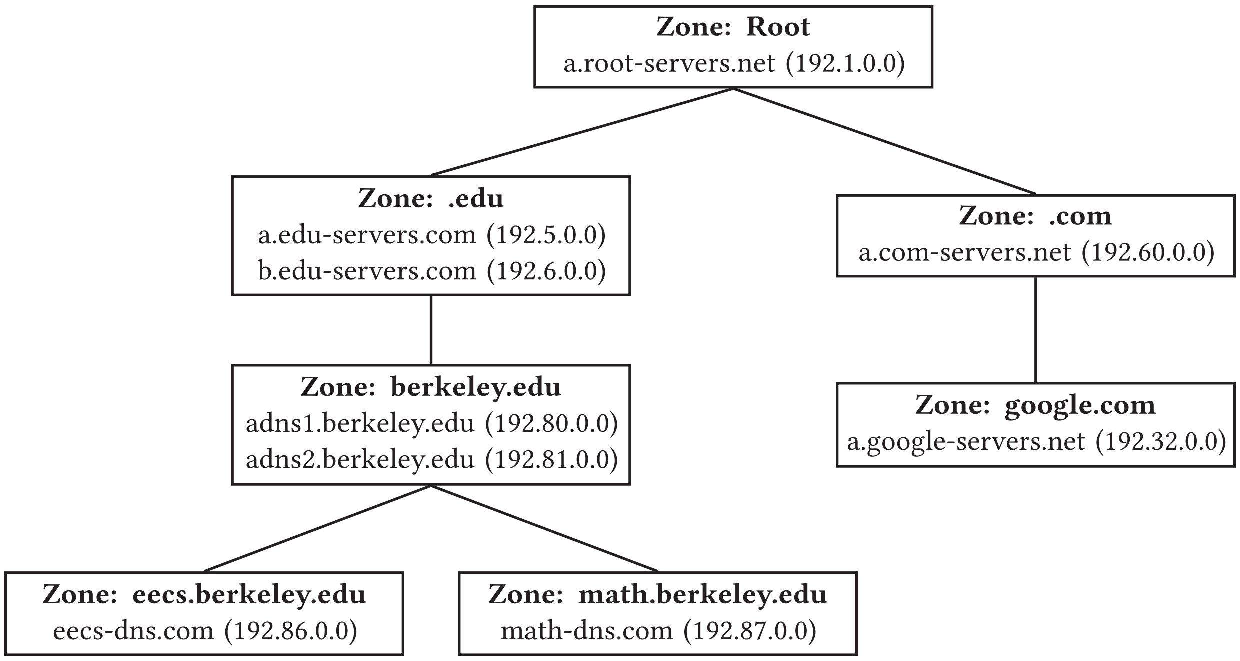
Why might anycast cause issues in this scenario?

- ○ Most links have a drop rate of 0.
- ○ Links use a standard MTU of 1500 bytes.
- ○ Each link has the same propagation delay.
- [✓ The file we're downloading is large.]

Solution: As mentioned in lecture, anycast with CDNs can cause issues because a TCP connection could suddenly get re-routed to a different server that doesn't know about the TCP connection. This is especially an issue with CDNs (compared to other anycast applications like DNS), because the files we're downloading can be large, leading to long-running TCP connections.

Q4 *DNS* (13 points)

Consider the following DNS name server hierarchy. Each node represents a zone. Each node contains the domains and corresponding IP addresses for all name servers that are authoritative for that zone.


Zone: Root
a.root-servers.net (192.1.0.0)
Zone: .edu
a.edu-servers.com (192.5.0.0)
b.edu-servers.com (192.6.0.0)
Zone: .com
a.com-servers.net (192.60.0.0)
Zone: berkeley.edu
adns1.berkeley.edu (192.80.0.0)
adns2.berkeley.edu (192.81.0.0)
Zone: google.com
a.google-servers.net (192.32.0.0)
Zone: eecs.berkeley.edu
eecs-dns.com (192.86.0.0)
Zone: math.berkeley.edu
math-dns.com (192.87.0.0)

Host A and Host B both use the recursive resolver with IP address 8.8.8.8.

Q4.1 (1 point) What protocol helps Host A and Host B learn the IP address of the recursive resolver?

○ ARP [✓ DHCP] ○ NAT ○ TLS

Solution: DHCP is used to configure the host when it first joins the network. The configuration includes the IP address of a DNS resolver.

For the next 3 subparts, assume all DNS caches start empty, and no packets are dropped.

Also, assume that the recursive resolver has hard-coded the domain and IP address of the root name server. (These hard-coded values do not count as a record in the cache.)

Q4.2 (2 points) Host A wants to learn the IP address of people.eecs.berkeley.edu.

In total, how many DNS packets are sent over the network to answer this query?

○ 0 ○ 2 ○ 6 ○ 8 [✓ 10] ○ 12

Solution:

A to recursive resolver.

Recursive resolver to root.

Root to recursive resolver.

Recursive resolver to edu.

edu to recursive resolver.

Recursive resolver to berkeley.edu.

berkeley.edu to recursive resolver.

Recursive resolver to eecs.berkeley.edu.

eecs.berkeley.edu to recursive resolver.

Recursive resolver to A.

Q4.3 (2 points) Continuing from the previous part (i.e. caches are not cleared): Host B wants to learn the IP address of people.math.berkeley.edu.

In total, how many DNS packets are sent over the network to answer this query?

○ 0 ○ 2 [✓ 6] ○ 8 ○ 10 ○ 12

Solution:

B to recursive resolver.

