CSC175 Spring 2026 Final Exam Sample Questions ANS

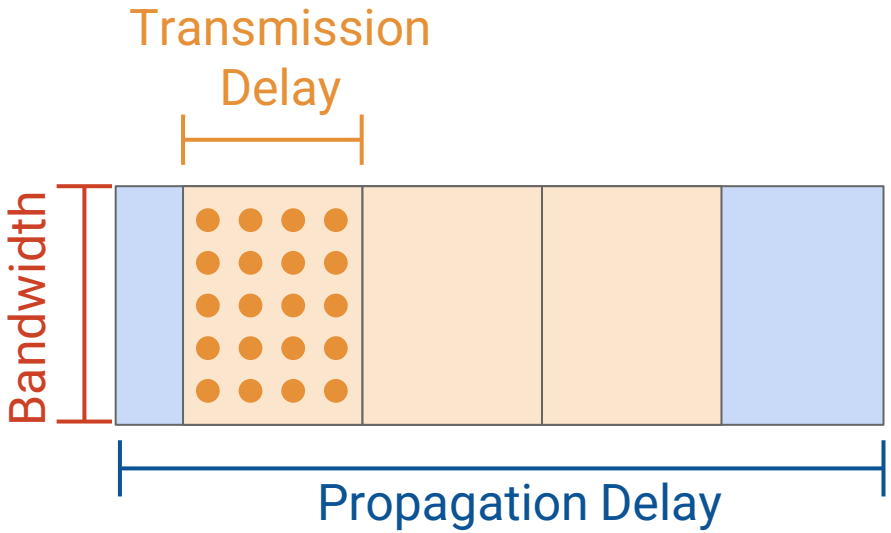
Z. Gu

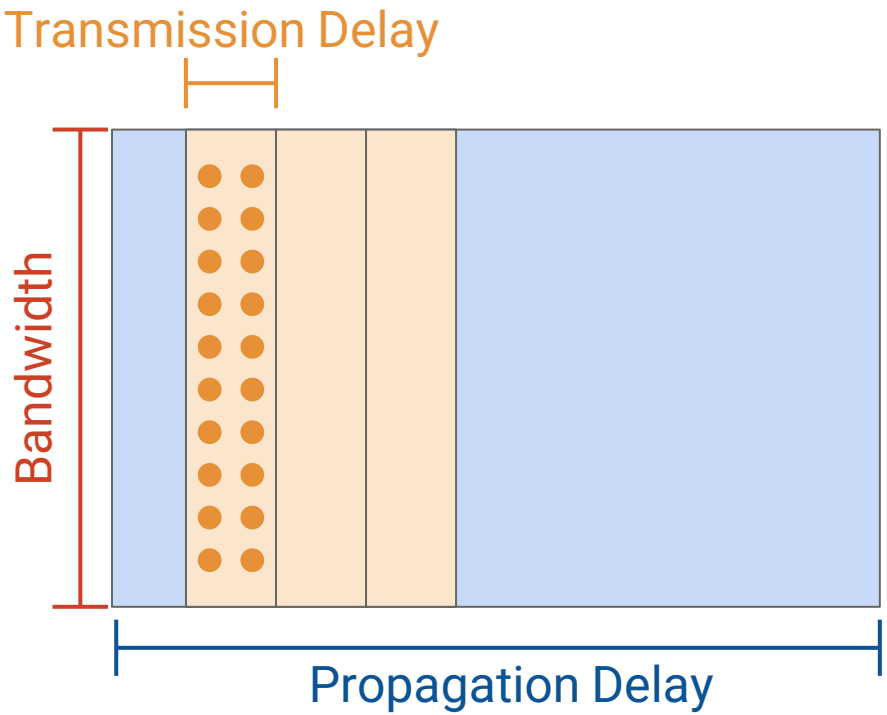
Spring 2026

IMPORTANT

The **width of the packet** in the pipe represents the transmission delay.

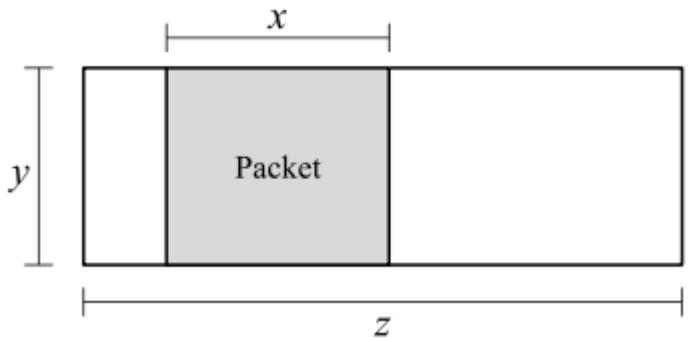
- How long it takes to push all the bits onto the link.
- **Link capacity = Propagation Delay × Bandwidth**
- **Packet size = Transmission Delay × Bandwidth**
  - For **a packet of given size**: higher bandwidth → shorter transmission delay (higher bandwidth = fatter pipe = more bits in pipe per unit time = narrower packet in pipe).





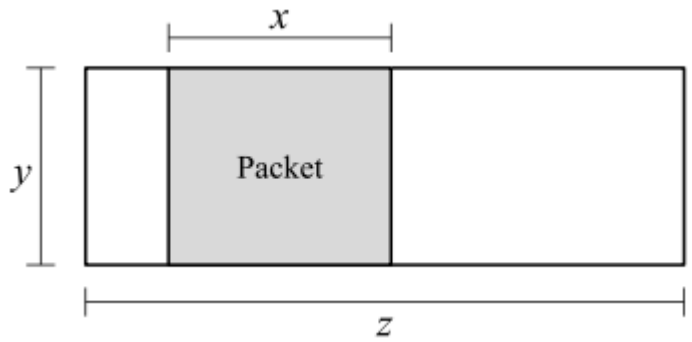
Lecture 3 - Links

- Consider the pipe diagram below, with a single packet in the pipe:
- Q1 What is the size of this packet?
- Q2 How long does it take to send a packet of the same size as the packet shown? (Count from the time the first byte is sent, to the time the last byte is received.)
- Q3 How long does it take to send 7 packets, all of the same size as the packet shown? (Count from the time the first byte of the first packet is sent, to the time the last byte of the last packet is received.)
- Q4 (2 points) What is the maximum number of packets that could be in the process of being sent along this link, at any given moment?



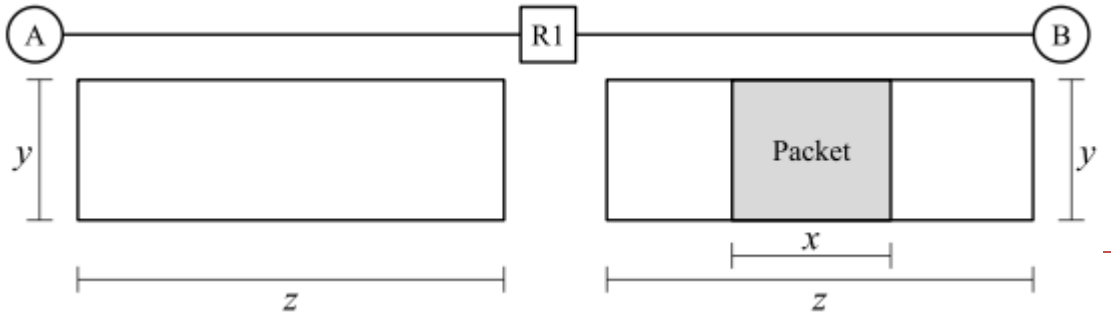
Lecture 3 - Links ANS

- Q1 What is the size of this packet?
  - ANS: xy
  - Explanation: x is Transmission Delay (how many seconds it takes to transmit the packet). y is link bandwidth (how many bytes can be sent per second). Multiplying x (seconds) and y (bytes/second) gives the overall size of this packet.
- Q2 How long does it take to send a packet of the same size as the packet shown? (Count from the time the first byte is sent, to the time the last byte is received.)
  - ANS: $x + z$
  - Explanation: x is Transmission Delay, z is Propagation Delay. Adding them gives the total delay for this packet.
- Q3 How long does it take to send 7 packets, all of the same size as the packet shown? (Count from the time the first byte of the first packet is sent, to the time the last byte of the last packet is received.)
  - ANS: $7x + z$
  - Explanation: It takes $7x$ seconds to transmit all 7 packets. z is the propagation delay. Adding them gives the total delay for 7 packets.
- Q4 (2 points) What is the maximum number of packets that could be in the process of being sent along this link, at any given moment?
  - ANS: z/x
  - Explanation:x is Transmission Delay, z is Propagation Delay. The maximum number of packets that can be in flight therefore is z/x.



