CS144
Intro to Computer Networks
Final Exam – Tuesday, March 19, 2019

2 NOTE PAGES, CLOSED BOOK, COMPUTERS OFF

Your Name: Answers

SUNet ID: root @stanford.edu

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

Signature: _____

- The exam has 12 questions totaling 99 points.
- You have 120 minutes to complete them.
- Some questions may be much harder than others.
- You must circle at least one option in order to receive credit for a multiple-choice question.
- No calculators allowed.
- You may express numerical answers as expressions if you don't do the arithmetic, but please explain each term when you do.
- Please box your final answers.

1	/3
2	/3
3	/3
4	/3
5	/2
6	/7
7	/15
8	/10
9	/10
10	/10
11	/18
12	/15
Total	/99

I Industrial Congestion Control

1. **[3 points]:**

   Nandita Dukkipati from Google discussed projects like *TIMELY: RTT-based Congestion Control for Datacenter*, *BBR: Congestion-based Congestion Control*, and *RCP: Rate Control Protocol*. What were some of the motivations for developing these projects? What were some of the reasons that a traditional loss-based congestion control algorithm (e.g., TCP Reno) were problematic? (Select all that apply.)

   A Loss-based algorithms cause increased latency by filling network buffers and fails to utilize bandwidth in the presence of moderate loss.

   B Loss is a single bit of information—a packet was lost or it wasn't. In order to get a multibit signal that reflects queueing in the path, we can measure RTTs.

   C Because losses are extremely rare in the internet, loss is not a good signal of congestion.

   D None of the above.

Answer:

Item 3 is not true. Losses occur on the internet, and they occur for a wide variety of reasons. At Google, losses occur for shallow-buffered switches as a result of coincident arrivals of small traffic bursts. Consequently, TCP CUBIC, a loss-based algorithm, would interpret this as congestion and fail to utilize the link fully. Delay-based algorithms like BBR achieve much better link utilization.

II Netflix at Scale

2. [3 points]:

In Ken Florance's lecture, he highlighted how Netflix streaming works, and that traffic on the internet has changed drastically since 1995. Based on his lecture, which of the following statements are true? (Select all that apply.)

(A) Today, the vast majority of data flows towards users, rather than bidirectionally like it was in 1995.

B Netflix leverages a single highly-optimized datacenter than serves all of their users.

(C) Netflix uses proactive caching to load balance and improve performance of their content distribution network.

D None of the above.

Answer:

Item 2 is not true. Netflix runs one of the largest content distribution networks in the world—pushing content out towards the edges near users, rather than having everything in a single place.

III Running a Cloud Network

3. [3 points]:

Jeff Mogul from Google gave a lecture where he highlighted considerations of a "cloud network", including network virtualisation and scalability. Based on his lecture, which of the following statements are true? (Select all that apply.)

A Customers like using cloud providers because it is often cheaper than managing their own systems—allowing them to grow or shrink their resources flexibly.

B Cloud providers use virtual networks and functional isolation to isolate customers from each other and give flexibility in distributing resources.

C Datacenter networks achieve high reliability from cheap, low-reliability components (e.g, using Clos topologies).

D None of the above.

IV Nick's Midlife Crisis

4. [3 points]:

Which of the following stakeholders stand to benefit from the paradigm of software-defined networking? (Select all that apply.)

(A) Companies like Facebook with large datacenters

B Router manufacturers

(C) Networking researchers

D None of the above

Answer:

SDN facilitates better link utilization due to traffic engineering, which makes certain types of packet loss acceptable in datacenters. Router manufacturers like Cisco depend on their ability to sell specialized, proprietary hardware; SDN enables consumers to buy simpler hardware for their routing needs. Networking researchers have many reasons to like SDN, including cheaper hardware, more flexibility, and less cruft from commercial routers trying to solve all consumers' needs at once.

5. [2 points]:

True or false: The control plane in a software-defined network must operate from one central entity (as opposed to being distributed across routers).

A True

(B) False

Answer:

The ability to have a centralized hypervisor with knowledge of a network's state is an important feature of SDN, but the control plane can also be implemented at the individual router level using a language like P4 specifically meant for programmable switches. Centralized and decentralized control planes are both common in practice.