Recursive resolver to berkeley.edu.

berkeley.edu to recursive resolver.

Recursive resolver to math.berkeley.edu.

math.berkeley.edu to recursive resolver.

Recursive resolver to B.

Q4.4 (2 points) For this subpart only, assume that name servers include information about *all* name servers in the next zone. For example, root will reply with information about both `.edu` name servers.

Also, assume that people.eecs.berkeley.edu and people.math.berkeley.edu each have one IP address.

Remember that it takes two records (an NS record and an A record) to provide information about a name server.

After both queries from earlier are performed, how many records are in the recursive resolver's cache?

○ 5 ○ 7 ○ 10 [✓ 14] ○ 22

Solution:

4 records for the .edu zone.

4 records for the berkeley.edu zone.

2 records for the eecs.berkeley.edu zone.

2 records for the math.berkeley.edu zone.

1 record for the IP address of people.eecs.berkeley.edu.

1 record for the IP address of people.math.berkeley.edu.

Q4.5 (1 point) Suppose that people.eecs.berkeley.edu has two IP addresses, an IPv4 address and an IPv6 address. Does this change your answer to Q4.4?

○ Yes, the cache has an additional A record.

[✓ Yes, the cache has an additional AAAA record.]

○ No, because IPv6 is newer, so we can discard the IPv4 record.

○ No, because IPv4 is more widely-adopted, so we can discard the IPv6 record.

Solution: AAAA records store IPv6 addresses.

Note that in DNS, we cache every record we receive.

Q4.6 (3 points) Which of the following name servers, if directly queried, can give the A record with the IP address of classes.berkeley.edu? Select all that apply.

- ☐ a.root-servers.net
- ☐ a.edu-servers.com
- ☐ b.edu-servers.com
- [✓ adns1.berkeley.edu]
- [✓ adns2.berkeley.edu]
- ☐ eecs-dns.com
- ☐ math-dns.com
- ☐ None of the above

Solution:

These are the two name servers responsible for the berkeley.edu zone, which contains www.berkeley.edu.

Note that the eecs.berkeley.edu and math.berkeley.edu name servers cannot serve www.berkeley.edu, because www.berkeley.edu is not a subdomain of eecs.berkeley.edu and math.berkeley.edu.

Also, note that the .edu name servers cannot serve www.berkeley.edu, because they have delegated authority over the entire berkeley.edu zone to the berkeley.edu name server.

Q4.7 (1 point) There are two IP addresses (192.5.0.0 and 192.6.0.0) for the .edu zone's name server(s).

True or False: This means that the .edu zone **must not** be using anycast.

○ True [✓ False]

Solution: False. There could be multiple machines advertising themselves as 192.5.0.0, and also multiple machines advertising themselves as 192.6.0.0. In fact, this is what the root zone does (each of the 13 root IP addresses is advertised by many machines).

Q4.8 (1 point) There is only a single IP address (192.87.0.0) for the math.berkeley.edu zone's name server(s).

True or False: This means that the math.berkeley.edu zone **must** be using anycast.

○ True [✓ False]

Solution: False. There might only be a single server advertising itself as 192.87.0.0. For example, this might be the case on smaller zones in real life.

Q5 *End-to-End* (16 points)

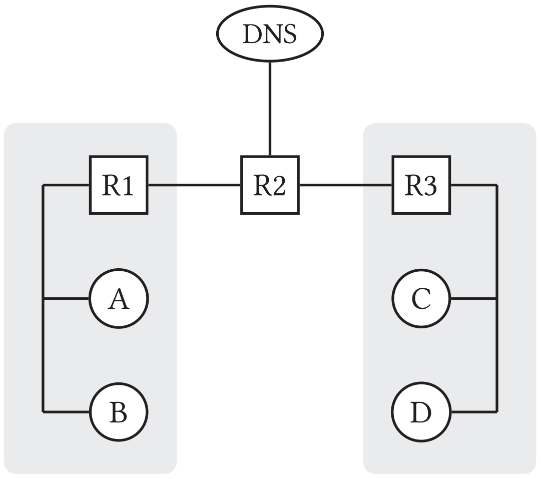
In this question, consider the network topology to the right.

A, B, and R1 are all in the same subnet, and are connected on a single shared medium (also called a bus).

Likewise, C, D, and R3 are all in the same subnet (different from the above subnet), and are connected on a single shared medium.

There is one DNS resolver in the network, labeled DNS. Assume everyone uses the same DNS resolver.

In this question, each subpart continues on from the previous one.



Q5.1 (2 points) Host A joins the network for the first time, with all caches empty and no active connections.

Host A broadcasts a DHCP Discover request. R1 sends a DHCP Offer reply containing some IP addresses. Host A accepts this offer.

Which IP addresses were contained in R1's DHCP Offer? Select all that apply.

■ R1's IP ☐ R2's IP ■ A's IP ☐ B's IP ■ DNS's IP

Solution: The offer contains R1's IP address, because this is the address of the next-hop router.

The offer also contains A's IP address, because this is the IP address assigned to A.

The offer also contains the IP address of the DNS resolver.

Q5.2 (2 points) Host A now wants to send an HTTP request to Host C.

Before any further steps, select all fields that Host A currently knows how to fill in.

■ Source IP　　☐ Destination MAC　　☐ None of the above

☐ Destination IP　　■ Source port

■ Source MAC　　■ Destination port

Solution:

Host A knows its own IP address from DHCP, so it can fill in the source IP field.