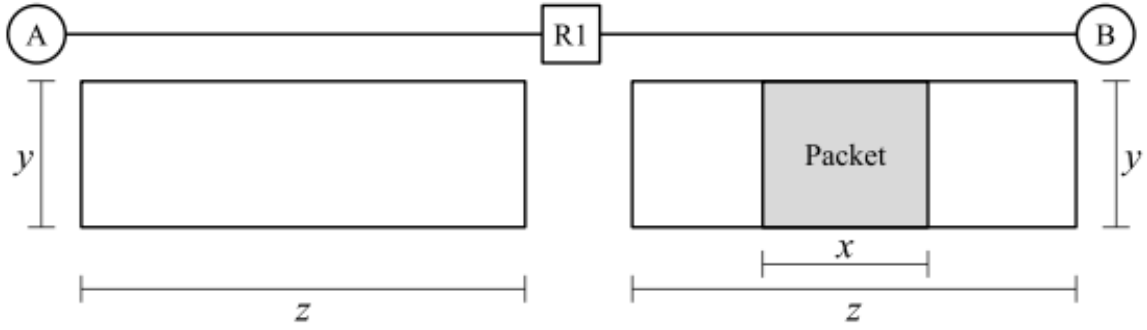
Lecture 3 - Links

- Now, consider the topology below. R1 has an infinite-size queue, and R1 continually processes packets in its queue in FIFO order. R1 can only start transmitting a packet once it has received the entire packet. All packets in the question are the same size as the packet shown in the diagram below. A wants to send a packet to B. No other packets are being sent along the links (i.e. suppose the packet in the diagram is not there). In the next two subparts, select how long it takes to send the packet from A to B. Count from the time the first byte of the packet is sent at A, to the time the last byte of the packet is received at B.
- Q5 In this subpart, suppose R1 has 1 other packet in its queue when A sends the packet. How long does it take to send a single packet from A to B?
- Q6 In this subpart, suppose R1 has 20 other packets in its queue when A sends the packet. How long does it take to send a single packet from A to B? Assume that the queue at R1 is not empty when this packet arrives at R1.
- Q7 What is the maximum number of packets that can be queued at R1, such that when the last byte of the A-to-B packet reaches R1, there is no queue at R1? Don't worry about fractions and rounding (e.g. assume x, y, and z are defined such that all answer choices are integers).



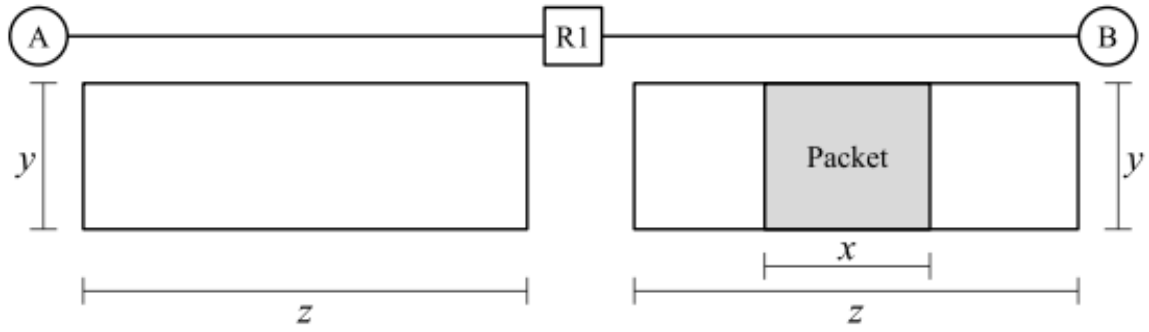
Lecture 3 - Links ANS

- Q5 In this subpart, suppose R1 has 1 other packet in its queue when A sends the packet. How long does it take to send a single packet from A to B?
  - ANS: $2x + 2z$
  - Explanation: Since all packets are the same size, by the time A transmits the packet onto the A-to-R1 link, R1 has finished clearing out its queue. Therefore, there will be no queuing delay at R1. The packet takes $x + z$ time to travel from A to R1 (transmission delay x and propagation delay z). At time $x + z$, R1 has received the entire packet. R1 then must send the packet along the R1-to-B link, which takes $x + z$ time again. The total time to transmit the packet from A to B is $2(x + z) = 2x + 2z$.
- Q6 In this subpart, suppose R1 has 20 other packets in its queue when A sends the packet. How long does it take to send a single packet from A to B? Assume that the queue at R1 is not empty when this packet arrives at R1.
  - ANS: $21x + z$
  - Explanation: By the time the A-to-B packet arrives at R1, the queue is not empty (per the assumption). Therefore, we have to account for queuing delay. We have to wait for R1 to finish sending all 20 packets, and then it can send the A-to-B packet. The time it takes for R1 to transmit 21 packets (the 20 in the queue, plus the A-to-B packet) is $21x$.
  - Then, the time it takes for the A-to-B packet to propagate along the R1-to-B link is z. Thus, the total packet delay is $21x + z$. Note that we don't need to consider the propagation delay on the A-to-R1 link, since it overlaps with the transmission delay $21x$ (i.e. the packet was sent from A to R1 while R1 was clearing out its queue).



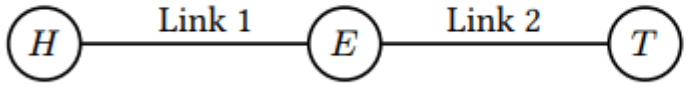
Lecture 3 - Links ANS

- Q7 What is the maximum number of packets that can be queued at R1, such that when the last byte of the A-to-B packet reaches R1, there is no queue at R1? Don't worry about fractions and rounding (e.g. assume x, y, and z are defined such that all answer choices are integers).
- ANS: $(x+z)/x$
- Explanation: The time it takes to send the packet from A to R1 is $x + z$. R1 needs x seconds to send out each packet. Therefore, in $x + z$ seconds, R1 is able to send out $(x+z)/x$ packets. Therefore, R1 can have at most $(x+z)/x$ packets in its queue, and all of those packets will be sent out by the time the last byte of the A-to-B packet reaches R1.



Lecture 3 - Links Queuing Delays ANS

- Starting at t = 0, H sends 10 payloads to T, one after the other. Each payload is 20 Gbits. E uses store and forward, i.e., it must receive all bits of a payload before sending that payload along the next link ().
- Q1 What is the queuing delay of the first payload? Note: For all queuing delay questions, count from the time the last bit of the payload arrives at E, to the time the first bit of the payload is sent out of E.
  - ANS: 0 sec
- Q2 (1 point) At what time does T receive the last bit of the first payload?
  - ANS: t = 15 sec
- Q3 (2 points) What is the queuing delay of the second payload?
  - ANS: 1 sec
- Q4 (2 points) At what time does T receive the last bit of the second payload?
  - ANS: t = 17 sec
- Q5 (2 points) What is the queuing delay of the 10th payload?
  - ANS: 9 sec
- Q6 (2 points) At what time does T receive the last bit of the 10th payload?
  - ANS: t = 33 sec



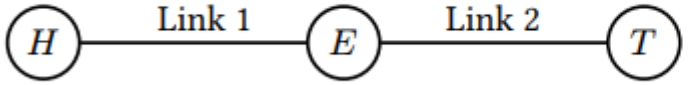
	Link 1	Link 2
Propagation Delay	2 sec	10 sec
Bandwidth	20 Gbps	10 Gbps

Lecture 3 - Links Queuing Delays ANS

For payload number k:
queueing *delay at* $E = (2k+1) - (k+2) = k-1$
So the delays are:
$0,1,2,3, \ldots, 9$, for payloads 1 through 10.
Packets arrive at E every 1 second, but E can only start sending one every 2 seconds, so the backlog grows by 1 second per packet.

