CS 168 Spring 2026 — Introduction to the Internet Final Exam

Solutions last updated: May 22, 2026

Print Your Name: _____

Print Your Student ID: _____

You have 170 minutes. There are 8 questions of varying credit. (100 points total)

Question:	1	2	3	4	5	6	7	8	Total
Points:	25	15	10	13	11	7	11	8	100

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

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

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

- ■ You can select
- ■ multiple squares
- ☑ 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: _____

Clarifications

- Q1.13: Packet loss occurs immediately after when CWND first reaches 31 packets.
- Q7.4: As a reminder, the k = ~~4~~ 2 topology from lecture is shown below
- Q2.1 and Q2.2 are independent of each other.
- Q5.5: “This” topology refers to the diagram at the beginning of Q5.

Q1 *TCP Congestion Control* **(25 points)**

For the entire question, all TCP values are measured in packets, not bytes, unless otherwise specified.

In Q1.1 to Q1.7, consider two modifications to the TCP implementation from lecture:
- In slow start, CWND is multiplied by 3 on each RTT (instead of 2).
- If loss is detected from duplicate ACKs, CWND is divided by 3 (instead of 2).

The rest of TCP is unchanged, except where these two modifications require corresponding updates.

Q1.1 (2 points) To implement the modified slow start with event-driven updates, CWND should increase by ___ on each new ACK received.

Ⓐ 0.5 Ⓑ 1 Ⓒ 1.5 Ⓓ 2 Ⓔ 2.5 Ⓕ 3

Solution: CWND starts at 1.

In the first RTT, we send out 1 packet and receive 1 ACK. We want CWND to increase to 3, so CWND should increase by 2 for the ACK we received.

In the next RTT, we send out 3 packets and receive 3 ACKs. We want CWND to increase from 3 to 9, for a total increase of 6. Therefore, CWND should increase by $\frac{6}{3} = 2$ for each ACK we received.

In the next RTT, we send out 9 packets and receive 9 ACKs. We want CWND to increase from 9 to 27, for a total increase of 18. Therefore, CWND should increase by $\frac{18}{9} = 2$ for each ACK we received.

This pattern continues throughout slow start.

Q1.2 (1 point) Consider sending an endless stream of data on a link with some constant, unknown bandwidth. No other users are using the link.

We run two separate, independent experiments on the same link:
- Run unmodified slow start (doubling CWND), and record the first non-infinite SSTHRESH value.
- Run modified slow start (tripling CWND), and record the first non-infinite SSTHRESH value.

Which algorithm has the higher initial finite value of SSTHRESH?

Ⓐ Unmodified (doubling) Ⓑ Modified (tripling) Ⓒ Not enough information

Solution: In both algorithms, the first finite value of SSTHRESH is the last safe value reached in slow start before we exceed the bandwidth and start encountering loss.

In unmodified slow start, SSTHRESH is the highest power of 2 that is less than the bandwidth.

In modified slow start, SSTHRESH is the highest power of 3 that is less than the bandwidth.

However, since we do not know the bandwidth, we do not know which of these values is greater.

For example, if the bandwidth is 100, then unmodified slow start sets SSTHRESH to 64, and modified slow start sets SSTHRESH to 81.

However, if the bandwidth is 70, then unmodified slow start sets SSTHRESH to 64, and modified slow start sets SSTHRESH to 27.

Q1.3 (1 point) When we enter fast recovery (after receiving 3 duplicate ACKs), what should CWND be?

Ⓐ $\frac{\text{CWND}}{2} + 2$ Ⓑ $\frac{\text{CWND}}{2} + 3$ Ⓒ $\frac{\text{CWND}}{3} + 2$ Ⓓ $\frac{\text{CWND}}{3} + 3$

Solution: In the unmodified algorithm, when we enter fast recovery, CWND is set to $\frac{\text{CWND}}{2} + 3$. The division by 2 comes from the multiplicative decrease factor, and the $+3$ comes from artificially inflating the window for the 3 duplicate ACKs received.

In the modified algorithm, the multiplicative decrease factor changes from division by 2 to division by 3. The duplicate-ACK threshold is still 3, so the $+3$ term stays the same.

Therefore, CWND should be set to $\frac{\text{CWND}}{3} + 3$.

Q1.4 (1 point) During fast recovery, CWND should increase by ___ on each duplicate ACK received.

Ⓐ 0.5 Ⓑ 1 Ⓒ 1.5 Ⓓ 2 Ⓔ 2.5 Ⓕ 3

Solution: No change is necessary here. On each duplicate ACK received, TCP should artificially extend the window by 1 packet to allow the next packet to be sent. The multiplicative constant (2 or 3) is not relevant here.

The two modifications, reprinted for your convenience:
- In slow start, CWND is multiplied by 3 on each RTT (instead of 2).
- If loss is detected from duplicate ACKs, CWND is divided by 3 (instead of 2).

(Question 1 continued...)

Next, we will derive a throughput equation for the modified TCP, using the same steady-state assumptions as the original equation:

- A single connection is sending an endless stream of data on a link.
- The link has fixed maximum bandwidth $W_{\max}$, and packet loss occurs when CWND reaches this value.
- CWND is adjusted only by AIMD updates (i.e. ignore slow start and fast recovery).
- RTT is fixed.

Q1.5 (1 point) Over one steady-state AIMD cycle, what is the average value of CWND?

Ⓐ $\frac{1}{6}W_{\max}$ Ⓑ $\frac{1}{3}W_{\max}$ Ⓒ $\frac{1}{2}W_{\max}$ **Ⓓ** $\frac{2}{3}W_{\max}$ Ⓔ $\frac{5}{6}W_{\max}$ Ⓕ $W_{\max}$

Solution: CWND climbs linearly from $\frac{1}{3}W_{\max}$ to $W_{\max}$, then drops back to $\frac{1}{3}W_{\max}$, and repeats.

The average value of CWND is the midpoint between its minimum and maximum values:

$$\frac{1}{2}\left(\frac{1}{3}W_{\max} + W_{\max}\right) = \frac{2}{3}W_{\max}$$

Q1.6 (2 points) In steady state, how many packets are sent between consecutive loss events?

Ⓐ $\frac{2}{9}W_{\max}$ Ⓒ $\frac{1}{3}W_{\max}$ Ⓔ $\frac{4}{9}W_{\max}$ Ⓖ $\frac{5}{9}W_{\max}$ Ⓘ $\frac{2}{3}W_{\max}$ Ⓚ $\frac{7}{9}W_{\max}$

Ⓑ $\frac{2}{9}W_{\max}^2$ Ⓓ $\frac{1}{3}W_{\max}^2$ **Ⓕ** $\frac{4}{9}W_{\max}^2$ Ⓗ $\frac{5}{9}W_{\max}^2$ Ⓙ $\frac{2}{3}W_{\max}^2$ Ⓛ $\frac{7}{9}W_{\max}^2$

Solution: The AIMD graph increases linearly from $\frac{1}{3}W_{\max}$ to $W_{\max}$, increasing by 1 per RTT.

Therefore, there are $W_{\max} - \frac{1}{3}W_{\max} = \frac{2}{3}W_{\max}$ RTTs between loss events.

The average value of CWND is $\frac{2}{3}W_{\max}$, so this is the average number of packets sent in each RTT.

Therefore, the total number of packets sent between two consecutive losses is:

$$\underbrace{\frac{2}{3}W_{\max}}_{\text{packets per RTT}} \cdot \underbrace{\frac{2}{3}W_{\max}}_{\text{RTTs between losses}} = \frac{4}{9}W_{\max}^2$$

Q1.7 (2 points) In this subpart only, ignore your answers to the previous two subparts and instead assume:

- The average value of CWND is $\frac{1}{3}W_{\max}$.
- The number of packets sent between two losses is $\frac{3}{4}W_{\max}^2$.