Host A does not know Host C's IP address, so it cannot fill in the destination IP field yet.

Host A knows its own source MAC address (burned in).

Host A does not know the destination MAC address. C could be local, in which case this field would be C's MAC address (learned from a future ARP request), or C could be non-local, in which case this field would be the router's MAC address (learned from a future ARP request).

Host A could pick any random source port.

Host A knows the destination port is 80 for HTTP, because this port is well-known by everybody.

Q5.3 (2 points) From the previous subpart, there are some fields that Host A currently does not know how to fill in.

Select all additional protocols that Host A needs to run in order to fill in those missing fields.

■ ARP　　☐ DHCP　　■ DNS　　☐ TLS　　☐ None

Solution:

DNS is needed to fill in the destination IP address.

ARP is needed to fill in the destination MAC address.

Eventually, A is able to fill in all fields, and sends a packet to C. The packet arrives at C.

Q5.4 (2 points) Select all devices that can deduce (possibly by reading packets they are not supposed to be reading) that Host A's IP address and Host C's IP address are communicating.

■ B ■ R1 ■ R2 ■ R3 ■ D □ None

Solution: When A forwards the packet, everyone on the shared medium (A, B, R1) sees the packet.

R2 sees the packet because it has to forward the packet from R1 to R3.

When R3 forwards the packet to C, everyone on the shared medium (R3, C, D) sees the packet.

Q5.5 (2 points) At this point, which of these IP-to-MAC mappings exists in Host A's cache? Assume no entries have expired, and nobody has sent an unsolicited ARP message. Select all that apply.

□ C's IP to C's MAC □ DNS's IP to DNS's MAC □ R2's IP to R1's MAC

■ C's IP to R1's MAC ■ DNS's IP to R1's MAC □ B's IP to B's MAC

Solution:

Host A had to send a packet to the DNS resolver, so it needs an entry for the DNS resolver's IP-to-MAC mapping.

Host A also sent a packet to C, so it needs an entry for C's IP-to-MAC mapping.

Since C and the DNS resolver are both non-local, their IPs map to R1's MAC, because Host A should forward non-local packets to R1.

At this point, Host A leaves and re-joins the network, which means that all of Host A's caches have been cleared.

Q5.6 (1 point) After Host A re-joins the network, does Host A still have the same IP address?

○ Always [✓ Sometimes] ○ Never

Solution:

When Host A re-joins, it needs to perform another DHCP handshake.

It's possible that Host A is offered the same IP it was previously using. It's also possible that Host A is offered a different IP this time.

For the rest of the question, assume that R1 is using NAT with Port Address Translation, as shown in lecture. Everyone in R1's subnet is using R1's public IP address, and A and B are assigned private IP addresses.

Q5.7 (1 point) Host A wants to communicate with Host C again.

At some point, Host A performs a DNS lookup for Host C's IP address.

Does R1 need to perform NAT translation on the DNS request packet?

- [✓ Yes, because Host A's IP address is private.]
- ○ Yes, because in NAT, the border router always modifies the source port for outgoing packets.
- ○ No, because DNS messages fit in a single UDP packet, so no long-term connection is formed.
- ○ No, because the DNS server's IP address is public.

Solution:

Option (B) is false, because NAT does not always rewrite the port.

Option (C) is false, because UDP packets still need the private IP replaced with a public IP.

Option (D) is false, because Host A's private IP still needs to be rewritten to a public IP so that the DNS server can reply.

Q5.8 (2 points) If Host A wants to send a packet to Host C, what additional steps does **Host A** need to do, now that NAT is enabled? Select all that apply.

- ☐ Choose a source port number that nobody else in the subnet is using.
- ☐ Choose a destination port number that nobody else in the subnet is using.
- ☐ Write R1's IP address in the source IP field.
- ☐ Write R1's IP address in the destination IP field.
- [✓ None of the above]

Solution: NAT is designed to give inside hosts the illusion that they have a dedicated IP address. Host A should never need to know that NAT is being used, so Host A does not need to do any extra steps.

Q5.9 (2 points) Now that NAT is enabled, select all devices that can deduce (possibly by reading packets they are not supposed to be reading) that Host A's IP address and Host C's IP address are communicating.

■ B ■ R1 ☐ R2 ☐ R3 ☐ DNS ☐ None

Solution:

Host A creates a packet that says "From A, To C." This packet gets broadcast on the shared medium, which means B receives this packet and can deduce that Host A and Host C are communicating.

R1 has the NAT table, so it knows that Host A and Host C are communicating.

R2 and R3 see a packet that says "From R1, To C," so it cannot deduce that Host A is the one communicating. From R2 and R3's perspective, either Host A or Host B could be communicating with Host C.

The DNS server receives a request "From R1, To DNS" about the IP address of C. The DNS server never learns about Host A, so from its perspective, either Host A or Host B could have made that request.

Q6 *Datacenters* (11 points)

Q6.1 (2 points) What are some advantages of using a Fat-tree Clos topology (i.e. a topology with pods, edge/aggregation layers, and core switches) instead of a Fat-tree topology? Select all that apply.

- ■ Links can be lower bandwidth.
- □ Bisection bandwidth is higher.
- ■ Avoids single point of failure.
- ■ Can be built out of commodity switches.
- □ None of the above

Solution:

Option (A) is true. In the fat-tree topology, the links near the top need to have higher bandwidth. In the Clos topology, all links can have the same lower bandwidth.

Option (B) is false. Both topologies can be built to achieve full bisection bandwidth.