V Integrity

6. [7 points]:

In this class, we've discussed several mechanisms that try to preserve the integrity of messages sent across a network and detect if a message has been modified. For each of the following "integrity check" mechanisms, here are some possible use cases that we might want to use the check for:

(A) Detecting if a message was modified by the random bitflips of a noisy link.

(B) Detecting and repairing bitflips in a message.

(C) Detecting messages that have been modified nefariously, when the receiver knows the value of the integrity check (e.g. the checksum) for sure.

(D) Detecting messages that have been modified nefariously (when the integrity check is transmitted as part of the message).

(E) Sending messages to the public, and letting any receiver make sure they have a correct, unmodified copy.

(F) Keeping messages secret from eavesdroppers.

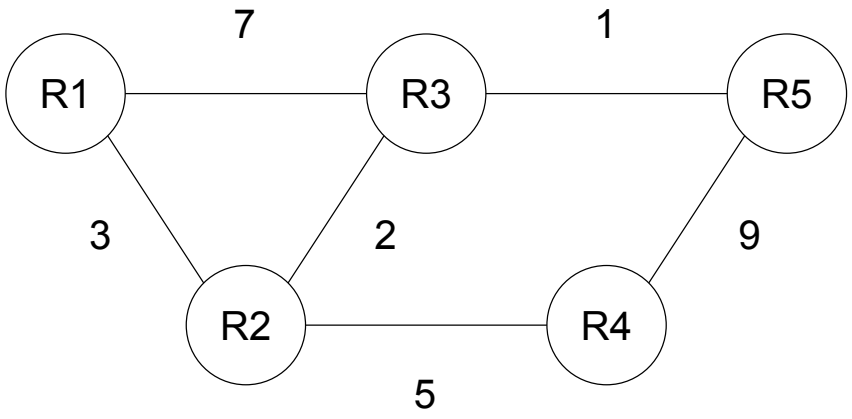
For each of the following "integrity check" mechanisms, please circle **all** use cases that the mechanism can reasonably and securely support.

Checksums	A	B	C	D	E	F
Cyclic redundancy checks (CRCs)	A	B	C	D	E	F
Hash functions (e.g. Java `hashCode()`)	A	B	C	D	E	F
Secure hash functions (e.g. SHA-256)	A	B	C	D	E	F
Secure hash of {a secret key + the message}	A	B	C	D	E	F
Message Authentication Codes (MACs)	A	B	C	D	E	F
Public-key signatures	A	B	C	D	E	F

VI Bellman-Ford Routing (TODO better name?)

7. [15 points]:

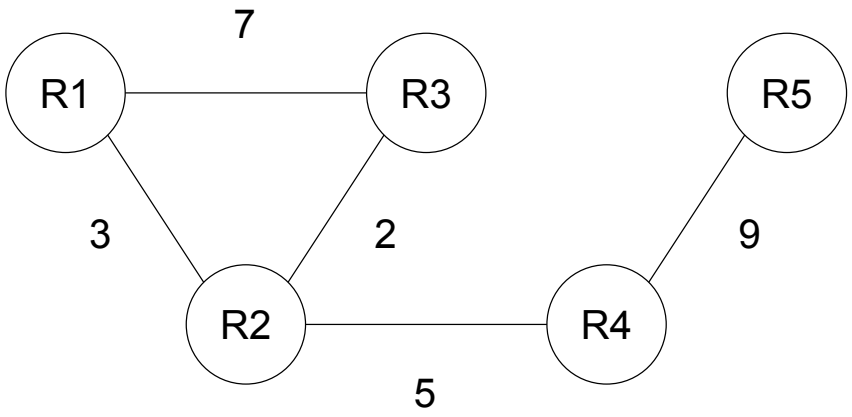
The figure below shows a network of five routers, some of which are connected by links. Each link has a cost. For example, the link from R1 to R2 has a cost of 3.



a. Use the Bellman-Ford algorithm to complete the table below. A row in the table corresponds to R5's distance vector at an iteration of the algorithm. Each entry in the table (element of a distance vector) consists of a router's lowest known cost to R5 as well as the next hop in the lowest-cost path, if applicable. The first two rows are completed for you. Stop filling in the table when further iterations do not change the distance vector. You may not need all rows. Ties are broken randomly.