- Q1-5 ANS. This table shows the time each payload arrives at E (time of last bit arriving), the time each payload is sent from E (time of first bit sent out), and the queuing delay of payload.
- Link 1: H -> E. Bandwidth = 20 Gbps, Transmission Delay for each 20 Gbit payload is 1s. Propagation Delay = 2s
  Link 2: H -> ET. Bandwidth = 10 Gbps, Transmission Delay for each 20 Gbit payload is 2s (service time). Propagation Delay = 10s
- To derive the time of arrival of the last bit of each payload at E column, we consider Link 1:
  - The first payload of 20 Gbps: First bit is transmitted from H at t = 0; the last bit is transmitted from H at t = 1, and arrives at E at t = 1 + Prop Delay (2) = 3.
  - At this point, a constant stream of payloads is sent from H to E back-to-back. Each payload takes 1 second to transmit, so the times of arrivals are 3, 4, 5, … (1 second per packet).
- To derive the time of sending of the first bit of each payload from E column, we consider Link 2:
  - Payload 1's last bit arrives at t = 3, so its first bit can be forwarded from E starting at t = 3, , with queuing delay of 3-3 = 0. The last bit is transmitted from E at t = 5, and arrives at T at t = 5 + Prop Delay (10) = 15.
  - Payload 2's last bit arrives at t = 4, so its first bit can be forwarded from E starting at t = 5, with queuing delay of 5-4 = 1. The last bit is transmitted from E at t = 7, and arrives at T at t = 7+10 =17.
  - Time of sending of the first bit of each payload from E: 3, 5, 7,…
- The queuing delay is the difference between the time of last bit arriving, and the time of first bit sending. It increases with each payload, since the queue builds up in size gradually, as Link 2 has a lower bandwidth than Link 1. Every second, backlog increases steadily → queue builds → delay grows
  - 1 payload arrives (20 Gbps link)
  - But only 0.5 payload leaves (10 Gbps link)

Payload number	Time of arrival at E	Time of sending from E	Queuing delay of payload
1	3	3	0
2	4	5	1
3	5	7	2
4	6	9	3
5	7	11	4
6	8	13	5
7	9	15	6
8	10	17	7
9	11	19	8
10	12	21	9

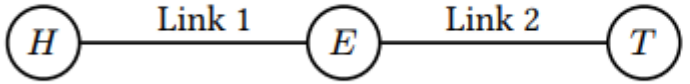


	Link 1	Link 2
Propagation Delay	2 sec	10 sec
Bandwidth	20 Gbps	10 Gbps

Lecture 3 - Links Queuing Delays ANS

- Q6 ANS: When the first bit of a payload is sent out from E, the last bit gets sent out 2 seconds later. This is because Link 2 has bandwidth 10 Gbps, and the payload is 20 Gbits.
- The last bit then takes another 10 seconds to arrive at T .This is the propagation delay of Link 2.
- Thus, the time the last bit of a payload arrives at T is 12 seconds after the time the first bit is sent out of E.

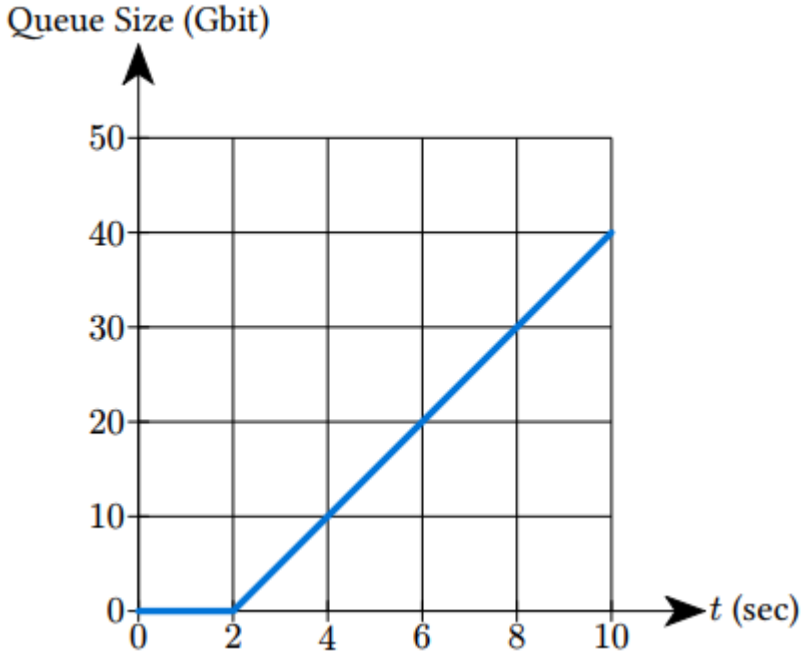
Payload number	Time of arrival at E	Time of sending from E	Time of arrival at T
1	3	3	15
2	4	5	17
3	5	7	19
4	6	9	21
5	7	11	23
6	8	13	25
7	9	15	27
8	10	17	29
9	11	19	31
10	12	21	33

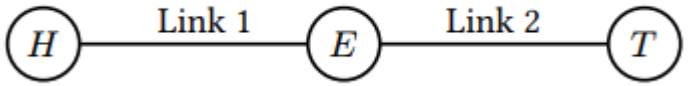


	Link 1	**Link 2**
Propagation Delay	2 sec	10 sec
Bandwidth	20 Gbps	10 Gbps

Lecture 3 - Links Queuing Delays ANS

- New Setup: Starting at $t = 0$, H sends an endless stream of data, at constant rate 20 Gbps to T. At E, each bit can be forwarded independently of other bits.
- Q7 Link 2 has a constant bandwidth of 15 Gbps. How does the queue size grow with time?
- ANS: The first bit is sent at $t = 0$, and arrives at E at $t = 2$, after 2-second propagation delay of Link 1. This means that from $t = 0$ to $t = 2$, no data arrives at E, so there is no queue.
- Starting at $t = 2$, a steady stream of 20 Gbps is arriving at E, and a steady stream of 15 Gbps is leaving E. This means that every second, 20 Gbits enter and 15 Gbits leave. This leaves $20 - 15 = 5$ Gbits, which get added to the queue in every second.

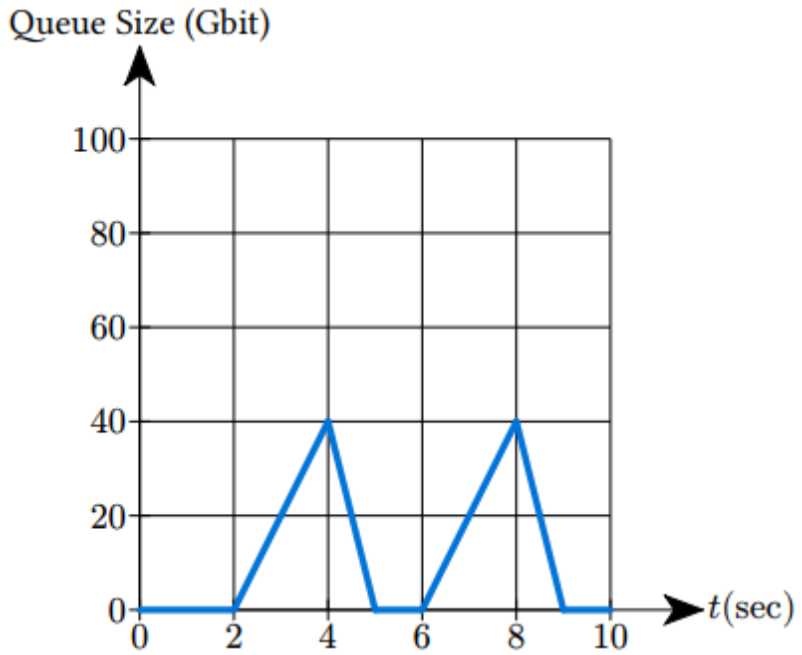


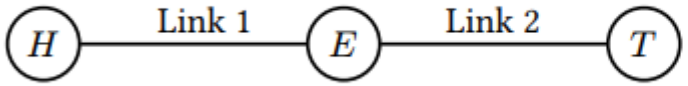


	Link 1	Link 2
Propagation Delay	2 sec	10 sec
Bandwidth	20 Gbps	*see subparts*

Lecture 3 - Links Queuing Delays ANS

- New Setup: Starting at t = 0, H sends an endless stream of data, at constant rate 20 Gbps to T. At E, each bit can be forwarded independently of other bits.
- Q8 Bandwidth of Link 2 alternates forever between 60 and 0 Gbps, changing every 2 seconds: From t = 0 to t = 2, the bandwidth is 60 Gbps. From t = 2 to t = 4, the bandwidth is 0 Gbps, and so on.
- How does the queue size grow with time?
- ANS: queue change rate (slope) = Link 1 bandwidth (arrival rate) – Link 2 bandwidth (departure rate)
  - Holds true when queue is not empty.
  - When queue is empty, queue slope = max(0, Link 1 bandwidth – Link 2 bandwidth) since queue size cannot be negative.
- From t = 0 to t = 2, no data arrives at E, so there is no queue.
- From t = 2 to t = 4, a steady stream of 20 Gbps is arriving at E, and 0 Gbps is leaving E. queue change rate = 20 – 0 = 20. This means that every second, 20 Gbits get added to the queue.
- From t = 4 to t = 6, a steady stream of 20 Gbps is arriving at E, and 60 Gbps is leaving E. queue change rate = 20 – 60 = -40. This means that every second, 40 Gbits get removed from the queue.