Based on these values, what is the value of k in the throughput equation?

$$\text{throughput} = k \cdot \frac{\text{MSS}}{\text{RTT}\ \sqrt{p}}$$

Ⓐ $\frac{\sqrt{3}}{9}$ Ⓑ $\frac{\sqrt{6}}{9}$ Ⓒ $\frac{2\sqrt{3}}{9}$ Ⓓ $\frac{2\sqrt{6}}{9}$ Ⓔ $\frac{2\sqrt{3}}{3}$ Ⓕ $\frac{\sqrt{3}}{3}$

Solution: First, we use the first value, the average value of CWND:

$$\begin{aligned}\text{throughput} &= \frac{\text{packets sent per RTT}}{\text{RTT}} \\ &= \frac{\text{average CWND value}}{\text{RTT}} \\ &= \frac{\frac{1}{3}W_{\max} \cdot \text{MSS}}{\text{RTT}} \\ &= \frac{1}{3}W_{\max} \cdot \frac{\text{MSS}}{\text{RTT}}\end{aligned}$$

Next, we use the second value to compute the loss rate, and solve for $W_{\max}$ in terms of p:

$$\begin{aligned}p &= \frac{1}{\frac{3}{4}W_{\max}^2} \\ p &= \frac{4}{3W_{\max}^2} \\ W_{\max}^2 &= \frac{4}{3p} \\ W_{\max} &= \frac{2}{\sqrt{3p}} \\ W_{\max} &= \frac{2\sqrt{3}}{3\sqrt{p}}\end{aligned}$$

Finally, we substitute this expression for $W_{\max}$ back into the throughput equation:

$$\begin{aligned}\text{throughput} &= \underbrace{\frac{1}{3}\left(\frac{2\sqrt{3}}{3\sqrt{p}}\right)}_{W_{\max}} \cdot \frac{\text{MSS}}{\text{RTT}} \\ &= \frac{2\sqrt{3}}{9} \cdot \frac{\text{MSS}}{\text{RTT}\ \sqrt{p}}\end{aligned}$$

In Q1.8 to Q1.9, consider the two modifications from earlier, with one additional modification: We enter fast recovery when we receive 2 duplicate ACKs (instead of 3 duplicate ACKs).

Q1.8 (1 point) When we enter fast recovery, what should CWND be set to?

Ⓐ $\frac{\text{CWND}}{2} + 2$ Ⓑ $\frac{\text{CWND}}{3} + 2$ Ⓒ $\frac{\text{CWND}}{2} + 3$ Ⓓ $\frac{\text{CWND}}{3} + 3$

Solution: In this new variant, the multiplicative decrease factor is still division by 3, so the first part of the expression is $\frac{\text{CWND}}{3}$.

The duplicate-ACK threshold is now 2, so we artificially inflate the window by 2 rather than 3.

Therefore, CWND should be set to $\frac{\text{CWND}}{3} + 2$.

Q1.9 (2 points) How does this new modification affect TCP (compared to the version with only two modifications)? Select all that apply.

[A] It has no effect on sending rate in networks where packet reordering occurs.

[B] It can reduce the average sending rate, because TCP may cut its window more often.

[C] During fast recovery, CWND should now increase by 2 on each duplicate ACK received.

[D] It affects the throughput equation (which models TCP using only AIMD).

[E] It makes TCP more likely to reduce CWND, even when no packet was dropped.

Ⓕ None of the above

Solution: Lowering the duplicate-ACK threshold from 3 to 2 makes TCP more sensitive to packet reordering. As a result, unnecessary fast retransmits become more likely.

Because TCP may enter fast recovery and reduce CWND more often, this change can reduce the average sending rate, especially on paths where reordering occurs.

This also makes TCP more likely to reduce CWND even when no packet was actually dropped.

However, during fast recovery, CWND should still increase by 1 on each duplicate ACK received.

Also, this change does not affect the steady-state AIMD throughput equation, because that model explicitly assumes CWND is adjusted only by additive increase and multiplicative decrease, and ignores fast recovery.

Q1.10 to Q1.11 are independent of earlier subparts (i.e. ignore the modifications above). For these subparts, assume:

- TCP transfers occur on a path with fixed RTT and fixed available bandwidth.
- Packet loss only occurs when CWND exceeds the available bandwidth.

Q1.10 (2 points) In this subpart only, consider unmodified TCP from lecture.

Two long, ongoing TCP connections, X and Y, are identical, except for one difference:
- Connection X currently has a **lower** SSTHRESH value.
- Connection Y currently has a **higher** SSTHRESH value.

How does Connection Y compare to Connection X? Select all that apply.

[A] Y may stay in slow start for more RTTs before switching to congestion avoidance.

[B] Y may reach a high sending rate sooner.

[C] Y may overshoot the available bandwidth by more before switching to congestion avoidance.

Ⓓ None of the above

Solution: A larger initial SSTHRESH means TCP may remain in slow start for more RTTs before switching to congestion avoidance.

Because slow start grows CWND exponentially, this can let TCP reach a high sending rate sooner at the beginning of the connection.

However, it also means TCP may overshoot the available bandwidth by more before leaving slow start.

This change can affect application throughput, especially at the beginning of the connection, even though additive increase is unchanged.

It also does not affect the steady-state AIMD throughput equation, because that model explicitly ignores slow start.

Q1.11 (2 points) In this subpart only, consider unmodified TCP from lecture.

A TCP connection is sending data on a path with a much larger RTT than a typical network (e.g. an Earth-to-Moon link).

How does this connection compare to a connection with a smaller RTT? Select all that apply.

[A] It takes longer in real time for slow start to grow CWND to a given value.

[B] At a given CWND value, the sending rate in packets per second is lower.

[C] If a timeout occurs, recovering to the previous sending rate may take longer in real time.

Ⓓ None of the above

Solution: A larger RTT means each round trip takes more real time, so slow start takes longer in real time to reach a given CWND.

At a given CWND value, sending rate is roughly CWND divided by RTT, so a larger RTT means a lower sending rate in packets per second.

If a timeout occurs, TCP may need to grow its sending rate again over multiple RTTs. With a larger RTT, this recovery takes longer in real time.

However, a larger RTT does not change the congestion avoidance rule itself: TCP still increases CWND by 1 packet per RTT.

Q1.12 (2 points) In this subpart only, we modify TCP by making the initial CWND larger (with no other modifications).

For a **short** transfer, how does this TCP version compare to unmodified TCP? Select all that apply.

[A] The sender can transmit more data before the first ACK returns.

[B] The transfer may finish in fewer RTTs.

[C] The sender may trigger packet loss sooner if the path cannot support the larger initial CWND.

[D] Application completion time is unchanged, because additive increase is unchanged.

[E] The change affects the throughput equation (which models TCP using only AIMD).

(F) None of the above

Solution: A larger initial CWND allows the sender to transmit more data before the first ACK returns.

For a short transfer, this can reduce the number of RTTs needed to finish sending the data.

However, if the path cannot support the larger startup window, this more aggressive startup behavior can trigger packet loss sooner.

This change can affect application completion time, especially for short transfers, even though additive increase is unchanged.

It also does not affect the steady-state AIMD throughput equation, because that model explicitly ignores slow start.

In Q1.13 to Q1.15 (which are independent of all earlier subparts), consider this modified version of TCP:

- In slow start, CWND is multiplied by 3 each RTT.
- In congestion avoidance, CWND increases by 1 packet per RTT.
- When loss is detected from duplicate ACKs, CWND is divided by 3.
- TCP enters fast recovery after 2 duplicate ACKs.

An ongoing TCP connection currently has these parameters:

- Current mode = slow start.
- Current CWND = 3 packets.
- Current SSTHRESH = 27 packets.
- RTT = 0.5 seconds.
- Packet loss occurs after CWND first reaches 31 packets.
- After TCP enters fast recovery, the retransmitted packet is acknowledged exactly 1 RTT later.

Assume packet transmission time and processing time are negligible.

Q1.13 (2 points) Starting from the given parameters, how many seconds pass before TCP first enters congestion avoidance?

1 seconds

Solution: TCP starts with CWND = 3.

In slow start:
- after 1 RTT, CWND = 9
- after 2 RTTs, CWND = 27

At this point, TCP first enters congestion avoidance. Since RTT = 0.5 seconds, the elapsed time is:

$$2 \cdot 0.5 = 1.0 \text{ second}$$

Note that 1.5 seconds = 3 RTTs is incorrect. Remember that the sliding window allows multiple packets to be in flight at any given time. Also, remember that event-driven slow start increases CWND on every ack received. Therefore, event-driven slow start would cause CWND to increase and the mode to switch on the very next ack received after $t = 1$, so the entire extra RTT is not needed.

Q1.14 (2 points) Starting from when TCP first enters congestion avoidance, how many additional seconds pass before TCP first enters fast recovery?