Iteration	R1	R2	R3	R4
0	∞	∞	∞	∞
1	∞	∞	*1, R5*	*9, R5*
2	*8, R3*	*3, R3*	*1, R5*	*9, R5*
3	*6, R2*	*3, R3*	*1, R5*	*8, R2*
4				
5				

b. Now assume that the link between R3 and R5 fails, as in the figure below.



Using your final distance vector from part (a) as the 0^{th} iteration, how many iterations does it take for the Bellman-Ford algorithm to converge on a new distance vector (not including the 0^{th} iteration)? Assume that **no optimizations are used to improve the algorithm**. We will only grade your final answer, but showing work in the table below will make you eligible for partial credit.

Iteration	**R1**	**R2**	**R3**	**R4**
0	*6, R2*	*3, R3*	*1, R5*	*8, R2*
1	*6, R2*	*3, R3*	*5, R2*	*8, R2*
2	*6, R2*	*7, R3*	*5, R2*	*8, R2*
3	*10, R2*	*7, R3*	*9, R2*	*9, R5*
4	*10, R2*	*11, R3*	*9, R2*	*9, R5*
5	*14, R2*	*11, R3*	*13, R2*	*9, R5*
6	*14, R2*	*14, R4*	*13, R2*	*9, R5*
7	*17, R2*	*14, R4*	*16, R2*	*9, R5*
8				

Number of Iterations: *7*

c. In some cases, removing a link from a topology prevents the Bellman-Ford algorithm from terminating. One technique to prevent infinite loops is to set "infinity" to some small integer. Using this finite substitute for infinity ensures that the Bellman-Ford algorithm will arrive at a stable distance vector, but that vector might not indicate the correct routes. What is the minimum value of infinity needed to guarantee that running Bellman-Ford on the scenario in part (b) will result in every router knowing the correct path (not necessarily the correct cost) to R5? You need not justify your answer, but a brief explanation will make you eligible for partial credit.

Value of Infinity: *14*

Answer:

If infinity is set to a value lower than 14, R2 will never accept that its best route is through R4. Testing the algorithm with $\infty = 14$, we see that R1 and R3 correctly realize they should go through R2.

d. The Bellman-Ford algorithm's convergence on a correct distance vector in part (b) took more iterations than strictly necessary. **In no more than 30 words**, describe an addition to the Bellman-Ford algorithm (introducing no new tools, i.e., only using route advertisements) that lets the algorithm reach the correct answer in fewer iterations when the link from R3 to R5 fails. You must explain how your proposal works, but you need not walk through what happens when you apply it to the scenario in part (b). Complete sentences are not required.

Answer:

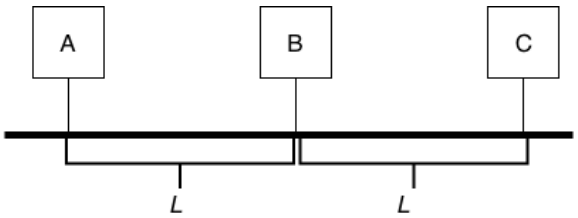
Split horizon: *R2 can refuse to advertise its route to R3, since R3 is its next hop. This prevents the counting-to-infinity loop.*

Split horizon with poison reverse: *When the link fails, R3 can advertise an infinite cost to R5. If R1 also refuses to advertise its route to R2–since R2 is its next hop–then R2 will switch its next hop to R4.*

VII Who Speaks Next?

8. [10 points]:

Suppose we had the following topology on a shared Ethernet cable.



A and B are separated by distance L, as are B and C. Packets travel across the wire at the speed of light c. Suppose that A to C at time 0. The hosts are following CSMA/CD: so they will not send a packet if they sense that the wire is busy.

a. Suppose that at time $t > 0$, C sends a packet that collides with A's packet. What is the latest time t that C can send the packet, and how long after A sends its packet will A hear about the collision? Express your answers in terms of L and c.

Answer:

C sends a packet right before the head of A's packet reaches it. This happens at $t = 2L/c - \epsilon$. It then takes another $2L/c$ for the collision to reach A. So A hears the collision at $4L/c - \epsilon$.

b. What is the earliest time $t > 0$ that C could have sent the packet such that the packet collides with A's? How long after A sends its packet will A hear the collision? Express your answers in terms of L and c.