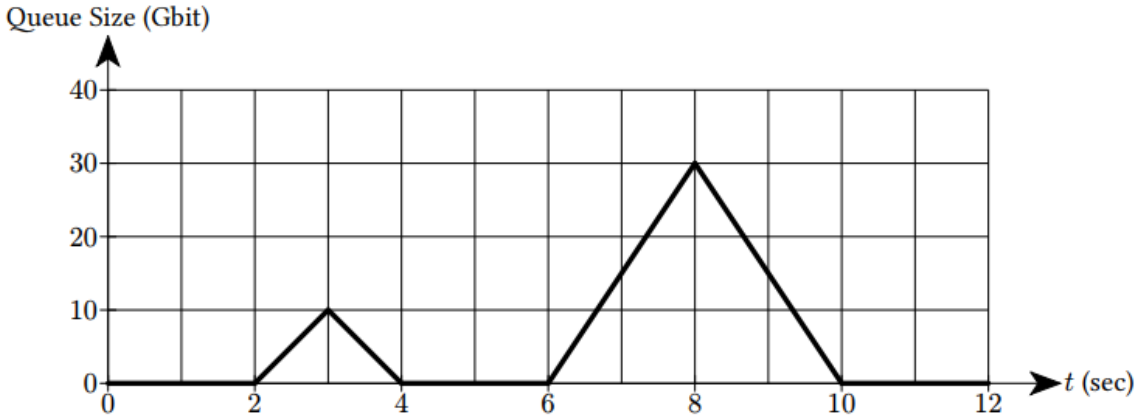




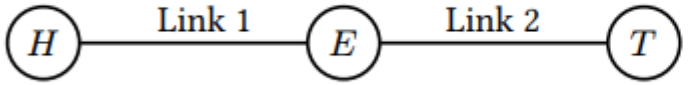
	Link 1	Link 2
Propagation Delay	2 sec	10 sec
Bandwidth	20 Gbps	*see subparts*

Lecture 3 - Links Queuing Delays ANS

- New Setup: Starting at $t = 0$, H sends an endless stream of data, at constant rate 20 Gbps to T. At E, each bit can be forwarded independently of other bits.
- Q9 Bandwidth of Link 2 now changes in an unknown pattern. Use the graph to determine the bandwidth of Link 2 at the following times.
- ANS: queue change rate (slope) = Link 1 bandwidth (arrival rate) – Link 2 bandwidth (departure rate)
  - Holds true when queue is not empty.
  - When queue is empty, queue slope = max(0, Link 1 bandwidth – Link 2 bandwidth) since queue size cannot be negative.
- At t=5, the queue has change rate of 0 Gbps, Link 2 bandwidth = Link 1 bandwidth - queue change rate = 20 – 0 = 20 Gbps. But the queue is empty, so any departure rate larger than or equal to 20 Gbps is OK.
- At t=7, the queue is increasing with a rate of 30 Gbit/2s, with queue slope = 15 Gbps, Link 2 bandwidth = Link 1 bandwidth - queue change rate = 20 – 15 = 5 Gbps.
- At t=9, the queue is draining at a rate of 15 Gbps, so Link 2 bandwidth = Link 1 bandwidth - queue change rate = 20 – (-15) = 35 Gbps.



Time	Bandwidth of Link 2
5 sec	$[20, \infty)$ Gbps
7 sec	$[5, 5]$ Gbps
9 sec	$[35, 35]$ Gbps



	Link 1	Link 2
Propagation Delay	2 sec	10 sec
Bandwidth	20 Gbps	*see subparts*

Store-and-Forward vs. Bit-by-Bit Forwarding (Cut-through)

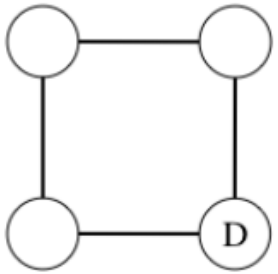
- **Q1–5: Store-and-Forward**
- "E must receive all bits before sending"
- E waits for the **entire payload**
- Then starts transmitting
- Packets move **as chunks**
- Consequence:
  - You get **packet-level queueing**
  - Clear "arrival time" (last bit)
  - Clear "send start time"

- **Q7–9: Bit-by-Bit Forwarding (Cut-through)**
- "each bit can be forwarded independently"
- E can start sending **as soon as bits arrive**
- No need to wait for full packet
- Data flows like a **continuous stream**
- Consequence:
  - No packet boundaries
  - Queue is measured in **bits (fluid model)**
  - Use **rates (Gbps)** instead of per-packet timing

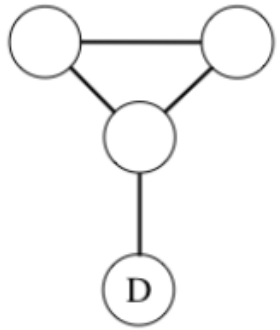
Lecture 5.2 - Distance-Vector ANS

- Distance-Vector: Distance-vector routing is plagued by the “count-to-infinity" problem. Poisoned reverse is one approach to mitigating this problem. For the following graphs, consider the case where distance-vector routing has converged to a stable set of routes, and then the destination (marked D) fails, taking down all of the links attached to it. Which of these graphs still have the count-to-infinity problem even when poisoned reverse is used?

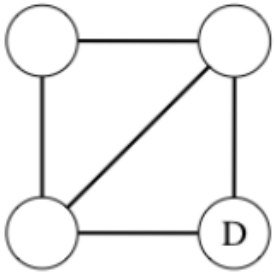
A.



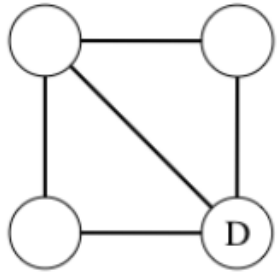
B.



C.

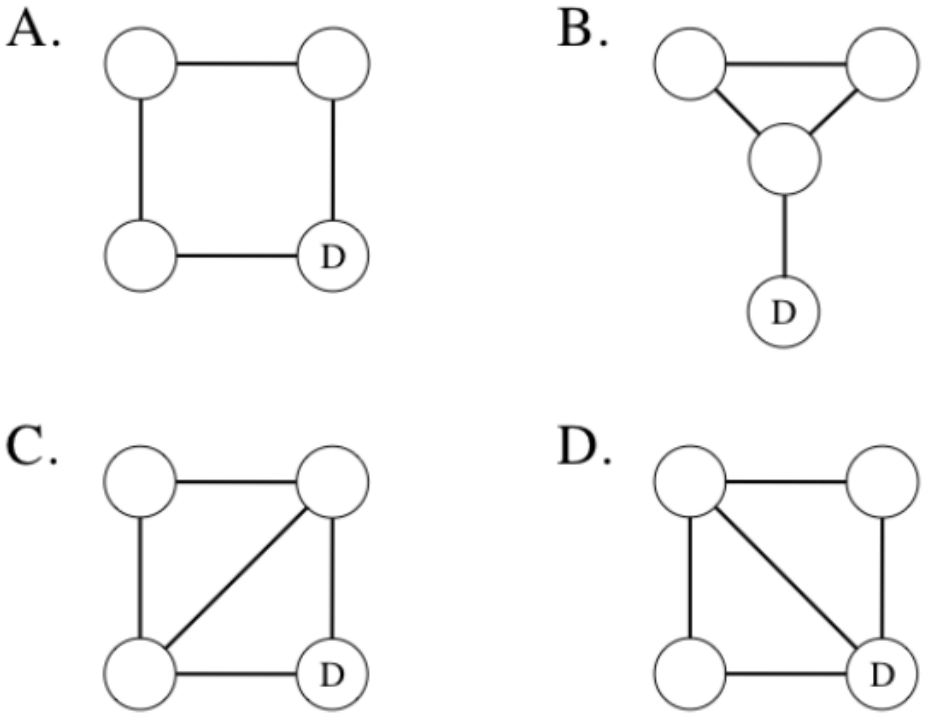


D.