2.5 seconds

Solution: In congestion avoidance, CWND increases by 1 packet per RTT.

Starting from CWND = 27:
- after 1 RTT, CWND = 28
- after 2 RTTs, CWND = 29
- after 3 RTTs, CWND = 30
- after 4 RTTs, CWND = 31

Packet loss occurs when CWND first reaches 31. However, at the instant the packet is lost, the sender is still unaware of the loss.

At this point, we need one additional RTT, so that the receiver sees the missing packet (i.e. a gap in consecutive sequence numbers) and starts sending back duplicate acks.

Therefore, TCP first enters fast recovery after 5 additional RTTs:

$$5 \cdot 0.5 = 2.5 \text{ seconds}$$

Partial credit was awarded for answering 2 seconds, i.e. missing the extra RTT needed for duplicate acks to arrive.

Q1.15 (2 points) Starting from when TCP first enters fast recovery, how many additional seconds pass before TCP exits fast recovery?

0.5 seconds

Solution: Once TCP enters fast recovery, we are told that the retransmitted packet is acknowledged exactly 1 RTT later. TCP exits fast recovery when that ACK arrives.

Therefore, the additional time spent in fast recovery is:

$$1 \cdot 0.5 = 0.5 \text{ seconds}$$

Q2 *DNS: Aliasing* **(15 points)**

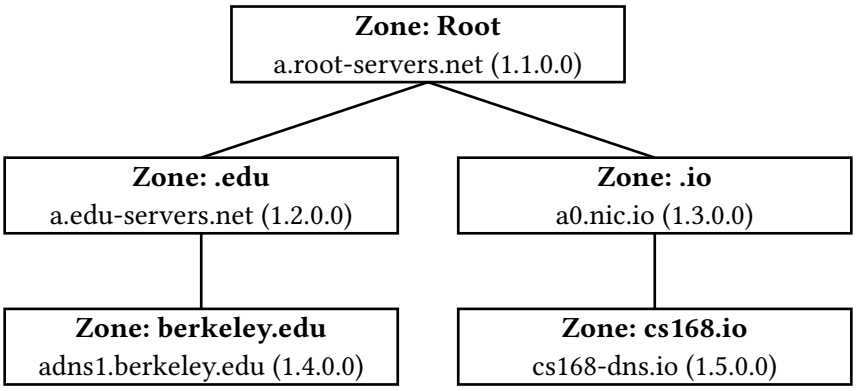
In DNS, the `CNAME` record type holds a name-value pair, where the name is a domain, and the value is another domain. A `CNAME` record indicates the name domain is an *alias* of the value domain.

"Alias" means that the name domain is another way to refer to the value domain. When a user queries for the name domain, they will find the `CNAME` record, which tells them to query for the value domain instead.

For example, `cs168.berkeley.edu CNAME sp26.cs168.io` is a record that indicates that `cs168.berkeley.edu` is another way to refer to `sp26.cs168.io`.

If a user queries for the IP address of `cs168.berkeley.edu`, they will find this `CNAME` record, which tells the user to look up the IP address of `sp26.cs168.io` instead.

Consider the DNS hierarchy below. Each box contains a zone, and the domain and IP of the authoritative name server for that zone. There are no other zones and name servers besides the ones shown.



In the berkeley.edu zone, we add the record: `cs168.berkeley.edu CNAME sp26.cs168.io`

In Q2.1 to Q2.2, a recursive resolver starts with an empty cache, and first makes a query to fill up the cache. Assume no DNS or network errors occur.

Then, some time later, without any cached records expiring, the resolver wants to know the IP address of `cs168.berkeley.edu`. How many DNS queries does the resolver send in order to resolve this query?

Q2.1 (1 point) The resolver starts with an empty cache.

How many queries are needed to learn the IP address of `cs168.berkeley.edu`?

Ⓐ 0 Ⓑ 1 Ⓒ 2 Ⓓ 3 Ⓔ 4 Ⓕ 5 Ⓖ 6 Ⓗ 7

Solution:

1. Ask root about `cs168.berkeley.edu`.
2. Ask the .edu server about `cs168.berkeley.edu`.
3. Ask the berkeley.edu server about `cs168.berkeley.edu`.

At this point, the berkeley.edu server gives us the `CNAME` record.

The cache started empty, and none of the records returned so far are relevant to `sp26.cs168.io`, so we need to start over and perform another query for the IP address of `sp26.cs168.io`.

4. Ask root about `sp26.cs168.io`.
5. Ask the .io server about `sp26.cs168.io`.
6. Ask the cs168.io server about `sp26.cs168.io`.

Q2.2 (1 point) The resolver first queries for the IP address of `sp26.cs168.io`, and caches all records from this query.

Now, how many queries are needed to learn the IP address of `cs168.berkeley.edu`?

Ⓐ 0 Ⓑ 1 Ⓒ 2 Ⓓ 3 Ⓔ 4 Ⓕ 5 Ⓖ 6 Ⓗ 7

Solution:

Note that none of the records from the `sp26.cs168.io` query are useful for learning the IP address of `cs168.berkeley.edu`, so we have to start this query from scratch (i.e. asking the root first):

1. Ask root about `cs168.berkeley.edu`.
2. Ask the .edu server about `cs168.berkeley.edu`.
3. Ask the berkeley.edu server about `cs168.berkeley.edu`.

At this point, the berkeley.edu server gives us the `CNAME` record. This tells us to look up the IP address of `sp26.cs168.io` instead.

From the first query, we already have the `A` record with the IP address of `sp26.cs168.io`, so we can use this cached record, and no further queries are needed.

EvanBot suggests storing the record `sp26.cs168.io  A  113.26.9.12` in the adns1.berkeley.edu name server (and not in any other name server).

Q2.3 (1 point) If no additional changes are made, EvanBot's suggestion is not allowed, because the _____ (i) domain is not in the _____ (ii) that the name server is _____ (iii) for.

Ⓐ (i): `sp26.cs168.io` (ii) cache (iii) the recursive resolver

Ⓑ (i): `sp26.cs168.io` (ii) zone (iii) authoritative

Ⓒ (i): `113.26.9.12` (ii) zone (iii) the recursive resolver

Solution:

If no additional changes are made, EvanBot's suggestion is not allowed, because the `sp26.cs168.io` domain is not in the zone that the name server is authoritative for.

The adns1.berkeley.edu name server is authoritative for the berkeley.edu zone. This means that the adns1.berkeley.edu name server should only be responsible for the authoritative copy of records in the berkeley.edu zone, i.e. records relating to domains that end in `.berkeley.edu`.

The `A` record for `sp26.cs168.io` is not in the berkeley.edu zone, and is instead in the `cs168.io` zone, so this record does not belong in the adns1.berkeley.edu name server.

Q2.4 (2 points) Which record should be added in order for EvanBot's change to be allowed?

Ⓐ `berkeley.edu NS adns1.berkeley.edu`

Ⓑ `berkeley.edu NS cs168-dns.io`

Ⓒ `sp26.cs168.io NS adns1.berkeley.edu`

Ⓓ `sp26.cs168.io NS cs168-dns.io`

Ⓔ `berkeley.edu A 1.4.0.0`

Ⓕ `berkeley.edu A 1.5.0.0`

Ⓖ `sp26.cs168.io A 1.4.0.0`

Ⓗ `sp26.cs168.io A 1.5.0.0`

Solution:

The new record `sp26.cs168.io NS adns1.berkeley.edu` delegates authority for the sp26.cs168.io zone to the adns1.berkeley.edu name server.

Now that the adns1.berkeley.edu name server is authoritative for the sp26.cs168.io zone (in addition to the berkeley.edu zone it was already authoritative for), this name server is allowed to add the `sp26.cs168.io A 113.26.9.12` record.

Q2.5 (1 point) Where should the record in the previous subpart be added?

Ⓐ In the adns1.berkeley.edu name server. Ⓑ In the cs168-dns.io name server.

Solution:

The cs168.io name server should delegate the sp26.cs168.io part of its own zone to the adns1.berkeley.edu name server.

Author's note: The definitive source for this subpart and the previous two subparts is in RFC 2181: https://www.rfc-editor.org/rfc/rfc2181#section-6.1

"A server for a zone should not return authoritative answers for queries related to names in another zone, which includes the NS, and perhaps A, records at a zone cut, unless it also happens to be a server for the other zone."