Option (C) is true. There are multiple links between two hosts in a Clos topology, as well as multiple ways to travel from one host to another, so if one path ends up failing, there are other paths a packet can take to travel to its destination.

Option (D) is true. Unlike in the fat-tree topology, the Clos topology allows switches to be cheap, commodity ones, allowing for lesser cost.

In the rest of this question, we'll extend Equal Cost Multi-Path (ECMP) routing so that a single elephant TCP flow can travel along two different paths in the network.

Q6.2 (1 point) Does the unmodified ECMP protocol from lecture allow packets in a single TCP flow to be split across two different paths?

- ○ Yes, because we include destination IP as an input to the hash.
- ○ Yes, because we include source IP as an input to the hash.
- ● No, because all packets in a single TCP flow all have the same 5-tuple.
- ○ No, because all packets in a single TCP flow have the same source/destination IP (but not necessarily the same source/destination port).

Solution: Since all packets in a single flow have the same source/destination IP, source/destination port, and protocol, they will all get hashed to the same output link.

Q6.3 (1 point) What is a disadvantage of splitting a single elephant TCP flow across different paths, compared to sending the entire flow along a single path?

- [✓ Packets that travel along different paths might experience different delays, and TCP performs poorly when packets arrive out-of-order.]
- ○ There is significant additional overhead in finding all routes to a destination. ECMP does have not this overhead.
- ○ There is significant additional overhead in performing an additional TCP handshake to start the connection.
- ○ There is significant additional overhead for routers to forward packets along two different paths. ECMP does not have this overhead.

Solution:

Option (A) is the best choice here. Packets may arrive out-of-order because they are sent along different paths, which requires the receiver to buffer out-of-order packets. This causes the receiver to perform poorly (more buffering, could erroneously detect loss from delayed packets, etc.).

Option (B) is false because ECMP also requires finding all routes to a destination.

Option (C) is false because TCP handshakes have minimal overhead compared to the rest of the elephant flow. Also, we're splitting a single TCP flow across multiple paths, so you only need one handshake anyway.

Option (D) is incorrect because there is no huge overhead in forwarding packets along two different paths. In ECMP, routers have to be able to forward packets along multiple paths as well.

To implement the splitting up of an elephant TCP flow onto two paths, we identify the two paths with unique IDs (e.g. "Red" and "Blue").

Q6.4 (1 point) What packet header field could be used to tag packets within a single TCP flow with different path IDs?

For this subpart only, assume we are not adding any additional headers to the TCP-over-IP packets.

- ● Flow label (IPv6)
- ○ Next header (IPv6)
- ○ Protocol (IPv4)
- ○ Source port (TCP)

Solution: The flow label can be used to tag packets with a path ID. There's no requirement that packets in the same flow have to have the same flow label.

The protocol and next header fields won't work, since they need to specify that the next protocol in the stack is TCP.

If all the packets are in the same TCP flow, they should have the same source port, so changing the source port won't work.

In the rest of the question, instead of using any existing headers to tag packets with a path ID, we add a new step to the encapsulation process:

1. The virtual machine (VM) creates a packet with overlay IP addresses in the header.
2. **New Step:** The virtual switch adds another header with the ID of the path that the packet should take. Within a TCP flow, half the packets are tagged with the "Red" header, and the other half are tagged with the "Blue" header.
3. The virtual switch adds another header with the underlay IP addresses.

The ECMP hashing process is extended so that in addition to the usual 5-tuple, routers also use the path ID to decide where to forward the packet.

Q6.5 (2 points) What headers are present in the packet as it travels through intermediate routers in the underlay network? Select all that apply.

- ■ Headers with the overlay IP addresses.
- ■ Headers with the underlay IP addresses.
- ■ Headers with the path IDs.
- □ None of the above

Solution: The packet has all encapsulation headers wrapped around it as it travels through the underlay network.

Q6.6 (2 points) Consider modifying the ECMP hash so that it uses the 5-tuple and the path ID as input.

With this modified hash function, which sets of packets will always get forwarded to the same next-hop? Select all that apply. Each choice is independent.

- ☐ All packets in all TCP flows to the same destination IP.
- ☐ All packets in all TCP flows from the same source IP.
- ☐ All packets in all TCP flows between the same pair of hosts.
- ☐ All packets within a single TCP flow.
- ■ None of the above

Solution:

(A) is false. Two flows to the same destination IP might come from different sources, and since the source IP is included in the hash function, the flows may get forwarded to different next-hops.

(B) is false. Two flows with the same source IP may go to different destination IPs, and since the destination IP is included in the has function, the flows may get forwarded to different next-hops.

(C) is false. Two flows will have different source/destination ports, and since these are included in the hash function, the flows may get forwarded to different next-hops.

(D) is false. Because the path ID is included in the hash function, packets within a single flow may get forwarded to different next-hops.

Suppose we want to send a TCP flow across a single path, but we want the ability to switch to a different path if we detect congestion.

For example, we start by assigning Path ID "Red" to all packets, but when we detect congestion, we switch to assigning Path ID "Blue" to all subsequent packets.

Q6.7 (2 points) What are some ways that the host can detect congestion? Select all that apply.

Assume the network uses unmodified commodity routers, e.g. ECN (Explicit Congestion Notification) is not in use.

- ■ The host detects that the delay between sending a packet and receiving an ack is getting longer.
- ■ The host receives many duplicate acks.
- ☐ The host detects that the queues at the routers are getting longer.
- ☐ None of the above

Solution:

Option (A) is true. This is detecting congestion based on delay, as seen in lecture (e.g. Google's BBR protocol).