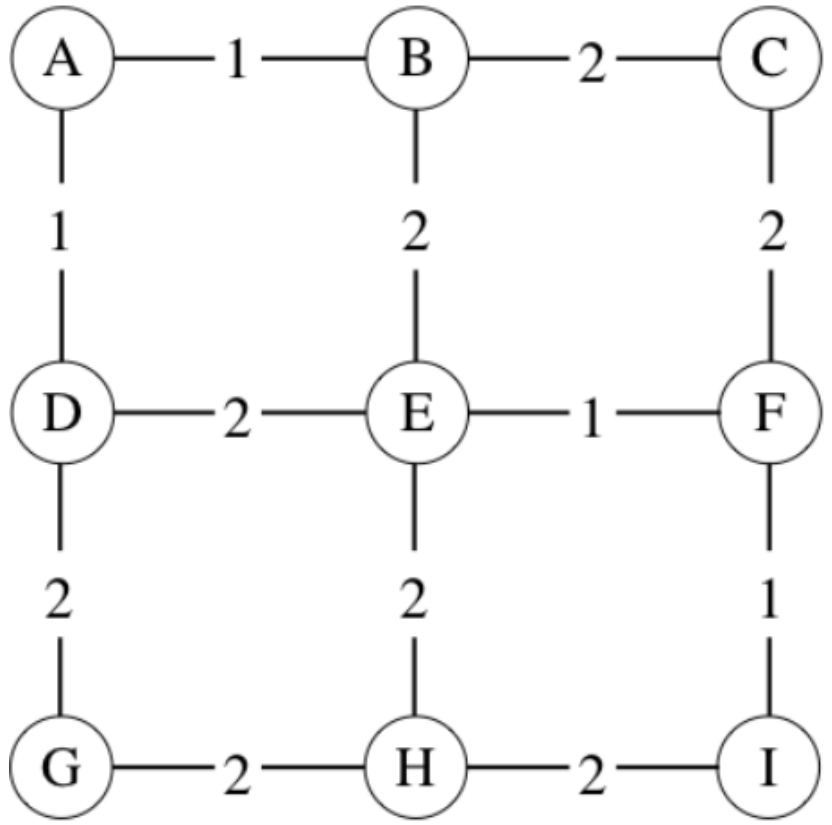
Lecture 5.2 - Distance-Vector ANS

- Destination (marked D) fails, taking down all of the links attached to it. Which of these graphs still have the count-to-infinity problem even when poisoned reverse is used?
- ANS:
- A & D: No. Once D is removed, the remaining three routers form a path, not a triangle. That means any mistaken route can only bounce between two neighbors, and poisoned reverse is designed to stop exactly that kind of 2-node loop. So A & D does not have count-to-infinity with poisoned reverse.
- B & C: Yes. When D fails, the remaining three routers form a triangle. A router may poison one neighbor but still advertise a finite path to another, allowing the bad route to travel around the 3-node cycle. Because poisoned reverse only hides the route from the neighbor being used as next hop, the false route can circulate around the triangle and the metric can increase repeatedly



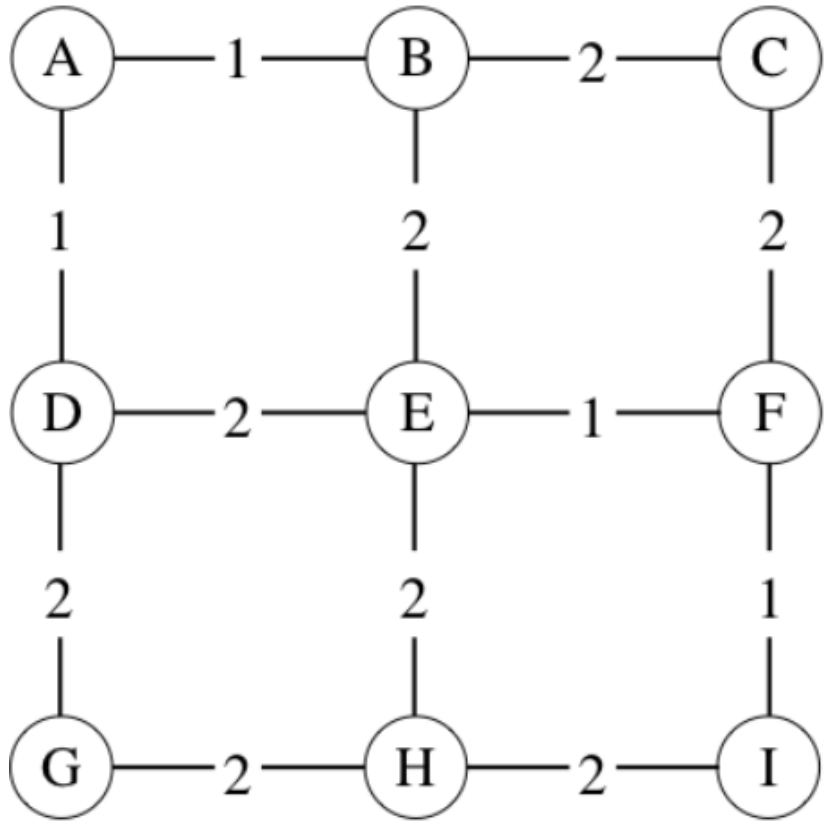
Lecture 5.3 - Link-State

- Link-State: For the network shown below, consider the case where all nodes have flooded their link-state information to everyone, shortest paths have been calculated, and the appropriate forwarding entries installed. But then the link between E and F goes down. The packets announcing this link-status change reach all nodes except B, and again routes are recalculated. When D sends a packet to F, what path does it take? Ties are broken in alphabetical order.



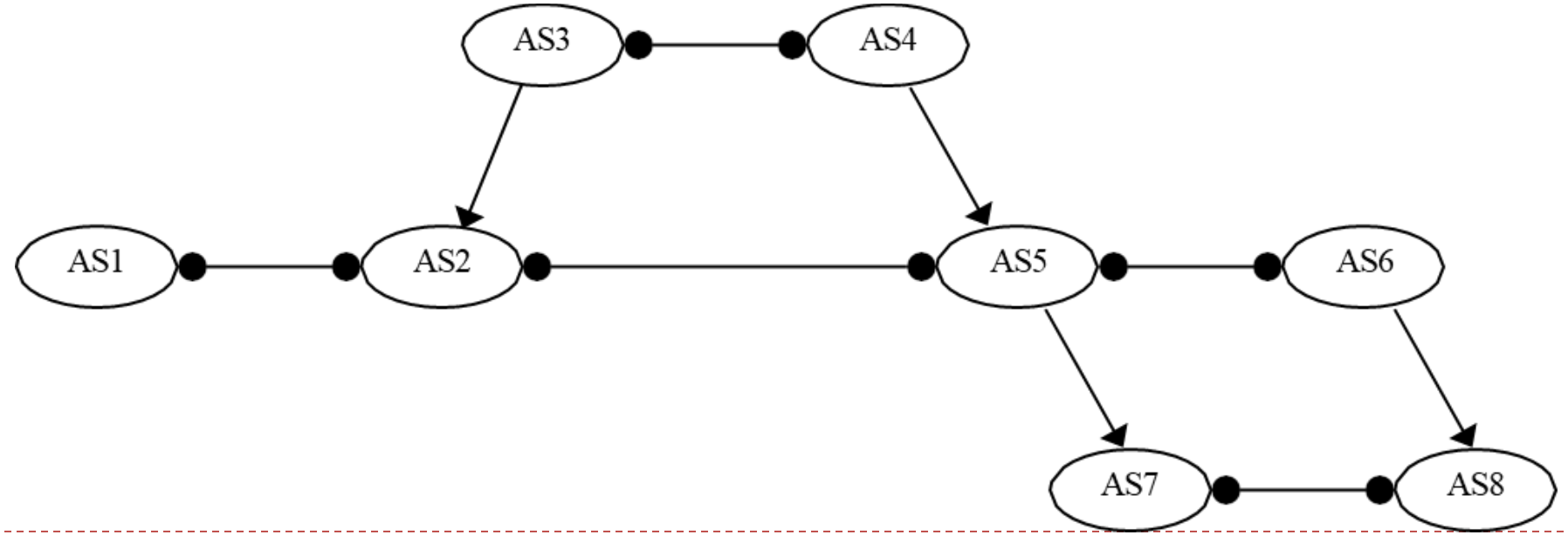
Lecture 5.3 - Link-State ANS

- The link between E and F goes down. The packets announcing this link-status change reach all nodes except B, and again routes are recalculated. When D sends a packet to F, what path does it take? Ties are broken in alphabetical order.
- ANS: DABEHIF
- Explanations:
  - After the E–F link goes down, D recomputes using the new topology, where the best path from D to F is $D \rightarrow A \rightarrow B \rightarrow C \rightarrow F$ with total cost 6.
  - A also recomputes correctly, so when it receives the packet from D, it forwards it to B. B did **not** receive the link-state update, so it still believes the old shortest path to F is $B \rightarrow E \rightarrow F$, with cost 3, and therefore sends the packet to E.
  - E did receive the failure update, so E knows the direct link to F is down and cannot forward there. From E's updated view, the best remaining route to F is $E \rightarrow H \rightarrow I \rightarrow F$, with cost 5, which is better than going back through B and C.
  - Putting those forwarding decisions together gives the actual packet path: **DABEHIF**.