In Q2.6 and Q2.7, suppose we implement EvanBot's suggestion and add the record in Q2.4.

Hint: If a name server is authoritative for both domains in a **`CNAME`** record, then a **`CNAME`** and **`A`** record can both be returned in a single DNS response.

Q2.6 (1 point) If EvanBot's suggestion is implemented, and a resolver starts with an empty cache, how many queries are needed to learn the IP address of `cs168.berkeley.edu`?

Ⓐ 0 Ⓑ 1 Ⓒ 2 Ⓓ 3 Ⓔ 4 Ⓕ 5 Ⓖ 6 Ⓗ 7

Solution:

As before, we need 3 queries to the root name server, .edu name server, and berkeley.edu name server.

Following the hint, the berkeley.edu name server is authoritative for both domains in `cs168.berkeley.edu CNAME sp26.cs168.io`, so it can return both the **`CNAME`** and **`A`** records in a single response.

Therefore, no further queries are needed, and we needed 3 queries in total.

Q2.7 (1 point) If EvanBot's suggestion is implemented, and a resolver starts with an empty cache, how many queries are needed to learn the IP address of `sp26.cs168.io`?

Ⓐ 0 Ⓑ 1 Ⓒ 2 Ⓓ 3 Ⓔ 4 Ⓕ 5 Ⓖ 6 Ⓗ 7

Solution:

As before, we need 3 queries to the root name server, .io name server, and cs168.io name server.

At this point, the cs168.io name server will give us EvanBot's new record:

```
sp26.cs168.io  NS  adns1.berkeley.edu
```

This tells us to go query `adns1.berkeley.edu` for the IP address of `sp26.cs168.io`.

Since we don't know the IP address of `adns1.berkeley.edu`, we need to make 2 more queries to the root name server and the .edu name server.

Now that we have the IP address of `adns1.berkeley.edu`, we make 1 final query to obtain the `A` record for `sp26.cs168.io`.

This gives us a total of $3 + 2 + 1 = 6$ queries.

Note: An earlier version of the solutions incorrectly marked 4 as the answer, because we forgot to include the two queries needed to learn the IP address of the `adns1.berkeley.edu` name server. The only correct answer is 6.

Q2.8 and Q2.9 are independent of the earlier subparts.

Q2.8 (3 points) Provide two records that could cause a DNS query to encounter an infinite loop.
In each empty box, write a domain (e.g. a.com), or an IP address (e.g. 1.1.1.1), or a record type (e.g. `A`).

	Name	Type	Value
Record 1:	a.com	CNAME	b.com
Record 2:	b.com	CNAME	a.com

Solution: This creates an infinite loop where a query to a.com triggers a query for b.com, which triggers a query for a.com, which triggers a query for b.com, and so on.

Q2.9 (2 points) The DNS specification requires that if a `CNAME` record exists for a domain, then no other records should exist with that same domain.

For example, if `a.com CNAME b.com` exists, then `a.com A 1.2.3.4` cannot exist.

Which option best describes the problem that would occur if somebody broke this rule?

(A) One domain could have two IP addresses, which is not allowed.

(B) Two domains could have the same IP address, which is not allowed.

(C) Two domains could be aliases, but map to different sets of IP addresses, which is not allowed.

(D) Two domains could be aliases, but map to the same IP addresses, which is not allowed.

Solution:

By definition, if two domains are aliases, looking either one up should produce equivalent results.

However, if we break the rule in this subpart, then it could be the case that looking up `a.com` gives IP address `1.2.3.4`, while looking up `b.com` does not give this IP address. This corresponds to option (C).

(A) is false, because it is legal in DNS for one domain to have two IP addresses (e.g. consider load-balancing across mirror servers).

(B) is false, because two domains can have the same IP address (e.g. consider hosting two websites on one computer).

(D) is false, because by definition, we do want two aliased domains to map to the same IP addresses (e.g. consider the examples earlier in this question).

CodaBot runs the `www.berkeley.edu` website, and wants to load-balance requests across 9 CDN servers with domains `cdn-1.berkeley.edu`, `cdn-2.berkeley.edu`, ..., `cdn-9.berkeley.edu`.

For the remaining subparts, assume the user directly makes DNS queries, i.e. ignore stub and recursive resolvers.

Q2.10 (1 point) When a user makes a DNS query for `www.berkeley.edu`, what DNS record should be returned to achieve CodaBot's load-balancing goal?

Assume `i` is a random number between 1 and 9, generated once per DNS request.

(A) `cdn-i.berkeley.edu CNAME www.berkeley.edu`

(B) `cdn-i.berkeley.edu NS www.berkeley.edu`

(C) `www.berkeley.edu CNAME cdn-i.berkeley.edu`

(D) `www.berkeley.edu NS cdn-i.berkeley.edu`

Solution: We want to redirect users from the origin server (`www.berkeley.edu`), to one of the CDN servers (`cdn-i.berkeley.edu`).

Q2.11 (1 point) Is it possible to include an additional `A` record in the DNS response, so that the user only needs to make a single DNS query to access the website?

Ⓐ Yes, the response can include an `A` record for `www.berkeley.edu`.

Ⓑ Yes, the response can include an `A` record for `cdn-i.berkeley.edu`.

Ⓒ No, because DNS queries cannot include additional records.

Ⓓ No, because the name server sending the response is not authoritative for the `A` record.

Ⓔ No, because an additional query would still be needed, even if the `A` record is included.

Solution:

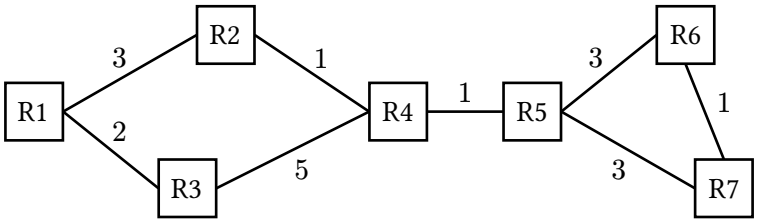
The `www.berkeley.edu CNAME cdn-i.berkeley.edu` record is returned by the berkeley.edu name server.

This name server is also authoritative for the `cdn-i.berkeley.edu A 1.2.3.4` record, so the response can also include this record.

By supplying both of these records in a single response, the resolver knows the IP address used to access the website.

Q3 *STP* **(10 points)**

(0.5 points per blank) Consider running the Spanning Tree Protocol (STP) on the topology below.



Suppose R1 is elected as the root. After running STP, in the left cycle, _____ (Q3.1) disables its link to _____ (Q3.2), and in the right cycle, _____ (Q3.3) disables its link to _____ (Q3.4).

Q3.1: Ⓐ R1 Ⓑ R2 Ⓒ R3 Ⓓ R4

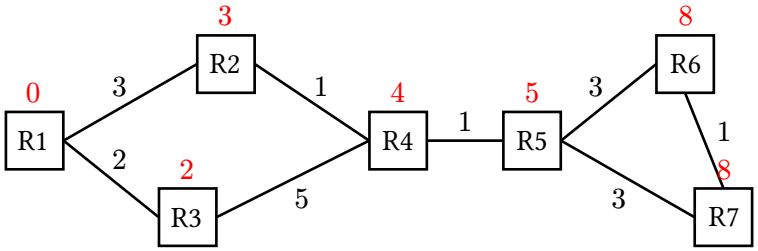
Q3.2: Ⓐ R1 Ⓑ R2 Ⓒ R3 Ⓓ R4

Q3.3: Ⓐ R5 Ⓑ R6 Ⓒ R7

Q3.4: Ⓐ R5 Ⓑ R6 Ⓒ R7

Solution:

The numbers in red show the distance from the root R1:



In the left cycle, R4 has two links pointing toward the root. The path through R2 has cost $1 + 3 = 4$, while the path through R3 has cost $5 + 2 = 7$.

Therefore, the link from R4 to R3 gets blocked.

In the right cycle, R6 and R7 are both distance 8 from the root. Since neither is further than the other, a tiebreaker would be needed to determine whether R6 disables its link to R7, or vice-versa (i.e. R7 disables its link to R6).

Since no tiebreaker was specified in the question, points were given for either answering R6 (Q3.3) and R7 (Q3.4), or R7 (Q3.3) and R6 (Q3.4).