Answer:

C sends a packet at $t = \epsilon$ (close to 0). The collision occurs at L/c, and it takes another L/c for A to hear about it. So A hears the collision at $2L/c + \epsilon$

c. What is the minimum allowable size of the packet that A sends to C. Express your answer in terms of L and c.

Answer:

$4L/c$. The packet needs to be long enough that A is still sending the packet when it hears of a collision with the same packet.

d. CSMA/CD does not translate very well to wireless networks. In one sentence, name one reason that CSMA/CA is much more suitable to wireless networks that CSMA/CD. Then (in one sentence) explain why CSMA/CD does not face this problem on a shared Ethernet cable as above.

Answer:

You aren't guaranteed to know if a collision occurred in wireless while you're still sending the packet, so you have to depend on an acknowledgement. On shared cable the interfering signal will propagate to the sender without much attenuation, so you can definitely hear the collision at some point.

VIII Punching Holes

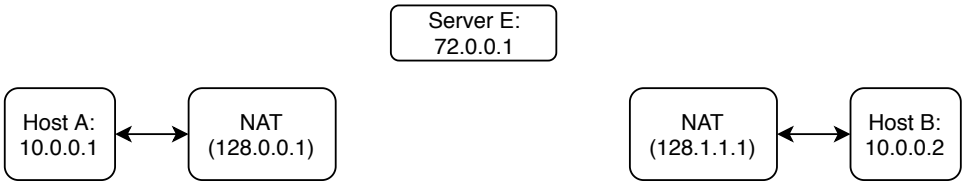
9. [10 points]:

In this question, we'll explore some scenarios for hole punching in NATs.

As a reminder: we have a few different types of NATs. Let an address be an IP:Port pair.

- **Full Cone**: Internal addresses ($i\_ipaddr : i\_port$) are mapped to external addresses ($e\_ipaddr : e\_port$). Messages from any outside host directed to (e_ipaddr, e_port) can traverse the mapping.
- **Port Restricted**: Full cone mapping, but messages from an outside host (h_ipaddr, h_port) can only traverse the mapping if (i_ipaddr, i_port) has previously sent a message to (h_ipaddr, h_port).
- **Symmetric:** A symmetric NAT is port-restricted. AdditionallyEach request from ($i\_ipaddr : i\_port$) to a specific destination (h_ipaddr, h_port) is mapped to a unique external address (e_ipaddr, e_port). If (i_ipaddr, i_port) tries sending to a different outside host, the NAT will assign a new external address.

Suppose that hosts **A** and **B** are behind *different* NATs. There is also an external server E that is not behind a NAT.



A and B both send packets to E in order to establish mappings. These are the packets they send (as well as what the server receives).

E then tells A what packet it received from B, and tells B what packet it received from A. You can assume that mappings don't expire.

a. Suppose the NAT is full-cone. A and B want to set up a direct connection to each other that is not through E. Describe: (1) what packet A should send to B and the translated packet B receives and (2) the packet B should send to A and the translated packet A receives in order for the packets to successfully reach the other side. Your packets should be in terms of source IP:port and destination IP:port below.

A sends the packet:
Src IP:port _____ : ___
Dst IP:port _____ : ___
B receives the translated packet:
Src IP:port _____ : ___
Dst IP:port _____ : ___

B sends the packet:
Src IP:port _____ : ___
Dst IP:port _____ : ___
A receives the translated packet:
Src IP:port _____ : ___
Dst IP:port _____ : ___

Answer:

A sends the packet $[10.0.0.1 : 8888, 128.1.1.1 : 90]$ *and B receives* $[10.0.0.2 : 9999, 128.0.0.1 : 70]$. *B sends the packet* $[128.0.0.1 : 70, 10.0.0.2 : 9999]$ *and A receives* $[128.1.1.1 : 90, 10.0.0.1 : 8888]$

b. From part (a) above, let P_A be the packet A sends to B and P_B be the packet B sends to A. Suppose that the NAT is port-restricted instead of full-cone.