Option (B) is true. This is how TCP detects congestion, by detecting loss.

Option (C) is false. The host has no way to know what's happening at the routers, and since ECN is not in use, and we are not making another modifications to the router, the router is not giving any feedback to the host.

Q7 *Multicast* (5 points)

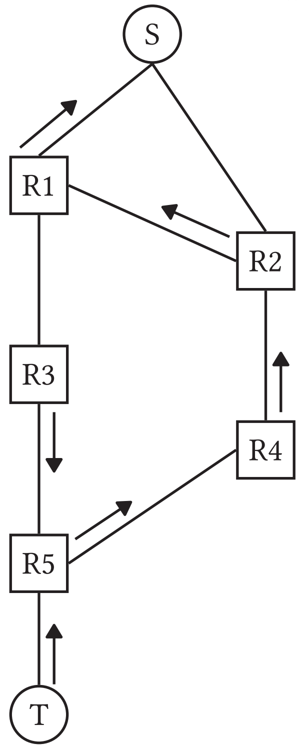
In this question, consider the network topology to the right.

The link costs are omitted (i.e. not all links necessarily cost 1, but their costs don't matter in this question).

The arrows show the directed delivery tree for destination S.

S and T are part of a multicast group.

In the next three subparts, S wants to send a multicast message to this group, using DVMRP. You can assume no pruning takes place.



Q7.1 (1 point) S forwards the packet to R2.

R2 takes this packet and $\underset{(1)}{\underline{\quad\quad}}$, because the incoming link $\underset{(2)}{\underline{\quad\quad}}$ the next-hop to S.

- ○ (1) Forwards it to R1 and R4, (2) is
- ○ (1) Drops it, (2) is
- ○ (1) Forwards it to R1 and R4, (2) is not
- [✓ (1) Drops it, (2) is not]

Solution: R2 drops the packet, because the incoming link is not the next-hop to S.

Q7.2 (1 point) S also forwards the packet to R1.

R1 takes this packet and $\underset{(1)}{\underline{\quad\quad}}$, because the incoming link $\underset{(2)}{\underline{\quad\quad}}$ the next-hop to S.

- [✓ (1) Forwards it to R2 and R3, (2) is]
- ○ (1) Drops it, (2) is
- ○ (1) Forwards it to R2 and R3, (2) is not
- ○ (1) Drops it, (2) is not

Solution: R1 forwards the packet to R2 and R3, because the incoming link is the next-hop to S.

Q7.3 (1 point) What path does the packet take from S to T?

[✓ S, R1, R2, R4, R5, T] S, R1, R3, R5, T

S, R2, R4, R5, T S, R2, R1, R3, R5, T

Solution: Continuing the protocol from the previous subparts:

R1 forwards the packet to R2.

R2 notices that the packet came from the next-hop to S, so R2 forwards it out of all its links, including to R4.

R4 receives the packet and notices that the packet came from R2 (the next-hop to S), so R4 forwards it out of all its links, including to R5.

R5 receives the packet and notices that R5 is directly connected to someone in the group (T), so R5 sends the packet to T.

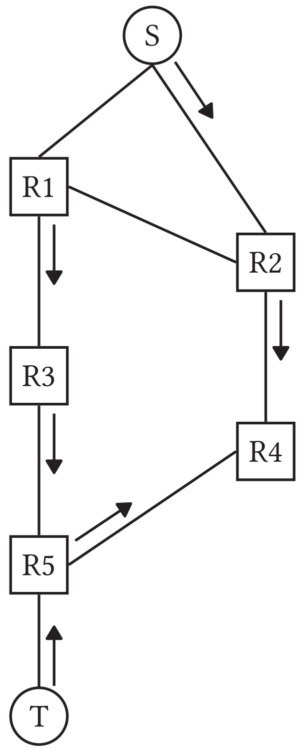
Note: R1 also forwards the packet to R3, but R3 will notice that the packet came from R1 (not the next-hop to S), so R3 will just drop the packet.

Now, consider the same topology from before.

The arrows show the directed delivery tree for destination R4.

As before, S and T are part of a multicast group.

In the next two subparts, S wants to send a multicast message to this group, using CBT, with a core router of R4.



Q7.4 (1 point) First, S and T join the group by sending join messages to R4.

At this point, which routers are on the tree? Select all that apply.

☐ R1 [✓ R2] ☐ R3 [✓ R4] [✓ R5]

Solution:

S sends a join message, which gets forwarded from S to R2 to R4. This causes R2 to join the tree.

T sends a join message, which gets forwaded from T to R5 to R4. This causes R5 to join the tree.

R4 is on the tree, since it's the core.

Q7.5 (1 point) Once the tree is built, S can now send a multicast message to the group.

What path does the packet take from S to T?

○ S, R1, R2, R4, R5, T ○ S, R1, R3, R5, T

[✓ S, R2, R4, R5, T] ○ S, R2, R1, R3, R5, T

Solution: Since S is already on the tree, S just needs to broadcast the packet along the tree.

S sends the packet to all its neighbors on the tree, namely just R2.

Then, R2 sends the packet to all its neighbors on the tree, namely just R4.

Then, R4 sends the packet to all its neighbors on the tree, namely just R5.

Finally, R5 sends the packet to any directly-connected group members, namely just T.