Starting from the original topology again, suppose R7 is elected as the root instead. After running STP, in the left cycle, _____ (Q3.5) disables its link to _____ (Q3.6), and in the right cycle, _____ (Q3.7) disables its link to _____ (Q3.8).

Q3.5: Ⓐ R1 Ⓑ R2 Ⓒ R3 Ⓓ R4

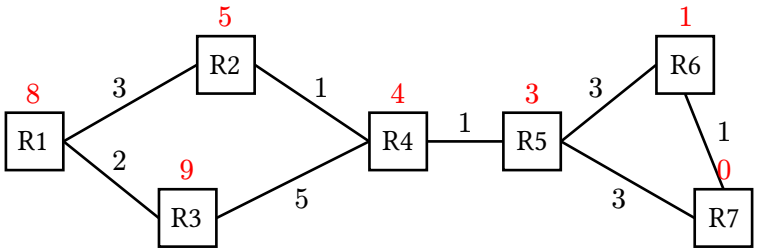
Q3.6: Ⓐ R1 Ⓑ R2 Ⓒ R3 Ⓓ R4

Q3.7: Ⓐ R5 Ⓑ R6 Ⓒ R7

Q3.8: Ⓐ R5 Ⓑ R6 Ⓒ R7

Solution:

The numbers in red show the distance from the root R7:



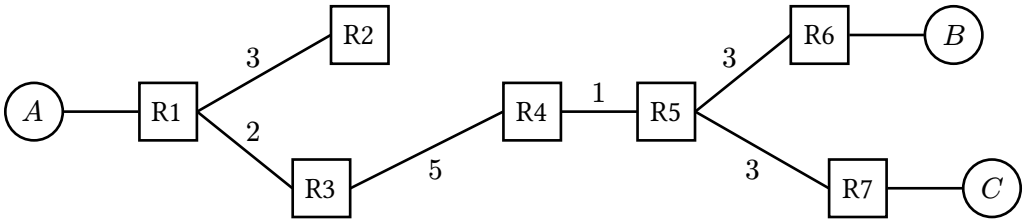
In the left cycle, R3 has two links pointing toward the root. The path through R4 has cost $5 + 4 = 9$, while the path through R1 has cost $2 + 8 = 10$.

Therefore, the link from R3 to R1 gets blocked.

In the right cycle, R5 has two equal-cost paths to the root. The path through R7 has cost $3 + 0 = 3$, and the path through R6 has cost $3 + 1 = 4$.

Therefore, the link from R5 to R6 gets blocked.

Regardless of your previous answers, suppose the network has converged on this spanning tree:



Switches R1 to R7 are all learning switches. All forwarding tables start out empty, except R5, whose table has a hard-coded entry with the least-cost routes to B.

In Q3.9 to Q3.11, 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.

Q3.9 (2 points) A sends a packet to B.

A R1 B R2 C R3 D R4 E R5 F R6 G R7

Solution:

- A sends the packet to R1.
- R1 floods the packet to both R2 and R3.
- R3 floods the packet to R4.
- R4 floods the packet to R5.
- R5 has a hard-coded entry, so it forwards the packet to R6.
- R6 sends the packet to B.

Q3.10 (2 points) C sends a packet to A.

[A] R1 [B] R2 [C] R3 [D] R4 [E] R5 [F] R6 [G] R7

Solution: Note that at this point, R1, R2, R3, R4, R5, R6 all have next-hops to A, since in the previous subpart, they received a packet from A.

- C sends the packet to R7.
- R7 floods the packet to R5.
- R5 forwards the packet to R4.
- R4 forwards the packet to R3.
- R3 forwards the packet to R1.
- R1 sends the packet to A.

Q3.11 (2 points) B sends a packet to C.

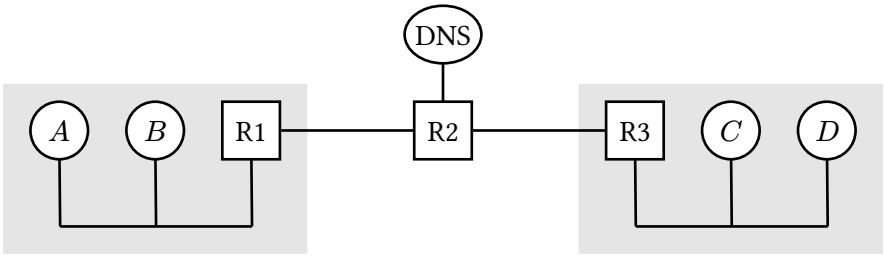
[A] R1 [B] R2 [C] R3 [D] R4 [E] R5 [F] R6 [G] R7

Solution: Note that at this point, R1, R3, R4, R5, R7 all have next-hops to C, since in the previous subpart, they received a packet from C.

- B sends the packet to R6.
- R6 floods the packet to R5.
- R5 forwards the packet to R7.
- R7 sends the packet to C.

Q4 *End-to-End* **(13 points)**

In this question, consider the network topology below, with A, B, R1 in a subnet connected with a single shared medium, and R3, C, D, in another subnet connected with a single shared medium. Everyone uses the same DNS resolver.



A joins the network and types `http://www.d.com` in its browser to load a webpage from D.

Q4.1 (2 points) Host A runs DHCP and learns that its own IP is 10.1.1.3, the router IP is 10.1.1.2, and the subnet mask is /24.

If Host B later joins the network and runs DHCP (while Host A is still on the same DHCP lease), what could be the IP address assigned to Host B? Select all that apply.

[A] 10.1.1.0 [B] 10.1.4.0 [C] 10.1.1.3 [D] 10.4.1.0 [E] None

Solution:

The intended answer was (A) only.

Note that the subnet mask is /24, which means the top 24 bits must be 10.1.1.x. This rules out 10.1.4.0 and 10.4.1.0.

10.1.1.3 cannot be used, since it is already in use for Host A.

This leaves 10.1.1.0 as the only valid choice, since its top 24 bits match the subnet prefix, and it is not already in use by another host.

However, after the exam, some students offered an alternate interpretation of the question wording, where the answer would be **None**.

In practice, gateway routers will usually, but not always, keep the "network address" reserved, and will not assign this address to hosts. The *network address* is the IP address with the network ID bits set to the subnet mask, and the host ID bits all set to zero. In this question, the network address is exactly 10.1.1.0, so under this interpretation, 10.1.1.0 should not be assigned to any host.

This fact was not in scope for the class, but we gave students benefit of the doubt and granted points for answering "None."

In Q4.2 to Q4.4, consider a packet sent by A during the HTTP connection.

Q4.2 (1 point) Which protocol was used to learn the value of the Source MAC field?

Ⓐ DHCP Ⓑ ARP Ⓒ DNS Ⓓ TCP Ⓔ HTTP Ⓕ None

Solution: The source MAC address is burned into hardware, and none of these protocols are needed to learn its value.

Q4.3 (1 point) Which protocol was used to learn the value of the Destination MAC field?

Ⓐ DHCP Ⓑ ARP Ⓒ DNS Ⓓ TCP Ⓔ HTTP Ⓕ None

Solution: The destination MAC is the router's MAC address. A must perform an ARP lookup to convert the router's IP address to the corresponding MAC address.

Q4.4 (1 point) Which layer's header is used in computing the sequence number of the next packet that A sends out?

Ⓐ Layer 7 Ⓑ Layer 4 Ⓒ Layer 3 Ⓓ Layer 2 Ⓔ None

Solution: The Layer 4 (TCP) header contains an ack number, which indicates the next sequence number that A should send out.

Q4.5 (1 point) In total, how many HTTP requests were sent by A in the process of loading the webpage?

Ⓐ Exactly 1, because A typed only one URL in its browser.

Ⓑ Exactly 2, because the DNS request added one more HTTP request.

Ⓒ 1 or more, because the webpage HTML could trigger more requests.

Ⓓ 2 or more, because of both the DNS request and the webpage HTML triggering more requests.

Solution:

Note that the DNS request is sent over UDP, so it does not count as an HTTP request.

The webpage HTML could trigger additional requests. For example, the HTML could include something like `<img src="www.d.com/logo.png">`, which requires making another HTTP request to fetch the image.

Q4.6 (1 point) True or false: If HTTP pipelining is **disabled**, then A initiates exactly 1 TCP handshake in the process of loading the webpage.