A and B try the following scheme: A and B alternate sending packets: A sends P_A to B, then B sends P_B to A, then A send P_A to B: so on and so forth. This scheme continues for a total of 3 back and forths (6 packets). P_A and P_B do not change over time. You can assume that the next packet is sent after the previous packet was either received (or dropped by the NAT), and that packets are transmitted reliably.

Let a successful packet be a packet that reaches the other side without being dropped by the opposing NAT. Under this scheme, is it possible for a successful packet to be transmitted? If so, how many packets of the 6 are successful? If not (in at most three sentences), explain why the scheme fails.

Answer:

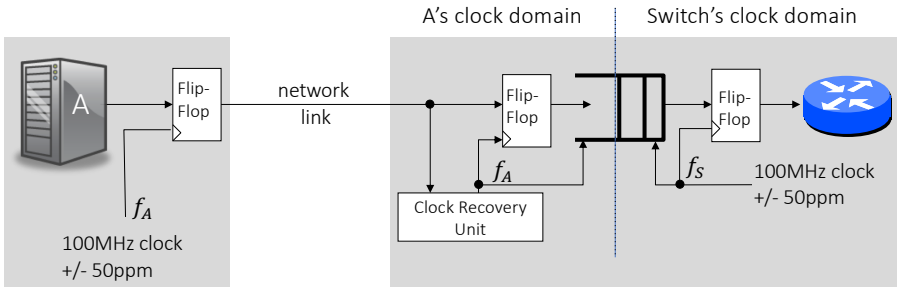
Yes it's possible. A will first send a packet to B, which will be dropped by B's NAT: however, now A will have established a mapping to B. Then B will send to A, and this packet will be successful, and all further packets will be successful: so 5 out of the 6 packets will reach

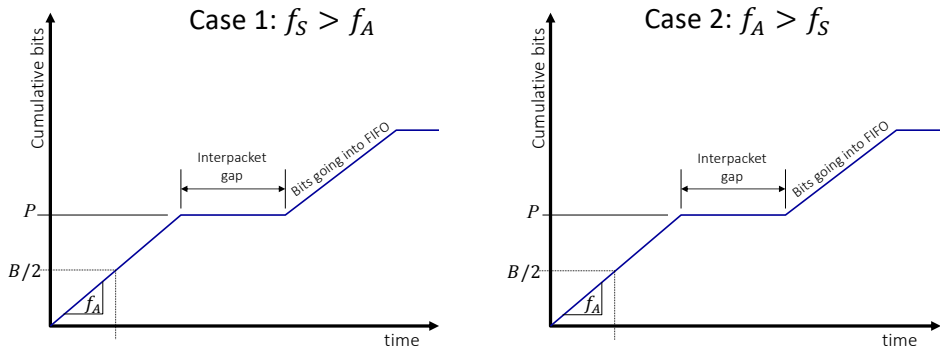
c. Suppose now the NAT is symmetric, and A and B try the same scheme as in part (b). Under this scheme, is it possible for a successful packet to be transmitted? If so, how many packets of the 6 are successful? If not (in at most three sentences), explain why the scheme fails.

Answer:

The scheme will fail. A will send P_A, but because the NAT is symmetric, P_A will be translated to some other mapping. Therefore, the mapping established by E (which is what B is using) is useless.

IX Elasticity Buffer





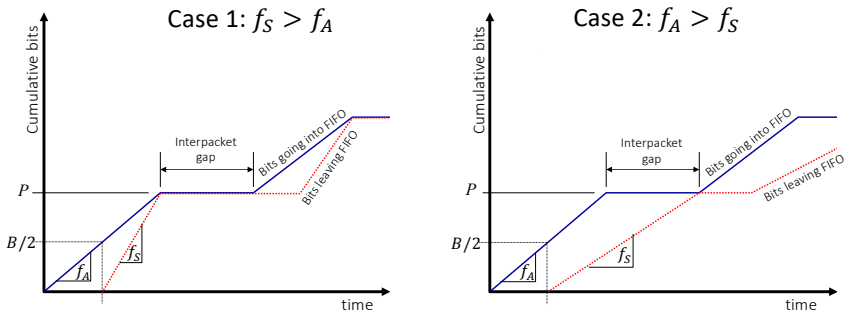
10. [10 points]:

