

## 1 True or False

- 1.1 IP multicast is widely deployed across different domains for global Internet communication.

False: IP multicast is primarily used within a single domain, such as enterprise networks or ISPs, and is rarely deployed across multiple autonomous systems due to challenges like scalability, billing disputes, and inter-domain coordination required for protocols like PIM and MSDP.

- 1.2 In the DVMRP protocol, the multicast forwarding table requires one entry per source, per multicast group.

True: DVMRP builds a source-specific shortest-path tree for each (source, multicast group) pair, requiring multicast forwarding table entries that specify the Reverse Path Forwarding (RPF) interface and outgoing interfaces (children) for each pair, updated via flood-and-prune mechanisms.

- 1.3 Core-Based Trees (CBT) guarantee that packets are forwarded along the least-cost paths to all group members.

False: CBT uses a single shared tree per multicast group rooted at a core router, prioritizing scalability over optimality. Paths depend on the core's location and may be suboptimal compared to shortest unicast paths, unlike source-specific trees in protocols like DVMRP.

- 1.4 Overlay multicast requires routers to implement specialized multicast protocols, unlike IP multicast.

False: Overlay multicast operates at the application layer, with end hosts or proxy servers managing multicast routing using unicast packets. Routers only need standard unicast protocols, unlike IP multicast, which requires specialized protocols like PIM or DVMRP. Examples include P2P streaming or ALM protocols like Narada.

- 1.5 The AllReduce collective operation can be implemented efficiently using a ring topology, where each node sends and receives data to/from its neighbors.

True: Ring-based AllReduce involves nodes exchanging partial sums in a logical ring, completing in  $2(n-1)$  steps (reduce and broadcast phases) for  $n$  nodes. This reduces bandwidth compared to full-mesh approaches, though performance depends on the logical ring's mapping to the physical network. It is widely used in MPI and AI training frameworks like Horovod.

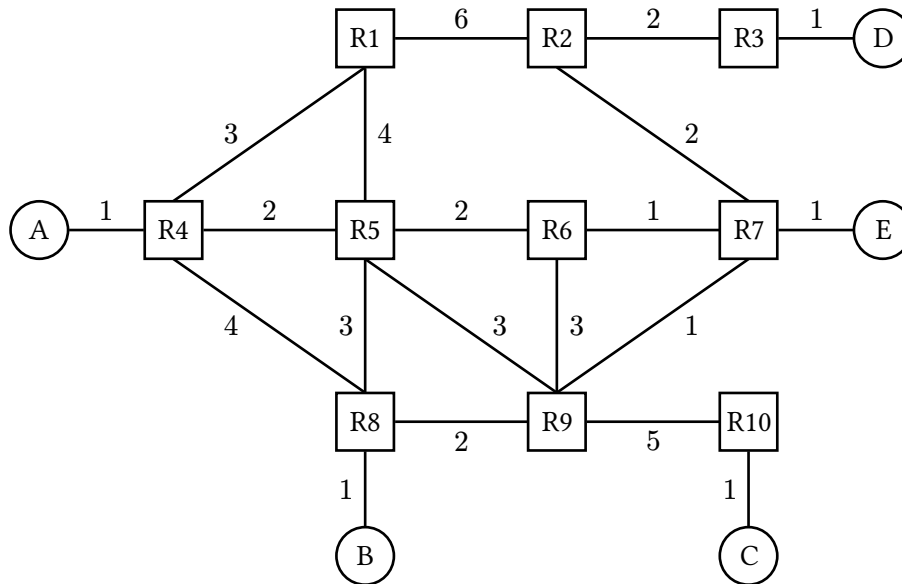
- 1.6 The stretch factor in overlay multicast measures the ratio of the overlay path cost to the underlay path cost, with lower stretch indicating better performance.

True: The stretch factor is the ratio of the overlay path cost (latency or hops in the overlay tree) to the underlay path cost (optimal unicast path). A lower stretch, closer to 1, indicates that the overlay topology closely matches the underlay, reducing latency and improving packet forwarding performance.

- 1.7 Collective operations in AI training, such as Broadcast and Reduce, are unrelated and cannot be combined to form other operations like AllReduce.

