ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Computer Networks - Midterm

October 31, 2014

Duration: 2:15 hours.

- This is a closed-book exam.
- Please write your answers on these sheets in a readable way, in English or in French.
- You can use extra sheets if necessary (don't forget to put your name on them).
- The total number of points is 100.
- This document contains 19 pages.
- Good luck!

Full Name (Nom et Prénom):

SCIPER No:

Division: ☐ Communication Systems ☐ Computer Science
☐ Other (mention it): . . . . . .

Year: ☐ Bachelor Year 2 ☐ Bachelor Year 3
☐ Other (mention it): . . . . . .

(answers to the questions are shown in italic and blue) (grades in red)

1 Short questions (10 points)

For each question, please circle a single best answer.

1. In a circuit switched network we can have
   (a) data loss, unpredictable delay.
   (b) data loss, predictable delay.
   (c) no data loss, unpredictable delay.
   (d) no data loss, predictable delay. *(Correct)*

2. Alice is connected to the Internet using a 100Mbps connection, Bob is connected to the Internet using a 10Mbps connection. When Alice sends a 100Mb file to Bob, the transfer will take
   (a) less than 10 seconds.
   (b) exactly 10 seconds.
   (c) more than 10 seconds. *(Correct)*
   (d) we cannot say precisely, all the above can be true.

3. Consider a switch with a buffer size of 20Mb and its output link with transmission rate of 10Mbps. The switch may introduce a maximum queuing delay of
   (a) 0 seconds.
   (b) less than 2 seconds. *CORRECT*
   (c) more than 4 seconds.
   (d) ∞

4. A host uses HTTP to access a web page that consists of a base file and one picture, both stored on the same web server. How many round-trip-times (RTTs) are required to retrieve the entire web page?
   (a) exactly 2 RTTs
   (b) at most 2 RTTs
   (c) at least 3 RTTs *(Correct)*
   (d) at most 3 RTTs

5. The following constitutes an attack against the DNS protocol:
   (a) impersonate the local DNS server.
   (b) denial-of-service against the root or TLD servers.
   (c) poison the cache of the local DNS server.
   (d) all of the above. *CORRECT*

6. A peer-to-peer application

   (a) always uses UDP, because we don't have dedicated infrastructure for the server.

   (b) always uses TCP, because peers act both as clients and servers.

   (c) may use TCP or UDP as a transport layer protocol. *(Correct)*

   (d) doesn't use any transport layer protocol.

7. A valid set of actions performed by a TCP client is:

   (a) create socket, wait for connection setup, receive service request, send reply, close the connection.

   (b) create socket, initiate connection setup, send service request, receive reply. *CORRECT*

   (c) create socket, send service request, receive reply, close the connection.

   (d) create socket, send service request, receive reply.

8. A web server that implements persistent HTTP communicates using

   (a) a different socket for each request.

   (b) the same socket for all requests.

   (c) a different socket for each connecting client. *CORRECT*

   (d) a different socket for each web page served.

9. For a Go-Back-N protocol, if the window size of the sender is N, then the window size of the receiver is

   (a) 1. *(Correct)*

   (b) N.

   (c) $\frac{N}{2}$.

   (d) any value smaller than N.

10. An application uses UDP as a transport layer. If the application wants to have reliable transfer it can implement

    (a) a stop-and-wait protocol.

    (b) a selective repeat protocol.

    (c) a go-back-N protocol.

    (d) any of the above. *(Correct)*

2 Application layer (30 points)

Setup:

Three users are logged into the workstations `user1.epfl.ch`, `user2.epfl.ch` and `user3.epfl.ch`, all located inside EPFL's network.

EPFL offers a web server `www.epfl.ch`, a local name server `ns.epfl.ch`, and a web proxy `proxy.epfl.ch`.

A web server `www.example.com` is located outside EPFL's network.

Make the following assumptions:

- All the computers inside EPFL's network use `ns.epfl.ch` as their local DNS server.
- `ns.epfl.ch` is also the *authoritative* DNS server for the `epfl.ch` domain.
- All DNS queries are resolved *iteratively*.
- All DNS servers and workstations maintain DNS caches.
- Web browsers and web proxies perform caching.
- All caches are initially empty. This is true for each question.
- Application layer messages fit in one packet.
- For simplicity, assume nobody else is generating traffic on the Internet.