Host A sends packets of length P bits to a switch over a 100Mb/s Ethernet link. Both ends of the link use 100MHz clocks ±50ppm. An elasticity buffer is used by the switch to transfer packets from Host A's clock domain to the switch's clock domain. When a new packet arrives, the switch always waits until the FIFO is half full ($B/2$ bits) until it starts reading from the FIFO.

a. Case 1: The switch's clock frequency f_S is faster than host A's clock frequency f_A. Sketch the progress of bits departing from the switch's elasticity FIFO on the graph for Case 1. Show the case when the buffer just avoids underflowing.

b. Case 2: The switch's clock frequency f_S is slower than host A's clock f_A. Sketch the progress of bits departing from the switch's elasticity FIFO on the graph for Case 2. Show the case when the buffer just avoids overflowing.

Answer:



c. How large does the interpacket gap need to be (in bits) to prevent an overflow? Express your answer as a function of B.

Answer:

The interpacket gap should be sufficient for a full elasticity buffer to drain before the next packet arrives. i.e. B bits.

X TCP-in-TCP

11. [18 points]:

In this question, our host has a TCP connection, the *outer connection*, with a remote peer. The outer connection's reliable datastream is exactly the TCP segments sent and received by another TCP connection, which we call the *inner connection*.

You can assume assume all of the following:

- Both connections use *stop-and-wait* TCP, and both retransmit after *exactly* 10 ms.
- Every transmitted and received segment of the outer connection contains *exactly* one inner connection segment (including headers) as its payload.
- Each of the inner connection's segments' headers occupies *exactly* 20 bytes.
- Outer connection segments can be dropped when they are sent to the remote peer. Inner connection segments are never dropped.
- The outer connection's RTT is *exactly* 5 ms; segments are never delayed (assuming they are not dropped). There is no processing delay at the remote peer.
- When the outer connection receives a segment from the remote peer, there is a processing delay of *exactly* 1 ms. After this, two events happen simultaneously:
  - The outer connection passes the payload of the received segment to the inner connection.
  - The outer connection transmits its next waiting segment to the remote peer, if there is such a segment.
- When the inner connection delivers a segment to the outer connection, there is a processing delay of 1 ms. After this, the outer connection encapsulates and sends the segment to the remote peer if there is no other segment in flight.

  If there is already a segment in flight, the outer connection queues the segment from the inner connection, and sends it when it can. (See the timing rules for sending the next waiting segment, described above).
- We are only concerned with the ESTABLISHED state, so every segment has the ACK flag set, and no other flags are ever set.

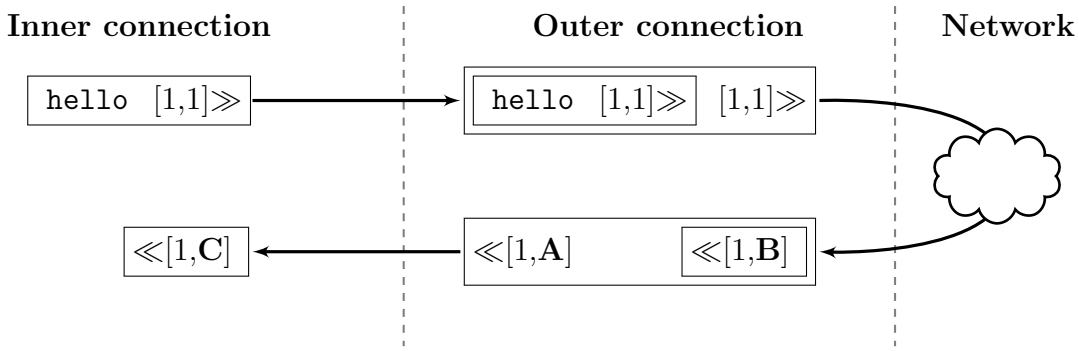
We use the following notation to denote segments and their payloads:

- An outbound TCP segment is represented by $\boxed{\text{payload} \quad [\text{seqno,ackno}]\gg}$.
- An inbound TCP segment is represented by $\boxed{\ll[\text{seqno,ackno}] \quad \text{payload}}$.
- Inner segments encapsulated in outer segments are thus

  $\boxed{\boxed{\text{inner payload} \quad [\text{iseq,iack}]\gg}\ [\text{oseq,oack}]\gg}$ (sending)

  $\boxed{\ll[\text{oseq,oack}]\ \boxed{\ll[\text{iseq,iack}] \quad \text{inner payload}}}$ (receiving)

a. Consider the following sequence of segments. Assume there is no loss.