False: Broadcast and Reduce are complementary operations in collective communication. AllReduce can be implemented as a Reduce phase (aggregating data, e.g., summing gradients) followed by a Broadcast phase (distributing the result), commonly using tree-based algorithms in frameworks like MPI or Horovod.

## 2 Multicast I



Consider running DVMRP on this topology. Hosts A, B, C are members of group G1.

- 2.1 D sends a multicast packet to G1. What path does the packet take to reach B?  
 D–R3–R2–R7–R9–R8–B
- 2.2 Does the multicast path from D to B change if host E joins the group? If so, what is the new path?

The path does not change.

The remaining subparts are independent of earlier subparts.

Now, consider running CBT on this topology. R2 is the core, and group G1 initially starts out with no group members.

Note: The join message is unicast toward the root, but if an intermediate router is already on the tree, it does not need to forward the join message any further. (This is because all subsequent routers toward the root will also be on the tree.)

- 2.3 C joins group G1. What path does the join message take?

C–R10–R9–R7–R2

2.4 Next, B joins group G1. What path does the join message take?

B–R8–R9

2.5 Next, A joins group G1. What path does the join message take?

A–R4–R5–R6–R7

2.6 B sends a multicast packet to group G1. What paths does the packet take?

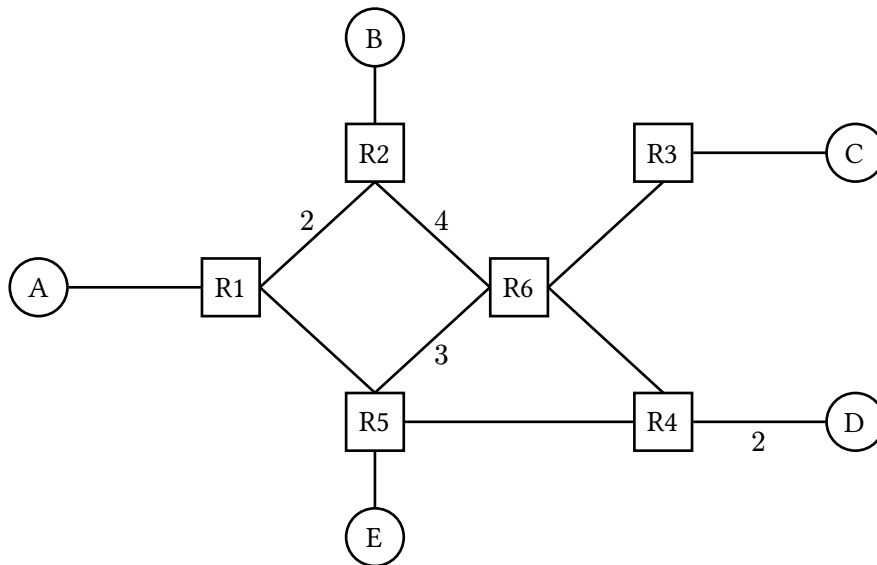
B–R8–R9. Then, R9 forwards to both R9–R10–C and R9–R7. Then R7 forwards to both R7–R2 and R7–R6–R5–R4–A.

2.7 Suppose that R3 was the core instead. B sends a multicast packet to group G1. What paths does the packet take?

B–R8–R9. Then, R9 forwards to both R9–R10–C and R9–R7. Then R7 forwards to both R7–R2–**R3** and R7–R6–R5–R4–A. (The only difference is in bold.)

### 3 Multicast II

Consider the network topology below. Link costs represent packet travel times, in seconds. Ignore packet processing or queuing delays when computing packet travel times.



Consider running DVMRP on this topology. Each subpart continues on from previous subparts.

At the start, A and E are the only members of group G1, and no other groups exist.

3.1 DVMRP runs for some time and reaches a steady-state. At steady-state, all prunable branches of multicast trees are pruned.

A sends a multicast packet to G1. What links are used to send this packet?

A–R1, R1–R5, R5–E

3.2 Now, C decides to join group G1. To do this, C runs a host-to-router protocol (e.g. IGMP).

Immediately after the host-to-router protocol reaches steady-state, which routers know that C has joined G1?