Ⓐ True Ⓑ False

Solution: False. If the webpage HTML triggers a second HTTP request, then a second TCP handshake would be needed, since pipelining is disabled.

Q4.7 (1 point) True or false: If HTTP pipelining is **enabled**, then A initiates exactly 1 TCP handshake in the process of loading the webpage.

Ⓐ True Ⓑ False

Solution: False. Pipelining avoids a new TCP handshake if the HTTP request is being made to the same server. However, if an HTTP request to a separate server is made, then a new TCP handshake would have to be made.

For example, consider something like `<img src="www.google.com/logo.png">` in the HTML. A has an open pipelined TCP connection to `www.d.com`, but in order to make an HTTP request to `www.google.com`, a separate TCP connection must be initiated to the Google server.

Q4.8 (2 points) For this subpart only, suppose R1 is using NAT (as shown in lecture) in its subnet.

Recall that NAT assigns private addresses to hosts inside the network. Which device(s) can deduce that NAT is in use by seeing a private address in the header of a packet (possibly a packet it is not supposed to be reading)? Select all that apply.

[A] A [B] B [C] C [D] DNS [E] D Ⓕ None

Solution:

A notices that packets it receives are addressed to a private address, and therefore can deduce that NAT is in use.

Since B is on the same shared medium as A, we know that B can see all packets sent/received by A. Therefore, B will also see that packets sent to A are addressed to a private address, so B can also deduce that NAT is in use.

C, D, and DNS will not see any private addresses, because by the time R1 forwards any packet, all private addresses have been swapped out for public addresses.

The rest of the subparts are independent of earlier subparts, i.e. we start over with empty caches.

Now, A joins the network and enters `https://www.d.com` to form an HTTPS connection to D.

Here are some reminders about TLS from lecture:
- HTTPS is a Layer 7 protocol running on top of TLS.
- TLS is a Layer 4.5 protocol running on top of TCP.
- TLS encrypts its entire payload, and adds an unencrypted header.

Q4.9 (1 point) When A visits `https://www.d.com`, in what order does A run these protocols?

Ⓐ TCP handshake, then TLS handshake, then DNS lookup.

Ⓑ DNS lookup, then TCP handshake, then TLS handshake.

Ⓒ DNS lookup, then TLS handshake, then TCP handshake.

Ⓓ TLS handshake, then TCP handshake, then DNS lookup.

Solution:

The DNS lookup must come before the TCP handshake. We need to first know the IP address of `www.d.com` (using DNS) before we can initiate a TCP connection to `www.d.com`.

The TCP handshake must come before the TLS handshake, because TLS runs on top of TCP. In other words, the TLS handshake relies on a TCP bytestream in order to exchange data reliably.

Q4.10 (2 points) During the TLS connection, what values can R2 learn, possibly by reading packets it is not supposed to be reading? Select all that apply.

- [A] R2 can read the records in the DNS response sent before the TLS connection.
- [B] R2 can learn A's IP address.
- [C] R2 can learn D's IP address.
- [D] R2 can learn the TCP initial sequence numbers.
- [E] R2 can read the HTTPS payloads.
- [F] R2 can read the status codes in the HTTP responses (e.g. 404 File Not Found).
- Ⓖ None of the above

Solution:

Everything inside the TLS payload is encrypted. In other words, everything that comes *after* the TLS header is encrypted, but everything *before* the TLS header is unencrypted:

L3: IP Header
L4: TCP Header
L4.5: TLS Header
L7: HTTPS Header (encrypted)
HTTPS Payload (encrypted)

(A): True. The DNS request/response are separate from the TLS connection, and they are sent over UDP, so they are not encrypted.

(B)-(C): True. The IP addresses are in the L3 IP header, outside the TLS payload, and therefore unencrypted.

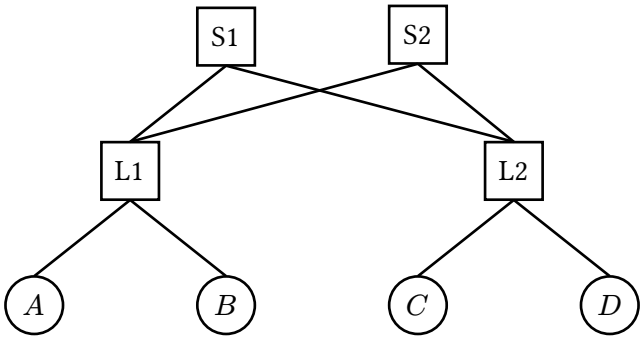
(D): True. The TCP sequence numbers are in the L4 TCP header, which is unencrypted.

(E): False. The HTTPS payload is inside the TLS payload and therefore encrypted.

(F): False. The HTTPS status codes are in the L7 HTTPS header, which are inside the TLS payload, and therefore encrypted.

Q5 *Datacenters* **(11 points)**

Consider the topology below. All links have a bandwidth of 1 Gbps.



Q5.1 (3 points) Server A hosts VM 1 (Overlay: 10.0.0.1, ID: 50) and Server D hosts VM 2 (Overlay: 10.0.0.2, ID: 50).

Suppose an attacker compromises S1 and runs a packet sniffer. If VM 1 sends a packet to VM 2, which of the following addresses will the attacker observe in the captured packet's headers? Select all that apply.

- [A] Underlay IP of A
- [B] Underlay IP of D
- [C] Overlay IP of VM 1 (10.0.0.1)
- [D] Overlay IP of VM 2 (10.0.0.2)
- [E] Underlay MAC address of A
- [F] Underlay IP of L1
- [G] None of the above

Solution:

When Server A encapsulates the packet, the outer (underlay) headers contain the IPs of the physical source (Server A) and destination (Server D). The inner (overlay) headers contain the IPs of the VMs.

The underlay MAC address of Server A is stripped by Leaf L1 when the packet is routed at Layer 3 towards the spine. The underlay IP of L1 is not in the packet because routers do not rewrite the source IP of routed packets.

Q5.2 (3 points) VM 1 (ID: 50) is moved from A to C.

Assuming an SDN-managed overlay network, which of the following *must* occur for VM 1 to successfully resume receiving packets at its new location? Select all that apply.

- [A] VM 1 must broadcast a DHCP request to obtain a new overlay IP address.
- [B] Underlay routers (L1, L2, S1, S2) must recalculate routing tables to locate VM 1's overlay IP.
- [C] The virtual switch on C must be updated with a new forwarding rule for VM 1.
- [D] The SDN controller must update the mapping of VM 1's overlay IP to C's underlay IP.
- [E] The virtual switch on A must forward all future datacenter traffic to C.
- (F) None of the above

Solution: (A): False. The core benefit of overlays is that VMs retain their overlay IPs regardless of physical location.

(B): False. Underlay routers have no knowledge of overlay IPs. They only route to physical server IPs.

(C) & (D): True. The SDN controller tracks the location of all VMs. When VM 1 moves, the controller updates the global mapping and installs the necessary encapsulation rules on the new host's vSwitch.

(E): False. Server A only needs to drop its rule for VM 1. It does not become a permanent proxy for all datacenter traffic.

Q5.3 (2 points) Suppose the switches in the network use 5-tuple ECMP for load balancing.

VM 1 on A needs to transfer a massive dataset to VM 2 on D. The application opens **two** concurrent TCP connections to maximize throughput.

Assuming no other traffic in the network, what is the throughput the application will achieve, and why?

(A) 2 Gbps, because ECMP will route one connection through S1 and the other through S2.

(B) 1 Gbps, because ECMP hashes both connections to the same path since they are between the same source-destination pair.

(C) It has a 50% chance of achieving 2 Gbps and a 50% chance of achieving 1 Gbps, depending on the ECMP hash results.

(D) None of the above

Solution:

(A) and (C) are false, because the links to the hosts (A-to-L1 and L2-to-D) bottleneck the connection to 1 Gbps.

(B) is false, because even though the connection is bottlenecked at 1 Gbps, the two connections could still be hashed onto different paths, since the two connections have different randomly-chosen source ports.

Therefore, (D) is the correct answer. The throughput achieved is 1 Gbps, but the traffic has a 50% chance of taking the same path and a 50% chance of taking two different paths.

An earlier version of the solutions incorrectly marked (C) as the answer, because we forgot to mark that the links to the hosts have infinite bandwidth (which was our original intention). Based on the question as printed, (D) is the only correct answer.