What are the acknowledgment numbers **A**, **B**, and **C**?

A:_____ **B**:_____ **C**:_____

What is the delay between the inner connection's sending its segment and receiving the corresponding acknowledgment?

_____ ms

Answer:

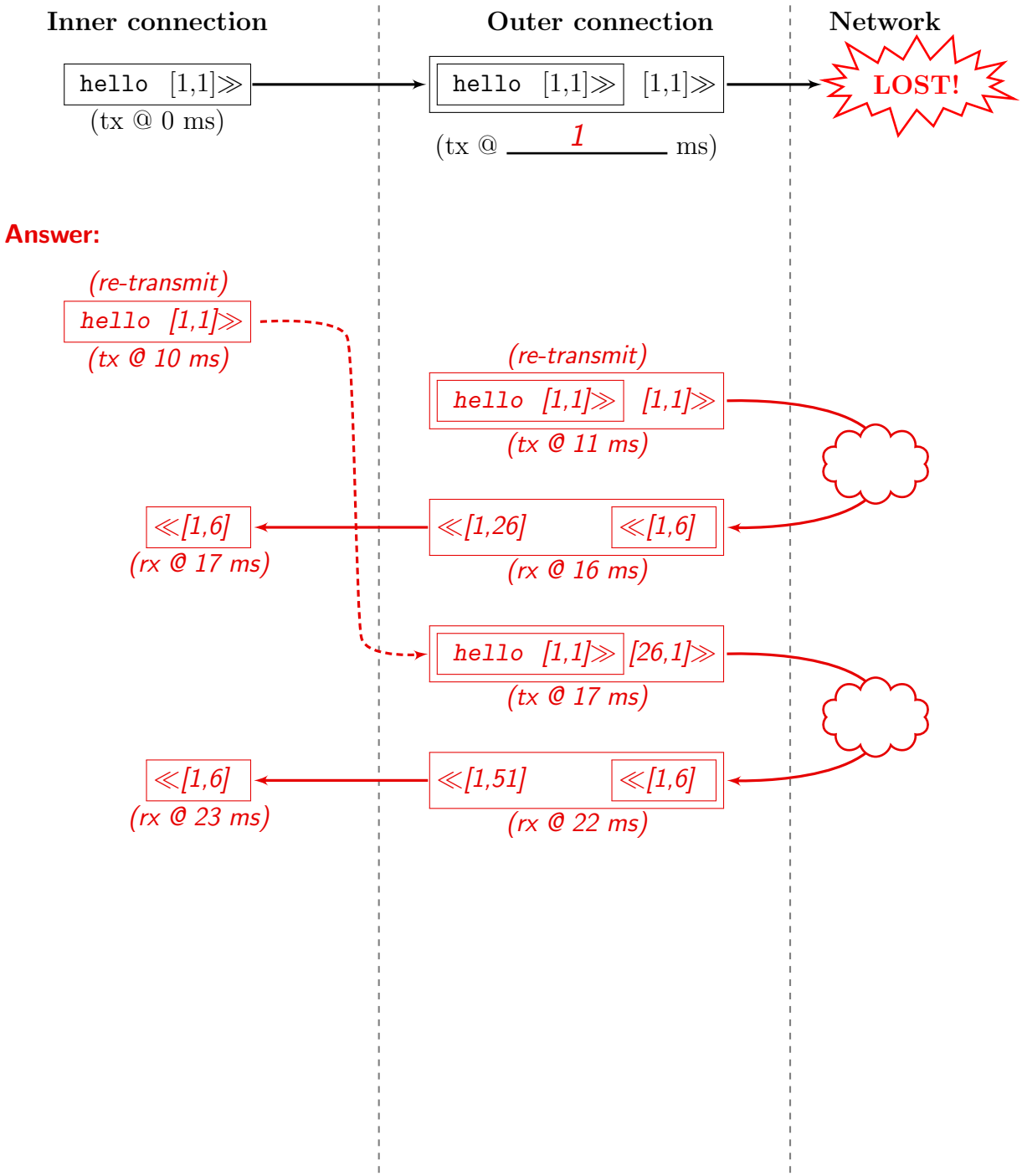
A = 26, B = C = 6, delay = 7 ms

b. This time, let's think about what happens if there is loss in the network.