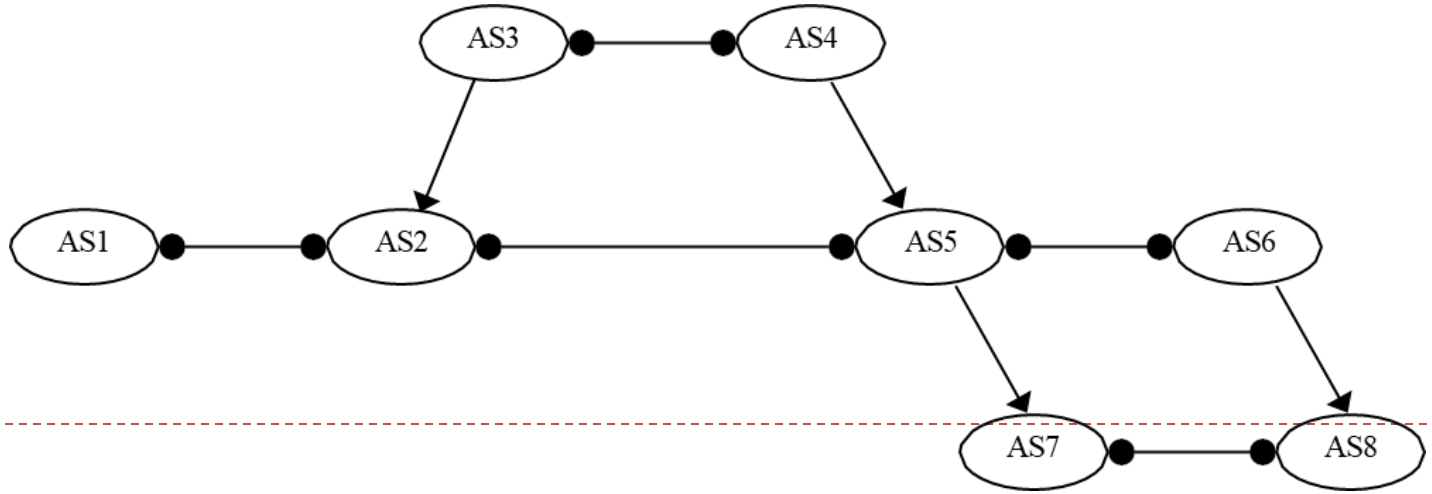
Lecture 8 - Inter-Domain Routing

- Consider the AS graph below, where each AS follows the Gao-Rexford import and export policies. Provider to Customer relationship is denoted by arrows; peer to peer relationship is denoted by horizontal lines with dots. (Hint: Every intermediate AS on a legal path must have at least one customer neighbor along that path. Valid AS paths should be valley-free: traffic goes up provider links zero or more times, may cross at most one peer link, and then goes down customer links zero or more times.) Determine the path (if any) between the following ASes. If there is no path possible, write "None".
- a) AS1 to AS8
- b) AS2 to AS7
- c) AS5 to AS8
- d) AS2 to AS5



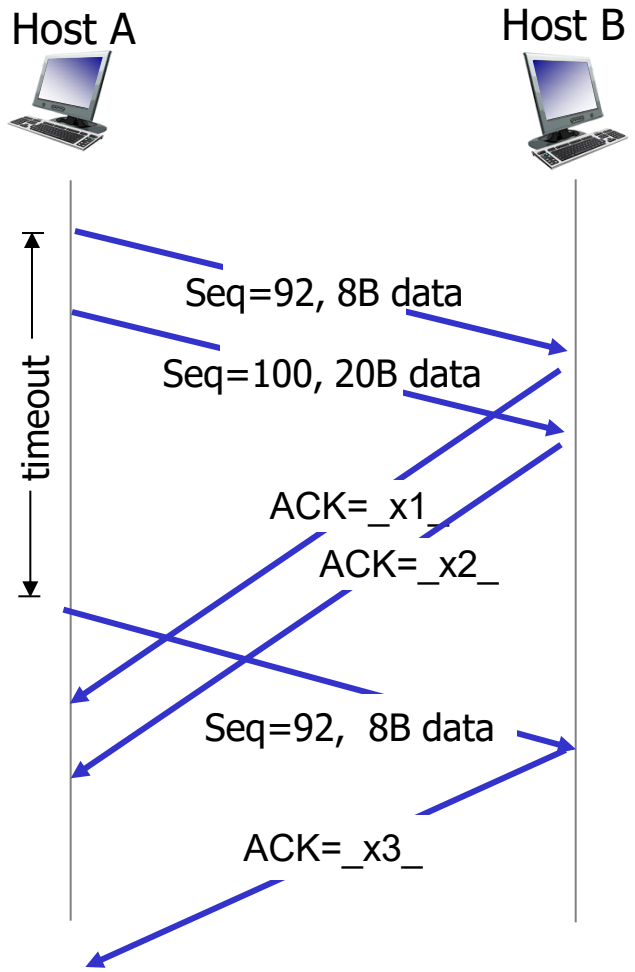
Lecture 8 - Inter-Domain Routing ANS

- a) AS1 to AS8
  - None
- b) AS2 to AS7
  - AS2, AS5, AS7
  - Note: AS2, AS3, AS4, AS5, AS7 is also legal, but AS2, AS5, AS7 is the correct path. Since AS2 learns that route from a peer (AS5), it is preferred over the provider-learned route via AS3
- c) AS5 to AS8
  - AS5, AS6, AS8
  - Note: AS5, AS7, AS8 is not a legal path, since AS7 does not have a customer neighbor along that path. It also violates the valley-free rule.
- d) AS2 to AS5
  - AS2, AS5
  - Note: AS2, AS3, AS4, AS5 is also legal, but AS2, AS5 is the correct path.



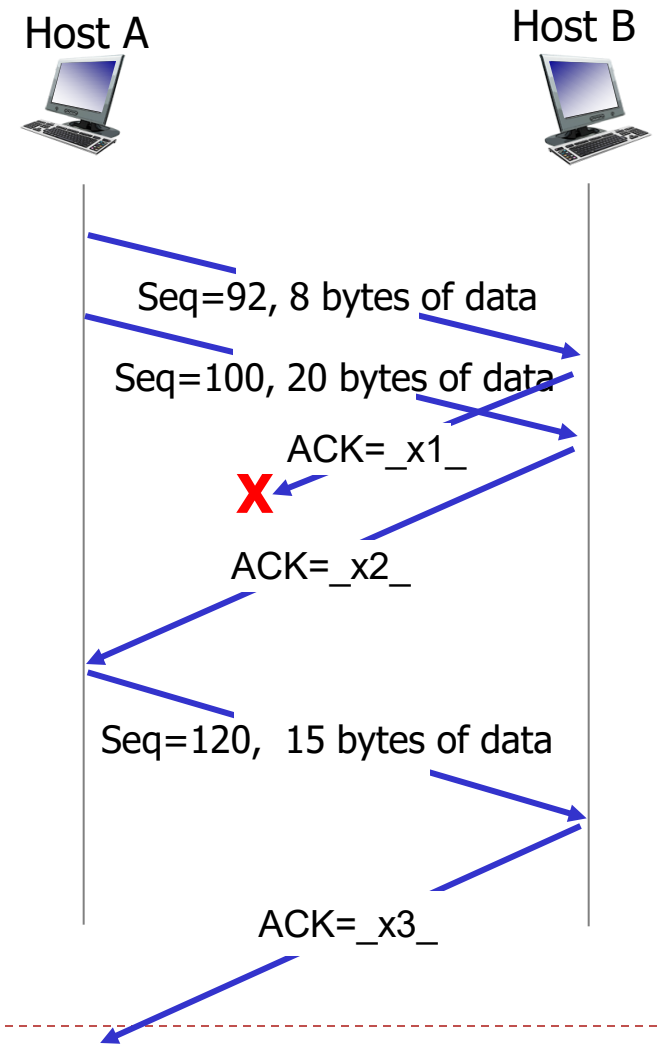
Lecture 11 TCP Implementation ANS

- Consider the TCP sequence diagram with sequence numbers in Bytes. What are the ACKed sequence numbers x1, x2, and x3?
- ANS: x1 = 100, x2 = 120, x3 = 120
- (Refer to Lecture Slide 13.)