Q5.4 (2 points) Suppose we want to run a collective operation among four VMs, one located on each of the four servers (A, B, C, D).

To optimize performance, the VMs will logically arrange themselves into a ring topology to pass data. Which logical ring minimizes traffic across the network?

If multiple options are equally optimal, select all the optimal options.

[A] $A \to B \to C \to D \to A$

[B] $A \to C \to B \to D \to A$

[C] $A \to C \to D \to B \to A$

Ⓓ None of the above

Solution: Communication between A and B stays local to Leaf L1. Communication between C and D stays local to Leaf L2.

Communication between any node in {A, B} and any node in {C, D} must traverse the spine, which is a bottleneck.

The goal is to minimize spine layer traversals.

- Option A has 2 spine crossings (B → C, and D → A) and 2 local links (A → B, C → D).
- Option B has 4 spine crossings (A → C, C → B, B → D, D → A).
- Option C has 2 spine crossings (A → C, D → B) and 2 local links (C → D, B → A).

Both Option A and Option C yield identical network footprints with exactly two local traversals and two spine traversals. They tie for mathematically optimal performance.

Q5.5 (1 point) What is the bisection bandwidth of this topology, in Gbps?

Ⓐ 1 Ⓑ 2 Ⓒ 3 Ⓓ 4 Ⓔ 6 Ⓕ 8

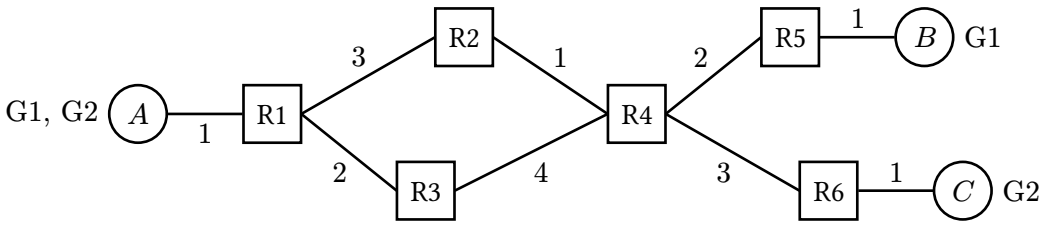
Solution: Bisection bandwidth is defined as the minimum capacity of a set of links that must be removed to partition the network's endpoints (servers) into two equal halves.

To divide the 4 servers into two equal halves (e.g., Partition 1: {A, B} and Partition 2: {C, D}), we evaluate the minimum cut isolating L1 from L2. Cutting the links L1-S1 and L1-S2 isolates L1.

Since this requires cutting exactly 2 links, each at 1 Gbps, the bisection bandwidth is 2 Gbps. (Note: A common misconception is to add all four leaf-to-spine links, which would represent disconnecting **all** leaves from the spines, not dividing the network in half).

Q6 *Multicast* **(7 points)**

Consider running DVMRP on the topology below. Each host belongs to group G1, or group G2, or both.



Fill in the DVMRP multicast table at R4, **before any pruning** takes place. Each row is worth 0.5 points.

- For example, if packets from A to G1 are forwarded to R2 and R5, select "R2" and "R5" in the first row.
- If a row should not be included in the table, select "None" for that row.

	R4's Multicast Table Before Pruning						
	Source	**Destination**	**Next-hop(s)**				
Q6.1	A	G1	A R2	B R3	C R5	D R6	E None
Q6.2	A	G2	A R2	B R3	C R5	D R6	E None
Q6.3	B	G1	A R2	B R3	C R5	D R6	E None
Q6.4	B	G2	A R2	B R3	C R5	D R6	E None
Q6.5	C	G1	A R2	B R3	C R5	D R6	E None
Q6.6	C	G2	A R2	B R3	C R5	D R6	E None

Solution:

Note that before any pruning, we have one tree per source, that shows the shortest paths from that source to every other destination.

Thus, the first two rows have the same answer (tree from A to all destinations),
and the next two rows have the same answer (tree from B to all destinations),
and the last two rows have the same answer (tree from C to all destinations).

To build the tree from A to all destinations, we can draw the directed delivery tree towards A, and then reverse the arrows. Then, we can repeat for B and C.

Note: An earlier version of the solutions incorrectly forgot to mark R3 in the last four entries of the table. We do need to include R3 in the spanning tree touching all destinations, because we don't know yet if R3 has any connected hosts that belong to the group. (We will only later discover, using pruning, that R3 has no connected hosts belonging to any group.) Alternatively, if you consider the directed delivery tree towards a destination (A for first two rows, B for next two rows, C for last two rows), R3 is on that tree, because R3 has forwarding entries in its unicast routing table toward those destinations. Therefore, in the trees pointing toward B and C, R3 ("child") will send a message toward R4 ("parent"), which causes R4 to send packets toward R3.

Next, fill in the DVMRP multicast table at R4, **at convergence**. Each row is worth 0.5 points.

	R4's Multicast Table at Convergence						
	Source	**Destination**	**Next-hop(s)**				
Q6.7	A	G1	[A] R2	[B] R3	[C] R5	[D] R6	(E) None
Q6.8	A	G2	[A] R2	[B] R3	[C] R5	[D] R6	(E) None
Q6.9	B	G1	[A] R2	[B] R3	[C] R5	[D] R6	(E) None
Q6.10	B	G2	[A] R2	[B] R3	[C] R5	[D] R6	(E) None
Q6.11	C	G1	[A] R2	[B] R3	[C] R5	[D] R6	(E) None
Q6.12	C	G2	[A] R2	[B] R3	[C] R5	[D] R6	(E) None

Solution:

A-to-G1: The only other G1 member is B, and the next-hop to B is R5.

A-to-G2: The only other G1 member is C, and the next-hop to C is R6.

B-to-G1: The only other G1 member is A, and the next-hop to A is R2.

B-to-G2: The G2 members are A and C, and the next-hops are R2 and R6, respectively.

C-to-G1: The G1 members are A and B, and the next-hops are R2 and R5, respectively.

C-to-G2: The only other G2 member is A, and the next-hop to A is R2.

Q6.13 (1 point) Regardless of your answers to the previous parts, suppose you filled in 20 options in the first table, and 15 options in the second table. Based on these numbers, how many distinct pruning messages were sent? (Ignore periodically resending messages.)

Ⓐ 0 Ⓑ 5 Ⓒ 10 Ⓓ 15 Ⓔ 20 Ⓕ Not enough information

Solution:

The provided table values (20 and 15) give us sufficient information to determine the number of prune messages arriving *at R6*, but they do not tell us how many prune messages were sent elsewhere in the network. Therefore, the best answer is "Not enough information."

Note: An earlier version of the solutions incorrectly marked 5 as the correct answer, under the assumption that the question was asking about pruning messages arriving *at R6*. The reasoning for this answer is below:

When a child tells a parent "prune me," the parent deletes the child from its list of next-hops. Therefore, each pruning message causes one entry in the multicast table to be deleted.

If we had 20 total next-hops in the initial table, and 15 next-hops after pruning, then this means that $20 - 15 = 5$ entries were pruned, so 5 pruning messages arrived at R6.

Note that the actual number of prune messages arriving at R6 is $16 - 8 = 8$ pruning messages, though the question asks you to use the given numbers instead.

To give students benefit of the doubt, we gave partial credit for the answer 5.

Q7 *Collectives* **(11 points)**

Consider the following computation problem on N nodes:

Inputs:
- Node 1 has an array of N elements, where every element is either `include` or `exclude`.
- The remaining nodes (2 through N) each have an N-integer array.

Outputs:
- If the ith element of Node 1 is `exclude`, then the corresponding element in the result should be 0.
- If the ith element of Node 1 is `include`, then the corresponding element in the result should be the **product** of the ith elements of every other node.
- After the operation completes, **every node** should have a copy of the N-integer result array.

An example with $N = 4$ is shown below. Your answers should work for any input, not just the example.

Node 1	Node 2	Node 3	Node 4		Result
`exclude`	7	9	5		0
`include`	2	3	10	→	60
`exclude`	5	1	4		0
`include`	3	4	2		24

To complete this operation, you will first need to convert the values in Node 1 to integers.

Q7.1 (1 point) What integer value should each `exclude` be converted to?