Q8 *Wireless* **(7 points)**

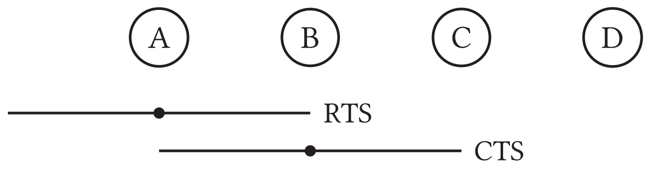
Consider four hosts communicating over a single frequency channel, using the MACA protocol from class (not MACAW, i.e. no ack, no DS, no RRTS). Each subpart is independent.

In the diagrams below, when a message is transmitted, the line segments show who receives the transmission.

Q8.1 (1 point) A transmits a Request to Send (RTS) packet, and the RTS is received by B.

Then, B transmits a Clear to Send (CTS) packet, and the CTS is received by A and C.

No other packets are transmitted.



At this point, who is allowed to transmit data packets?

[✓ A] B C D

Solution: A sent an RTS and received a CTS, so A is clear to send data.

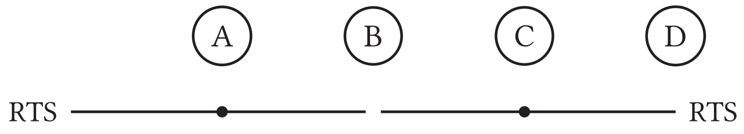
Note that D is not allowed to transmit data at this point, unless it initiates an RTS/CTS exchange of its own.

Q8.2 (2 points) Desired communications:

- A wants to talk to B.
- C wants to talk to B.

Suppose A and C each simultaneously transmit an RTS packet with equal signal strength, and these transmissions arrive at B at the same time.

No other packets are transmitted.



At this point, what will happen next at B?

○ B picks the RTS from the host who wants to transmit the least amount of data, and transmits a CTS packet for that host's RTS.

[✓ B receives both packets at the same time, causing a collision.]

○ B randomly picks one of the two hosts, and transmits a CTS packet for that host's RTS.

○ B sends a CTS packet for both hosts' RTS packets, allowing both A and C to transmit to B.

Solution: Since B received both RTS packets at the same time, this is a collision, and MACA isn't able to differentiate between these two signals, so both packets are dropped at B.

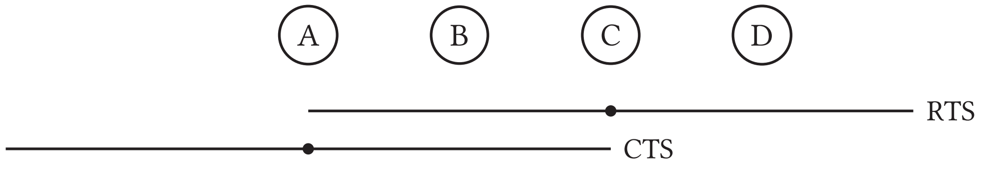
The nuance here is that the hidden terminal problem is not when both A and C transmit at the same time. Rather, it is when A transmits first, and after some time C thinks A hasn't transmit anything, leading to C transmitting its data, and therefore a collision at B.

For the next three subparts, consider this scenario:

Desired communications:

- C wants to talk to A.
- B wants to talk to D.

C transmits an RTS packet, and A, B, and D hear it. Then, A transmits a CTS packet, and B and C hear it. No other packets are transmitted.



Q8.3 (1 point) At this point, what will happen next at B?

- ○ B immediately transmits its data packets, without sending an RTS packet first.
- ○ B immediately transmits an RTS packet.
- [✓ B waits for A and C to finish, and then attempts to transmit an RTS packet.]
- ○ B never sends an RTS packet, and gives up on transmitting any data.

Solution: Since B is in the range of A, B hears A's CTS packet, and therefore waits for A and C to exchange their data in order to avoid any collisions. Only after that will B start transmitting an RTS packet to D.

Q8.4 (2 points) At this point, can A and C both transmit data packets to each other simultaneously (with no further RTS/CTS exchanges)?

- ○ Yes, because A transmits a CTS, so everyone in A's range must be quiet. This guarantees that A's data transmission will be received by C.
- ○ Yes, because in MACA, after a single RTS/CTS exchange, the two hosts can take turns transmitting data.
- ○ No, but if C also transmitted a CTS at this point, then A and C can both transmit data simultaneously.
- [✓ No, because if A and C both transmit data packets to each other at this point, this could cause collisions.]

Solution: Intuitively, the answer is no, because if both hosts try to transmit to each other simultaneously after an RTS/CTS exchange, their signals would just clobber each other out.

Option (A) is false, because we actually need everyone in the receiver's range (C's range) to be quiet, not just everyone in the sender's range (A's range). For example, if D is also transmitting to C, then C will not receive A's transmission. A transmitting a CTS will not help because A's CTS may not reach D.

Option (B) is false, because in MACA, there is no notion of taking turns after an RTS/CTS exchange. Each "turn" in the protocol must be preceded by its own RTS/CTS exchange.

Option (C) is false, because even if C transmitted a CTS at this point, A and C's signals would still clobber each other out.

Option (D) is true, since A and C both transmitting data at the same time will cause collisions.

Q8.5 (1 point) For this subpart only, suppose the CTS transmitted by A is dropped, and C does not receive the CTS.

At this point, what will happen next, and what happens to C's contention window (CW)?

- [✓ C increases its CW, and tries again by transmitting another RTS later.]
- ○ C decreases its CW, and tries again by transmitting another RTS later.
- ○ C increases its CW, and starts transmitting its data packets.
- ○ C decreases its CW, and starts transmitting its data packets.