R3 only. The host-to-router protocol only notifies the first-hop router.

3.3 DVMRP runs for some time and again reaches steady-state. At steady-state, all prunable branches of multicast trees are pruned.

Now, E sends a multicast packet to group G1. What links are used to send this packet?

A–R1, R1–R5, R5–E  
R5–R4, R4–R6, R6–R3, R3–C

3.4 At this steady-state, is it possible for B to send a message to group G1?

Yes. Non-members can send messages to the group.

3.5 Now, B sends a multicast to group G1.

What route does the packet take from B to E?

B–R2–R1–R5–E

3.6 When B sends a multicast packet to group G1, how long does it take for the packet to reach all G1 members?

7 seconds. The path from B to C is the longest: B–R2–R6–R3–C

pagebreak

3.7 Now, suppose that hosts A, B, D join a new group, G2. (Remember that A, C, E are the members of G1.)

DVMRP runs for some time and again reaches steady-state. At steady-state, all prunable branches of multicast trees are pruned.

Fill out the multicast forwarding table at router R6.

From:	To:	Children:
A	G1	R3
B	G1	R3
C	G1	R4
D	G1	R3
E	G1	R3

From:	To:	Children:
C	G2	R2, R4

Some rows are omitted (e.g. from D to G2), because that tree does not involve R6 at steady-state (i.e. R6 has been pruned off that tree).

3.8 Now, these multicast packets get sent:

- A sends a packet to G1.
- A sends a packet to G2.
- B sends a packet to G2.
- D sends a packet to G1.

Who is the last host to receive any of the four packets?

The B-to-G2 packet takes 7 seconds to arrive at D.

3.9 Who is the first host to receive any of the four packets?

The A-to-G1 packet takes 3 seconds to arrive at E.

Now, consider running CBT instead on the same topology. The remainder of this question is independent of all earlier subparts. Each subpart continues on from previous subparts.

At the start, G1 has no members. R1 is selected as the core for group G1.

3.10 C sends a Join message for group G1. What routers forward this request?

R3, R6, R4, R5, R1

3.11 Now, D sends a Join message for group G1. What routers forward this request?

Note: The join message is unicast toward the root, but if an intermediate router is already on the tree, it does not need to forward the join message any further. (This is because all subsequent routers toward the root will also be on the tree.)

R4 only. Per the note, R4 is already on the tree, so it does not need to forward the packet any further. (The subsequent routers toward the root are R5 and R1, which are already part of the tree.)

3.12 Now, B sends a multicast packet to G1. How long does the packet take to reach all group members?

8 seconds. B–R2–R1 takes 3 seconds, and then R1–R5–R4–R6–R3–C takes another 5 seconds. (R1–R5–R4–D only takes another 4 seconds.)

3.13 Now, C sends a multicast packet to G1. How long does the packet take to reach all group members?

5 seconds. C–R3–R6–R4–D.

3.14 Now, the A–R1 link goes down. Who can still send messages to G1?

B, C, D, E can still send messages to G1.

3.15 Which G1 members will still receive messages sent to G1?

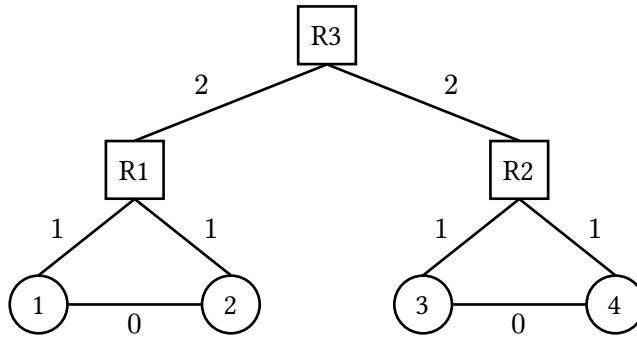
6 *Multicast and Collectives*

C, D can both still receive messages sent to G1.

# 4 Collectives

Consider the **underlay** topology below.

- Link costs represent packet travel time, in seconds.
- Ignore any processing and queuing delays.
- Assume links have very high bandwidth, i.e. packet travel time is the only source of delay.