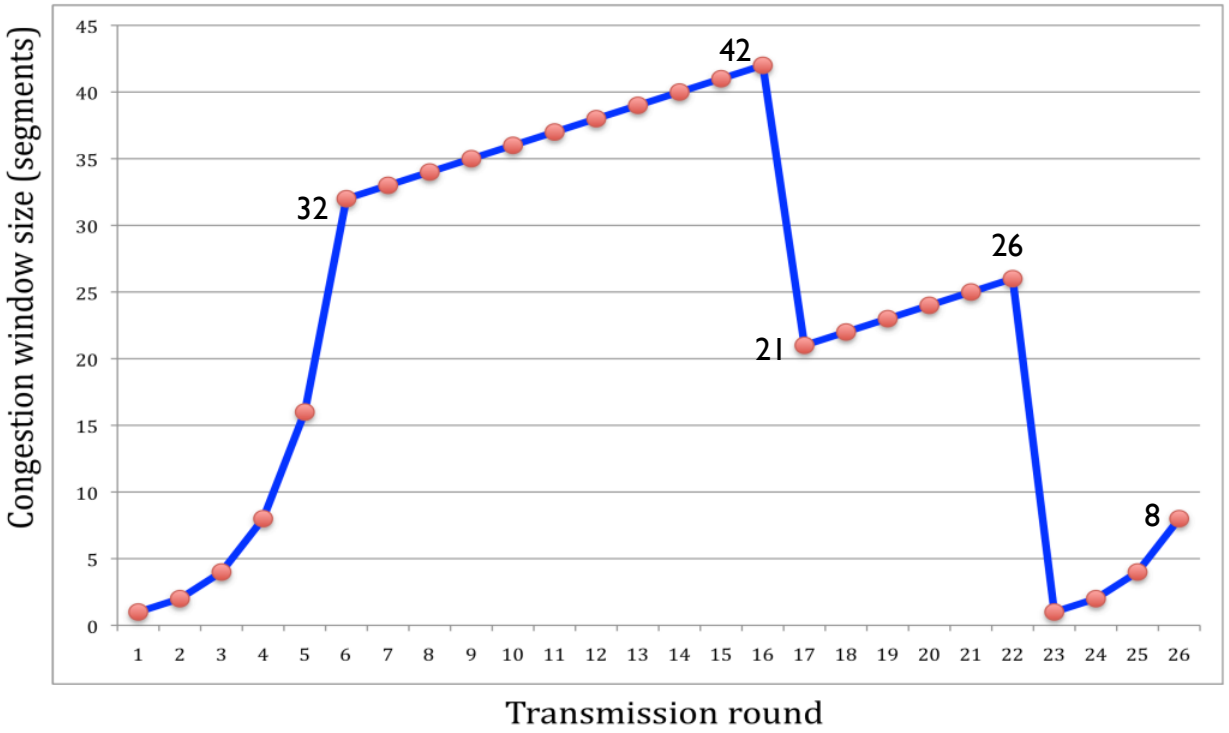
Lecture 11 TCP Implementation ANS

- Consider the TCP sequence diagram with sequence numbers in Bytes. What are the ACKed sequence numbers x1, x2, and x3?
- ANS: x1 = 100, x2 = 120, x3 = 135
- (Refer to Lecture Slide 15.)



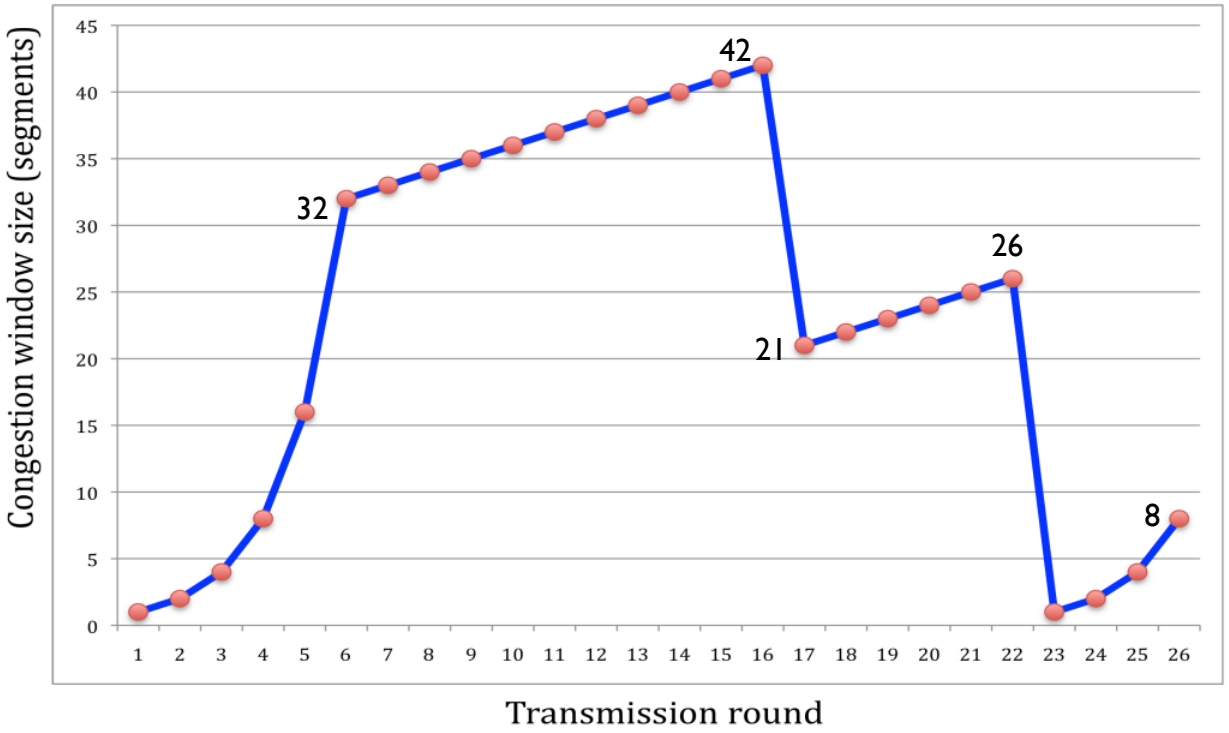
Lecture 12-13 - Congestion Control

- Consider the figure that plots the evolution of TCP's congestion window at the beginning of each time unit (where the unit of time is equal to the RTT). TCP sends a "flight" of packets of size cwnd at the beginning of each time unit. Thereafter, either (i) all packets are ACKed at the end of the time unit, (ii) there is a timeout for the first packet, or (iii) there is a triple duplicate ACK for the first packet. The initial value of cwnd is 1. Assume TCP Reno protocol (CWND=1 on timeout. CWND = CWND/2 after triple duplicate acks. If CWND is an odd number, then CWND = [CWND/2].) **Assume no fast recovery.**) Assume no gap between detecting a packet loss and sending out the next packet.
- a) Identify the interval of time when TCP slow start is operating.
- b) Identify the interval of time when TCP congestion avoidance is operation.
- c) After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
- d) After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?
- e) What is the initial value of ssthresh at the first transmission round?
- f) What is the value of ssthresh at the 18th transmission round?
- g) What is the value of ssthresh at the 24th transmission round?
- h) During what transmission round is the 70th segment sent?
- i) Assuming a packet loss is detected after the 26th round by the receipt of a triple duplicate ACK, what will be the value of the congestion window size and of ssthresh?