Solution: CW tells you how long to wait before trying again. Since the RTS/CTS exchange failed, C will think that the channel is congested, and increase its CW to wait longer before transmitting the RTS again.

Because the RTS/CTS exchange failed, C cannot send data. It must try another RTS/CTS exchange later, and can only send data after a successful RTS/CTS exchange.

Q9 *Cellular* (5 points)

Recall the steps of using a cellular network, as seen in lecture. For each subpart, select the step in which the event occurs. Not all choices may be used, and some choices may be used more than once. Each subpart is independent.

If the event never occurs in the process of using a cellular network, select None of the above.

Q9.1 (1 point) The user receives an IMSI (subscriber ID).

- [✓ Step 0: Registration]
- Step 1: Discovery
- Step 2: Attachment
- Step 3: Data Exchange
- Step 4: Handover
- None of the above

Solution: When the user first registers for a new mobile data plan, they receive a subscriber ID.

Q9.2 (1 point) The routers in the cellular core use a routing algorithm (e.g. distance-vector) to find the shortest path to the user.

- Step 0: Registration
- Step 1: Discovery
- Step 2: Attachment
- Step 3: Data Exchange
- Step 4: Handover
- [✓ None of the above]

Solution: The cellular core routers never perform distance-vector routing. Instead, the mobility manager configures tunnels to tell the cellular core how to forward packets to/from the user.

Q9.3 (1 point) Suppose an attacker steals a victim user's IMSI. The attacker does not know any other information about that victim user.

The attacker tries to connect to the cellular network by presenting the victim's IMSI to the mobility manager. The mobility manager detects that the connection is invalid at this step.

- ○ Step 0: Registration
- ○ Step 1: Discovery
- [✓ Step 2: Attachment]
- ○ Step 3: Data Exchange
- ○ Step 4: Handover
- ○ None of the above

Solution:

During the Attachment step, the attacker must present additional cryptographic information to prove that they own the IMSI, but will be unable to do so.

Note that the cellular network will not detect the invalid connection during the Discovery step, because the attacker is just listening to beacons during this step, and does not actually attempt to initiate a connection.

Q9.4 (1 point) Consider a user who is already connected to the cellular network. The user is moving in and out of the range of various towers.

The user's device listens for nearby towers. The user's device (not the cellular network) picks the next tower to use during this step.

- ○ Step 0: Registration
- ○ Step 1: Discovery
- ○ Step 2: Attachment
- ○ Step 3: Data Exchange
- ○ Step 4: Handover
- [✓ None of the above]

Solution: During handover, the next tower to use is decided by the cellular network, not by the user.

Q9.5 (1 point) The mobility manager tells the cellular core routers to delete an existing tunneled path, and install a new tunneled path.

- ○ Step 0: Registration
- ○ Step 1: Discovery
- ○ Step 2: Attachment
- ○ Step 3: Data Exchange
- [✓ Step 4: Handover]
- ○ None of the above

Solution: During handover, the tunneled path between the user and the network changes.

Note that Attachment is incorrect here, because the routers are deleting an existing path, which implies that the user has already been connected to the network before this step.

This page intentionally left (mostly) blank.

The exam continues on the next page.

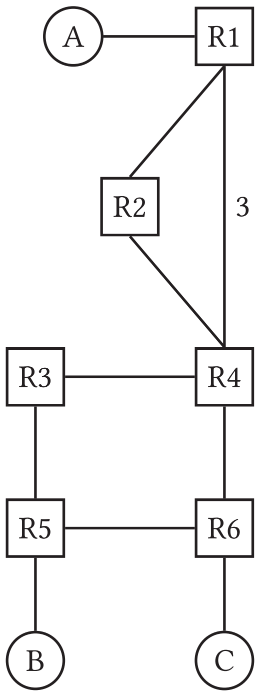
Q10 *Midterm Clobber: STP* **(12 points)**

This question will only be used to potentially replace your score on the midterm STP question. It will not affect your final exam score.

Consider running the Spanning Tree Protocol (STP) for the network topology to the right.

Assume the IDs are ordered according to the router labels. For example, R4 has a lower ID than R5.

Assume the links with no label have a cost of 1.



Q10.1 (2 points) After running STP, how many links get disabled?

○ 1 [✓ 2] ○ 3 ○ 4 ○ 5 ○ 6

Solution:

R1 is the root, because it has the lowest-numbered ID.

STP will produce the shortest paths tree rooted at R1. This tree does not include R1-to-R4 and R5-to-R6, so those two links get disabled.

Q10.2 (2 points) Select all routers that disable at least one of their links.

☐ R1 ☐ R2 ☐ R3 [✓ R4] ☐ R5 [✓ R6]

Solution: R6 disables its link to R5.

R4 disables its link to R1.

Q10.3 (1 point) During STP, the **first** message that R4 announces to its neighbors is:

"The root is _____, and I can reach the root with cost _____."

○ R1; 0	○ R1; 3	[✓ R4; 0]
○ R1; 2	○ R3; 1	○ A; 3

Solution: All routers initially think that they are the root, so the first message R4 announces is:

"The root is R4, and I can reach the root with cost 0."

Q10.4 (1 point) After STP converges, R4 thinks that the root is _____, and that R4 can reach the root with cost _____.

○ R1; 0	○ R1; 3	○ R4; 0
[✓ R1; 2]	○ R3; 1	○ A; 3

Solution:

After STP converges, all routers agree on the root, which is R1 (lowest ID).

Also, all routers have a shortest path to the root, which means that R4 thinks that it can reach the root with cost 2.