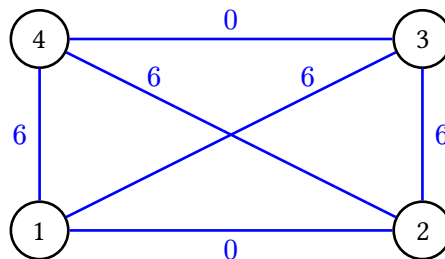
4.1 In this topology, some links are labeled with cost 0. Why might this be a reasonable model for an AI training datacenter network?

Hosts 1 and 2 might be on the same physical machine (e.g. two GPUs in the same physical server). This means they can communicate without using the network at all, which effectively gives infinite bandwidth and zero packet travel time.

We want to perform an AllReduce operation on nodes 1, 2, 3, 4. Each node has a 4-Gbit vector, and each vector is split into four 1-Gbit vector elements.

4.2 Consider using a **full mesh** overlay topology to run the AllReduce operation.

What does the **overlay** topology look like? Draw virtual links in the diagram below. Then, label each virtual link with its packet travel time (according to the underlay network).



4.3 To execute AllReduce with the full-mesh overlay, how much data is sent by each node? How much data is sent in total?

**12 Gbit per node.** Each node must send its full vector, which is 4 Gbit, to the three other nodes.

**48 Gbit total,** because there are 4 nodes, each sending 12 Gbit.

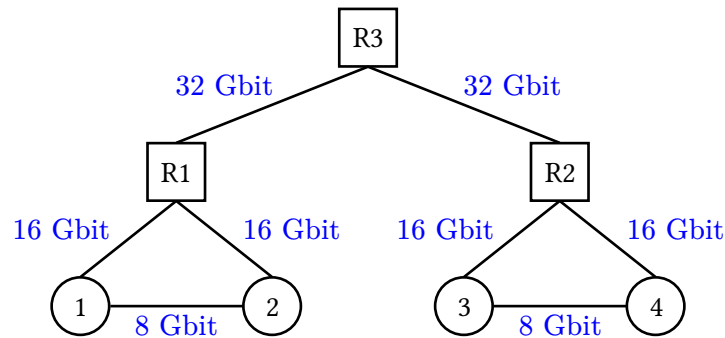
4.4 How many seconds are needed to execute AllReduce with the full-mesh overlay?

**6 seconds.** In a full-mesh, every node sends its vector to every other node. From the overlay, we can see that the longest-delay links take 6 seconds for packets to be sent.

4.5 To execute AllReduce with the full-mesh overlay, how much data is sent across each link (in either direction)?

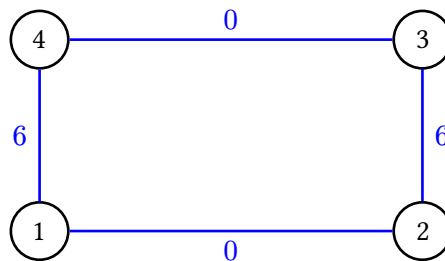
Label each link with the total amount of data sent across that link (in either direction).

Assume that all data is unicast.



4.6 Next, consider using a **ring** overlay topology to run the AllReduce operation. You can assume that we ignore ring optimization, so each node sends its entire vector to the next, instead of one element at a time.

What does the **overlay** topology look like? Draw virtual links in the diagram below. Then, label each virtual link with its packet travel time (according to the underlay network).



For subparts 4.7 and 4.8: Assume Node 4 is the first to send data, and the ring works clockwise. This means that Node 4 sends its data to Node 3, who calculates the sum of its vector with Node 4's vector, and passes it on, so on and so forth.

4.7 How many seconds are needed to execute AllReduce with the ring-based overlay? Assume that we start at Node 4, and work clockwise.

**18 seconds.** In the ring topology, data travels once around the ring to be summed up, and a second time around the ring to distribute the sum vector.