Lecture 12-13 - Congestion Control ANS

- a) TCP slow start is operating in the intervals [1,6] and [23,26]
- b) TCP congestion avoidance is operating in the intervals [6,16] and [17,22]
- c) After the 16th transmission round, packet loss is recognized by a triple duplicate ACK.
- d) After the 22nd transmission round, segment loss is detected due to timeout, and hence the congestion window size is set to 1.
- e) The threshold is initially 32, since it is at this window size that slow start stops and congestion avoidance begins.
- f) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 16, the congestion window size is 42. Hence the threshold is 21 during the 18th transmission round.
- g) The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during transmission round 22, the congestion window size is 26. Hence the threshold is 13 during the next round.
- h) During the 1st transmission round, packet 1 is sent; packets 2–3 are sent in the 2nd transmission round; packets 4–7 are sent in the 3rd transmission round; packets 8–15 are sent in the 4th transmission round; packets 16–31 are sent in the 5th transmission round; packets 32–63 are sent in the 6th transmission round; packets 64–96 are sent in the 7th transmission round. Thus packet 70 is sent in the 7th transmission round.
- i) The threshold will be set to half the current value of the congestion window (8) when the loss occurred and congestion window will be set to the new threshold value. Thus the new values of the threshold and window will be both 4.



Lecture 15 - Block Cipher

Consider the 3-bit block cipher in the Table below

Plain	000	001	010	011	100	101	110	111
Cipher	111	110	101	100	011	010	000	001

Suppose the plaintext is 100101100.

(a) Initially assume that CBC is not used. What is the resulting ciphertext?

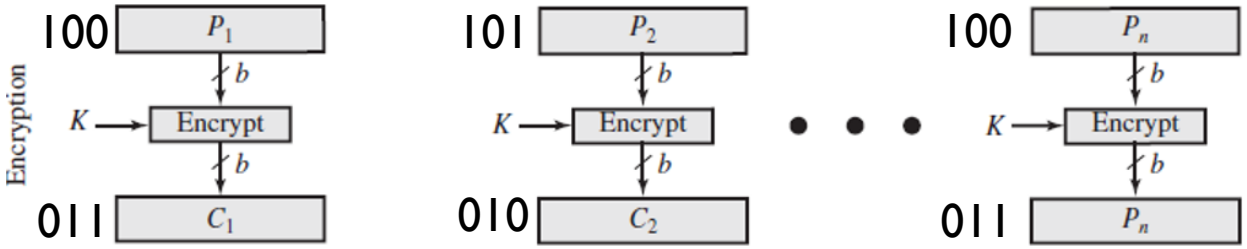
(b) Suppose Trudy sniffs the ciphertext. Assuming she knows that a 3-bit block cipher without CBC is being employed (but doesn't know the specific cipher), what can she infer?

(c) Now, suppose that CBC is used with IV-111. What is the resulting ciphertext?

Lecture 15 - Block Cipher ANS

- (a) Initially assume that CBC is not used. What is the resulting ciphertext?
  - ANS: Ciphertext for plaintext 100101100 is 011010011, since 100 maps to 011, 101 maps to 010, 100 maps to 011
- (b) Suppose Trudy sniffs the ciphertext. Assuming she knows that a 3-bit block cipher without CBC is being employed (but doesn't know the specific cipher), what can she infer?
  - ANS: Since the same plaintext block always maps to the same ciphertext block, repeated blocks can be spotted. Here, the first and third plaintext blocks are both 100, and they both encrypt to 011. That reveals a pattern.
- (c) Now, suppose that CBC is used with IV=111. What is the resulting ciphertext?
  - ANS: With CBC and IV = 111, resulting ciphertext for plaintext 100101100 is 100110101. (See next page.)

Plain	000	001	010	011	100	101	110	111
Cipher	111	110	101	100	011	010	000	001

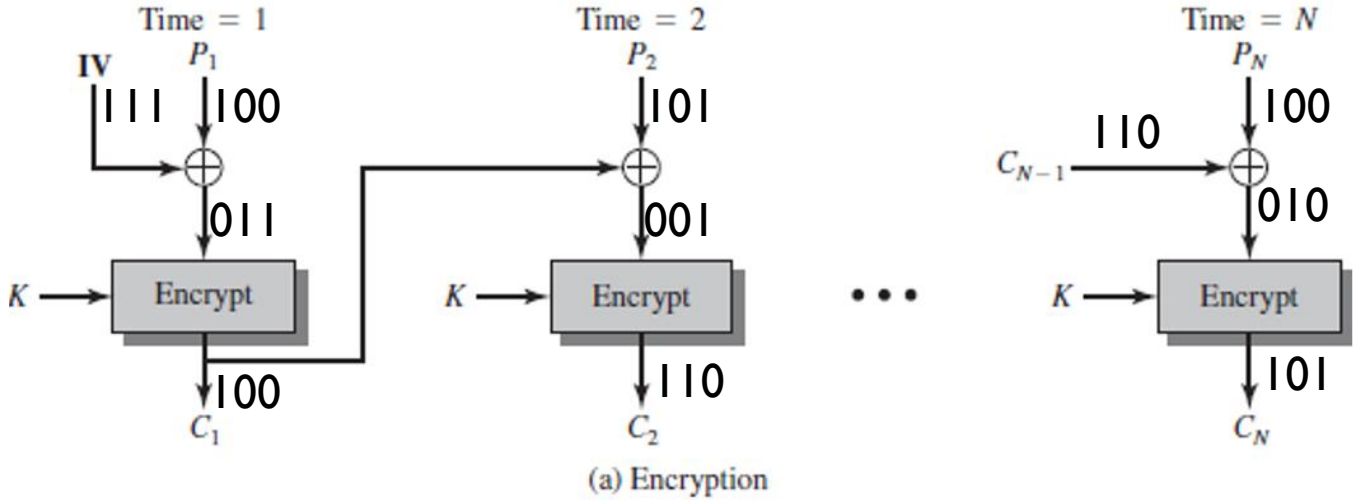


The same plaintext 100 is encrypted into the same cyphertext (011) at different positions in the input, making it possible to attacker to perform frequency analysis.

Lecture 15 - Block Cipher ANS

- Plaintext 100101100
- The first step is to XOR the first plaintext block with IV = 111
  - First plaintext block: 100, so 100⊕111=011
  - Now we encrypt this result (011) using our cipher table: 011 maps to 100.
- Second Block: Now we XOR the second plaintext block with the first ciphertext block:
  - Second plaintext block: 101, so 101⊕100=001
  - Now we encrypt this result (001) using our cipher table: 001 maps to 110.
- Third Block: Finally, we XOR the third plaintext block with the second ciphertext block:
  - Third plaintext block: 100, so 100⊕110=010
  - Now we encrypt this result (010) using our cipher table: 010 maps to 101.
- Resulting ciphertext for plaintext 100101100 is 100110101.

Plain	000	001	010	011	100	101	110	111
Cipher	111	110	101	100	011	010	000	001



The same plaintext 100 is encrypted into different cyphertexts (100 or 101) at different positions in the input, thanks to CBC.

Lecture 15 - Diffie-Hellman

- Suppose Alice and Bob wish to do Diffie-Hellman key exchange. Alice and Bob have agreed upon a prime p = 13, and a generator g = 2. Alice has chosen her secret number (private exponent) to be a = 5, while Bob has chosen his private exponent to be b = 4.
- Show the intermediate quantities that both Alice and Bob calculate, as well as the final (shared) secret that Diffie-Hellman produces.

Lecture 15 - Diffie-Hellman ANS

- Suppose Alice and Bob wish to do Diffie-Hellman key exchange. Alice and Bob have agreed upon a prime p = 13, and a generator g = 2. Alice has chosen her secret number (private exponent) to be a = 5, while Bob has chosen his private exponent to be b = 4.
- Show the intermediate quantities that both Alice and Bob calculate, as well as the final (shared) secret that Diffie-Hellman produces.
- ANS: Alice sends to Bob: A=g^a mod p= 2^5 mod 13 = 6.
- Bob computes the secret: S=A^b mod p = 6^4 mod 13 = 1296 mod 13 = 9.
- Bob sends to Alice B=g^b mod p = 2^4 mod 13 = 3.
- Alice computes the secret: S=B^a mod p = 3^5 mod 13 = 243 mod 13 = 9.