More detailed solution, showing how STP operates:

Initially, all routers believe they are the root.

R1 (eventually the true root) announces to R2 and R4: "The root is R1, and I can reach the root with cost 0."

R2 initially thinks that itself is the root, but it hears a message from R1 and concludes that the root must be R1, not R2 (because R1 has a lower ID). Also, R2 realizes that it can reach the root with cost 1 (0 from announcement, plus 1 from link cost).

Then, R2 can announce its new findings to its neighbors, including R4: "The root is R1, and I can reach the root with cost 1."

R4 initially thinks that itself is the root, but it hears the same message from R1 and concludes that the root must be R1, not R4 (because R1 has a lower ID). Also, R4 realizes that it can reach the root with cost 3 (0 from announcement, plus 3 from link cost).

Then, R4 can announce its new findings to its neighbors: "The root is R1, and I can reach the root with cost 3."

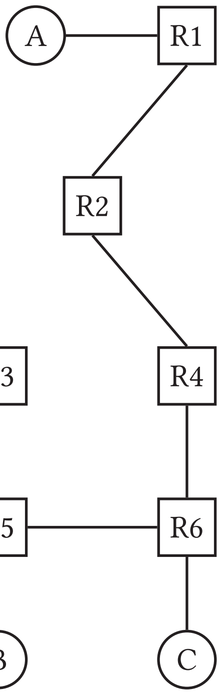
Next, R4 hears the message that R2 sent earlier. It realizes that the cost to the root via R2 is 2 (1 from announcement, plus 1 from link cost). This cost (2) is better than R4's best-known cost to root (3), so R4 now knows that its new best cost to the root is 2.

Then, R4 can announce its new findings to its neighbors: "The root is R1, and I can reach the root with cost 2."

This is the last message that R4 sends, because its cost to R1 will not update any further.

Note that even if the messages came in a different order, e.g. R4 hears from R1 before it hears from R2, the answer would be unchanged. R4 would first announce that it can reach the root with cost 2, then it would not announce that it can reach the root with cost 3 (because that cost is worse).

Suppose STP has converged. Regardless of your answers to the previous subparts, assume that the R3-to-R4 and R1-to-R4 links are disabled:



Switches R1 to R6 are all learning switches.

The forwarding tables start out empty, except for R5's forwarding table, which starts with one hard-coded entry:

R5's Forwarding Table	
Destination	Next Hop
A	R6

In each of the next 3 subparts, select all switches that will receive the given packet.

The packets are sent one after the other. In other words, forwarding table entries created in one subpart carry over to later subparts.

Q10.5 (2 points) C sends a packet to A.

■ R1 ■ R2 □ R3 ■ R4 ■ R5 ■ R6

Solution: C forwards the packet to R6.

R6's forwarding table is empty, so R6 floods the packet to R4 and R5.

R5 receives the packet, and because its table has a hard-coded entry for destination A, it forwards the packet back to R6 only. This means that R3 does not receive the packet.

(Note: R6 will not flood the packet a second time, because R6 has already seen and flooded the packet. Even if you assumed that R6 flooded the packet a second time, the answer would still be the same.)

R4 receives the packet, and its forwarding table is empty, so it floods the packet to R6 and R2.

R2 receives the packet, and its forwarding table is empty, so it floods the packet to R1 and R4.

R1 receives the packet, and forwards it to destination A.

Q10.6 (2 points) A sends a packet to C.

■ R1 ■ R2 □ R3 ■ R4 □ R5 ■ R6

Solution:

From the previous subpart, routers have learned next-hops to C.

From the C-to-R6 message, R6 learned a next-hop to C.

From the R6-to-R4 message, R4 learned that its next-hop to C is R6.

From the R4-to-R2 message, R2 learned that its next-hop to C is R4.

From the R2-to-R1 message, R1 learned that its next-hop to C is R2.

Using these forwarding table entries, the A-to-C packet gets forwarded from A to R1, then to R2, then to R4, then to R6, then to C.

Note that R5 does not receive this packet, because R6 knows the next-hop to C (and does not flood).

Q10.7 (2 points) B sends a packet to A.

■ R1 ■ R2 □ R3 ■ R4 ■ R5 ■ R6

Solution:

From the previous subpart, routers have learned next-hops to A.

From the A-to-R1 message, R1 learned a next-hop to A.

From the R1-to-R2 message, R2 learned that its next-hop to A is R1.

From the R2-to-R4 message, R4 learned that its next-hop to A is R2.

From the R4-to-R6 message, R6 learned that its next-hop to A is R4.

Now we can think about forwarding the B-to-A packet.

B forwards the packet to R5.

R5 has a hard-coded entry for destination A, so R5 forwards the packet to R6. (Notably, R3 never receives the packet.)

R6 receives the packet, and directly forwards the packet to R4.

R4 receives the packet, and directly forwards the packet to R2.

R2 receives the packet, and directly forwards it to R1.

R1 forwards the packet to A.

Nothing on this page will affect your grade.

Comment Box

Congrats for making it to the end of the exam! Leave any thoughts, comments, feedback, or doodles here.

CS 168's new course mascot is an octopus! Suggest any designs, names, backstories, etc. below:

Ambiguities

If you feel like there was an ambiguity on the exam, you can put it in the box below.

For ambiguities, you must qualify your answer and provide an answer for both interpretations. For example, "if the question is asking about A, then my answer is X, but if the question is asking about B, then my answer is Y." You will only receive credit if it is a genuine ambiguity and both of your answers are correct. We will only look at this box if you request a regrade.