For instance, if we start at Node 4 and travel clockwise,

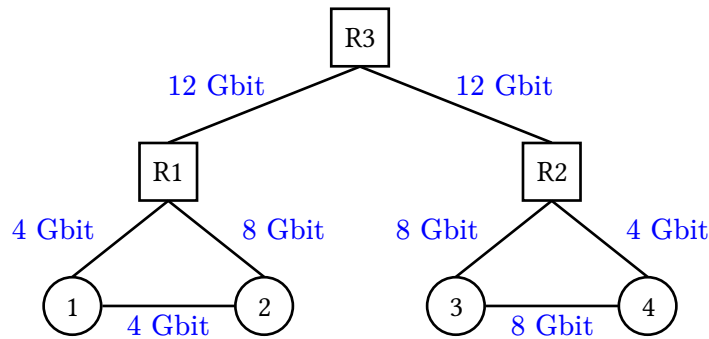
- Node 3 receives Node 4’s vector and sums Node 3 and Node 4’s vectors together, sending that summed vector to Node 2.
- Node 2 receives Node 3’s summed vector and calculates the sum between Node 2’s own vector and the summed vector from Node 3. Node 2 sends this vector to Node 1.
- Node 1 receives Node 2’s summed vector and calculates the sum between Node 1’s own vector and the summed vector from Node 2. Note: This vector contains the sum already! This means Node 1 already has the sum vector. Node 1 sends this vector now to Node 4.
- Node 4 receives the sum vector from Node 1, and passes it on to Node 3.
- Node 3 receives the sum vector from Node 4, and passes it on to Node 2.

Now, all the nodes have the sum vector and AllReduce is complete. This took a total of  $0 + 6 + 0 + 6 + 0 + 6 = 18$  seconds.

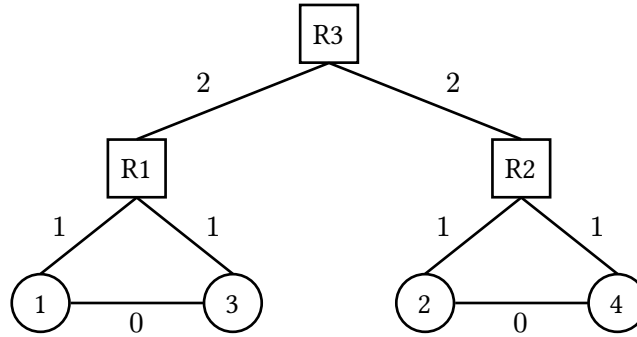
**4.8** To execute AllReduce with the ring-based overlay, how much data is sent across each link (in either direction)?

Label each link with the total amount of data sent across that link (in either direction).

Assume that all data is unicast.

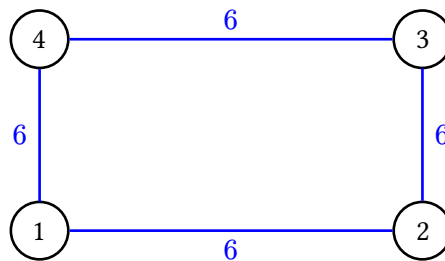


Next, suppose we relabel the nodes in the **underlay** topology. (Recall that the operator can decide what numbers to assign to each node, and the operator’s choice of numbering can affect the performance of the resulting collective operation.)



- 4.9 Consider using a **ring** overlay topology again to run AllReduce, but with the re-numbered nodes shown above. Again, you can assume that we ignore ring optimization, so each node sends its entire vector to the next, instead of one element at a time.

What does the **overlay** topology look like? Draw virtual links in the diagram below. Then, label each virtual link with its packet travel time (according to the underlay network).



For subparts 4.10 and 4.11: Assume Node 4 is the first to send data, and the ring works clockwise. This means that Node 4 sends its data to Node 3, who calculates the sum of its vector with Node 4's vector, and passes it on, so on and so forth.

- 4.10 How many seconds are needed to execute AllReduce with the ring-based overlay (and the re-numbered nodes)?

**36 seconds.** Similar to Question 4.7, it takes  $2(n - 1)$  total steps (where  $n$  is the number of nodes) to complete AllReduce in a ring-based topology. We have  $n = 4$  here, so 6 total steps leaves us with  $6 \cdot 6 = 36$  seconds.

- 4.11 To execute AllReduce with the ring-based overlay, how much data is sent across each link (in either direction)?

Label each link with the total amount of data sent across that link (in either direction).

Assume that all data is unicast.