Ⓐ $-N$ Ⓑ -1 **Ⓒ 0** Ⓓ 1 Ⓔ $N-1$ Ⓕ N

Solution: Multiplying 0 into the product will zero out the result.

Q7.2 (1 point) What integer value should each `include` be converted to?

Ⓐ $-N$ Ⓑ -1 Ⓒ 0 **Ⓓ 1** Ⓔ $N-1$ Ⓕ N

Solution: Multiplying 1 into the product leaves the result unchanged (without zeroing out the result).

Q7.3 (1 point) After converting the values in Node 1 to integers, which collective operation can be used to complete the operation?

Ⓐ Broadcast Ⓒ Gather Ⓔ Reduce Ⓖ ReduceScatter

Ⓑ Scatter Ⓓ AllGather **Ⓕ AllReduce**

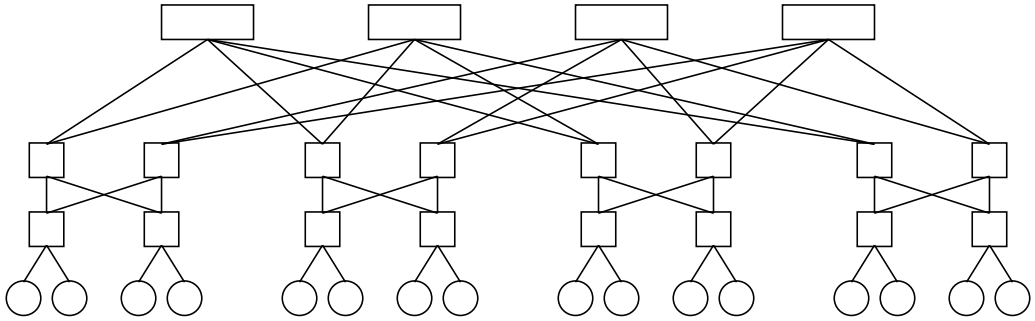
Solution: AllReduce can be used to take the element-wise product of all the vectors, and send a copy of the result to all nodes.

The rest of the question is independent of the earlier subparts.

Consider a k-ary fat-tree (folded Clos) topology, where:

- k is even,
- there are k pods,
- each pod contains $\left(\frac{k}{2}\right)^2$ hosts,
- each host has uplink bandwidth R,
- and the pods are split evenly into a left half and a right half.

As a reminder, the $k = 4$ topology from lecture is shown below:



Q7.4 (2 points) How many hosts are in the left half of the datacenter?

Ⓐ $\frac{k^2}{4}$ Ⓑ $\frac{k^3}{16}$ Ⓒ $\frac{k^3}{8}$ Ⓓ $\frac{k^3}{4}$ Ⓔ $\frac{k^2}{2}$

Solution: Half of the k pods are in the left half, so there are $\frac{k}{2}$ pods on the left.

Each pod contains $\left(\frac{k}{2}\right)^2$ hosts, so the total number of hosts is:

$$\left(\frac{k}{2}\right) \cdot \left(\frac{k}{2}\right)^2 = \frac{k^3}{8}$$

Q7.5 (2 points) During one phase of a collective operation, the datacenter runs an all-to-all communication pattern: every host sends one packet to every other host in the datacenter.

In this all-to-all phase, how many packets cross from the left half of the datacenter to the right half?

Ⓐ $\frac{k^3}{8}$ Ⓑ $\frac{k^3}{4}$ Ⓒ $\frac{k^6}{128}$ Ⓓ $\frac{k^6}{64}$ Ⓔ $\frac{k^6}{32}$

Solution: There are $\frac{k^3}{8}$ hosts in the left half and $\frac{k^3}{8}$ hosts in the right half.

In an all-to-all pattern, each host in the left half sends one packet to each host in the right half.

So the total number of packets crossing from left to right is:

$$\left(\frac{k^3}{8}\right) \times \left(\frac{k^3}{8}\right) = \frac{k^6}{64}$$

Q7.6 (2 points) For this subpart only, assume the topology's bisection bandwidth is $\left(\frac{k^3}{8}\right)R$.

If one link of bandwidth R that crosses the bisection cut fails, what is the new worst-case oversubscription ratio of the topology?

Ⓐ $1:1$ Ⓑ $k^2:(k^2-4)$ Ⓒ $k^3:(k^3-8)$ Ⓓ $k^3:(k^3-4)$

Solution: The failed link reduces the bandwidth of the worst-case cut from $\left(\frac{k^3}{8}\right)R$ to $\left(\left(\frac{k^3}{8}\right)-1\right)R$.

So the new worst-case oversubscription ratio is:

$$\frac{\left(\frac{k^3}{8}\right)R}{\left(\left(\frac{k^3}{8}\right)-1\right)R} = \frac{k^3}{k^3-8}$$

Therefore, the ratio is $k^3:(k^3-8)$.

Q7.7 (2 points) Regardless of your previous answers, assume a full k-ary fat-tree can support $\frac{k^3}{4}$ hosts with no oversubscription.

What is the smallest even value of k that can support at least 200 hosts?

Ⓐ 4 Ⓑ 6 Ⓒ 8 Ⓓ 10

Solution: We need:

$$\frac{k^3}{4} \geq 200$$

Multiplying both sides by 4 gives:

$$k^3 \geq 800$$

Since $10^3 > 800$ and $8^3 < 800$, the smallest even value of k is 10.

Q8 *Potpourri* **(8 points)**

Q8.1 (1 point) In Project 3 (Transport), why did we call `set_pending_ack()` instead of immediately sending out the ack?

(A) To support piggybacking.

(B) To support pipelining.

(C) To implement flow control.

(D) To pick random initial sequence numbers.

Solution:

(A) is correct. Piggybacking means we don't send the ack right away. Instead, we wait until we have more data to send, and we send both the ack and the data together in a single TCP packet.

(B) is incorrect. Pipelining is used to send multiple requests/responses in a single TCP connection, and it is implemented at the HTTP layer, not the TCP layer.

(C) is incorrect. Flow control is implemented by setting and sending RWND values, and the `set_pending_ack()` function is unrelated to RWND.

(D) is incorrect. The `set_pending_ack()` function is unrelated to setting up initial sequence numbers.

Q8.2 (2 points) In Project 3 (Transport), Stage 3.2 processed any in-order segments from the receive queue. What does this line of code (provided in the starter code as a hint) do?

```
data = packet.app[self.rcv.nxt |MINUS| packet.tcp.seq:]
```

(A) Extracts all of the payload in the TCP packet.

(B) Extracts the TCP payload, discarding bytes overlapping with already-received bytes.

(C) Checks if the next packet in the receive queue is in-order.

(D) Removes bytes from the receive queue if they overlap with already-received bytes.

Solution:

`self.rcv.nxt` is the next byte we haven't received yet.

`packet.tcp.seq` is the first byte in the packet.

By slicing out the first `nxt-seq` bytes, we remove any bytes at the start of the packet that overlap with already-received bytes.

(A) is false since we aren't extracting the entire payload, only the non-overlapping part of the payload.

(C) and (D) is false because the line of code doesn't access `self.rx_queue`, the receive queue.

Q8.3 (2 points) If we use Core-Based Trees (CBT) for multicast routing, which columns are needed in a router's multicast routing table? Select all that apply.

[A] Source [B] Destination group [C] Next-hops (D) None

Solution: CBT builds one tree per destination group, so we need to map each destination group to a set of next-hops.

Note that the source column is not needed in CBT. (By contrast, in DVMRP, we build one tree from each source to each destination group, so the source column is needed.)

Q8.4 (1 point) True or false: When RDMA completes a data transfer, an interrupt is triggered, which tells the recipient's CPU to process the received data.

(A) True (B) False

Solution: False. RDMA asynchronously adds a Completion Queue Element (CQE), and the server has to check the CQE to see that the job is completed.

Q8.5 (2 points) Which of the following benefits does RDMA provide? Select all that apply.

[A] Lower latency

[B] Easier deployment of new protocols

[C] Reduced CPU usage for host networking

(D) None of the above

Solution: By offloading the transport protocol to NIC hardware RDMA lowers latency and CPU usage for packet processing. However, it reduces flexibility since protocols can only be modified by hardware vendors.)

Comment Box

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

Nothing in the comment box will affect your grade.

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.