Draw the sequence of packets that results when the first segment sent to the network is lost, using the same notation as in the prior part. (Please draw neatly!)

Be sure to include sequence and acknowledgment numbers.

Also, assuming that the inner connection sends its first segment at time 0, **label every segment with the time at which it is transmitted or received**.



c. Based on the behavior from the prior part, describe in two or three sentences (50 words max) what goes wrong when running TCP-in-TCP over a lossy network.

Answer:

TCP assumes it is running over best-effort datagrams, but TCP-in-TCP breaks this assumption. As a result, when the inner connection re-transmits, it ends up causing the outer connection to (reliably) send useless data.

d. Other protocols that we've discussed in CS144 would be a better choice for the outer connection. Please name one, and explain your answer (20 words max).

Answer:

UDP is a better choice, because it gives a datagram abstraction, which matches TCP's design.

IP is also a good choice, but note that in general unprivileged programs are not allowed to send and receive raw IP datagrams.

XI Putting It All Together

12. [15 points]:

You plug a new laptop into a wired Ethernet jack for the first time. You have already told the network administrators your MAC address, and can join the network. After your laptop has successfully connected to the internet, you type the following URL into your web browser:

`http://stanford.example.edu/inquiry/onlineinq.html`

Assume only the following:

- Your Ethernet address is `aa:bb:cc:dd:ee` and you'll be assigned the IP address `171.64.7.10`.
- Your web browser is using TCP, not something like QUIC.
- Your DHCP server is `171.64.7.99`.
- Your DNS resolver is `171.64.7.88`.
- Your gateway is `171.64.7.77`, with Ethernet address `00:11:22:33:44:55`.
- You have an empty ARP cache.
- Neither your laptop nor your DNS resolver have any cached DNS entries.
- DNS never needs to fail over to TCP.
- You do not request a persistent connection.
- The HTML response returns 200 OK with a web page.
- The HTML request and response each fit in a single segment.
- The web page does not require loading any additional resources.

(Question on next page. . .)

Arrange the following sequence of events that occur for your host to receive the web page. Do not make any additional assumptions. Note that you do not have to include all of the options, leave out sequences you believe are irrelevant or incorrect. Write your answer in the form (`XYZ...`) on the answer line below.

A. The resolver on `171.64.7.88` iterates from root server (for `edu`) to TLD server (for `example`) to Example's server (for `stanford`), finally getting the address and sending it back to you using UDP.

B. The DHCP server sends a DHCP ack to `aa:bb:cc:dd:ee`.

C. The browser checks if the domain is in its DNS cache.

D. You send a DNS `A` request for `stanford.example.edu` to `171.64.7.88`, using UDP port 53.

E. You broadcast a DHCP request for `171.64.7.10`.

F. The server sends a `SYN-ACK` to you.

G. You send an ARP request for `171.64.7.77`.

H. You send a `GET` request as TCP data. The `GET` request is for `/inquiry/onlineinq.html`.

I. You send a TCP `SYN` packet to port 80 of the destination IP address returned by the resolver.

J. You and the server perform the TLS handshake.

K. The DHCP server sends a DHCP offer of `171.64.7.10` to `aa:bb:cc:dd:ee`.

L. The server sends back the web page and closes its sending end of the connection, sending a `FIN` packet.

M. The browser parses the URL to understand the protocol and resource requested.

N. Your laptop broadcasts a DHCP discover message from `aa:bb:cc:dd:ee`.

O. Your browser renders the web page.

Answer: _____

Answer:

The correct sequence is: `NKEBMCGDAIFHLO`.

Item J is ignored since this website is using HTTP, not HTTPS, and we cannot assume that we will be redirected to a HTTPS version of the page.