Question 1 (10 points):

User 1 browses `http://www.epfl.ch/index.html`. The only object referenced by the base file is the image `http://www.epfl.ch/image.png`. The web browser uses a *single non-persistent* connection.

List all the packets that are exchanged, including any connection setup packets. Show the source and destination of each packet, the application protocol, the transport protocol, and the purpose of the packet. Fill in the answer in Table 1.

Packet	Source	Destination	Application protocol	Transport protocol	Message
1	`user1.epfl.ch`	`ns.epfl.ch`	DNS	UDP	Name: www.epfl.ch
2	`ns.epfl.ch`	`user1.epfl.ch`	DNS	UDP	IP: `www.epfl.ch`
3	`user1.epfl.ch`	`www.epfl.ch`	HTTP	TCP	TCP SYN
4	`www.epfl.ch`	`user1.epfl.ch`	HTTP	TCP	TCP SYN ACK
5	`user1.epfl.ch`	`www.epfl.ch`	HTTP	TCP	GET /index.html
6	`www.epfl.ch`	`user1.epfl.ch`	HTTP	TCP	200 OK ... *index.html*
7	`user1.epfl.ch`	`www.epfl.ch`	HTTP	TCP	TCP SYN
8	`www.epfl.ch`	`user1.epfl.ch`	HTTP	TCP	TCP SYN ACK
9	`user1.epfl.ch`	`www.epfl.ch`	HTTP	TCP	GET /image1.png
10	`www.epfl.ch`	`user1.epfl.ch`	HTTP	TCP	200 OK ... *image1.png*

Table 1: Packets transmitted on the Internet, in Question 1.

Question 2 (12 points):

User 2 browses `http://www.epfl.ch/help.html` through the web proxy. This web page does not reference any objects. The browser uses a *single non-persistent* connection.

List all the packets that are exchanged, including any connection setup packets. Show the source and destination of each packet, the application protocol, the transport protocol, and the purpose of the packet. Fill in the answer in Table 2.

Packet	Source	Destination	Application protocol	Transport protocol	Message
1	`user2.epfl.ch`	`ns.epfl.ch`	DNS	UDP	Name: proxy.epfl.ch
2	`ns.epfl.ch`	`user2.epfl.ch`	DNS	UDP	IP: `proxy.epfl.ch`
3	`user2.epfl.ch`	`proxy.epfl.ch`	HTTP	TCP	TCP SYN
4	`proxy.epfl.ch`	`user2.epfl.ch`	HTTP	TCP	TCP SYN ACK
5	`user2.epfl.ch`	`proxy.epfl.ch`	HTTP	TCP	GET /help.html Host: www.epfl.ch
6	`proxy.epfl.ch`	`ns.epfl.ch`	DNS	UDP	Name: www.epfl.ch
7	`ns.epfl.ch`	`proxy.epfl.ch`	DNS	UDP	IP: `www.epfl.ch`
8	`proxy.epfl.ch`	`www.epfl.ch`	HTTP	TCP	TCP SYN
9	`www.epfl.ch`	`proxy.epfl.ch`	HTTP	TCP	TCP SYN ACK
10	`proxy.epfl.ch`	`www.epfl.ch`	HTTP	TCP	GET /help.html
11	`www.epfl.ch`	`proxy.epfl.ch`	HTTP	TCP	200 OK ... *help.html*
12	`proxy.epfl.ch`	`user2.epfl.ch`	HTTP	TCP	200 OK ... *help.html*

Table 2: Packets transmitted on the Internet, in Question 2.

Question 3 (4 points):

User 1 is about to browse `http://www.example.com/`.

User 3 knows this and wants to intercept User 1's web traffic. For this purpose, User 3 wants User 1 to connect to `user3.epfl.ch` instead of `www.example.com`.

Explain how User 3 can achieve this and describe the DNS messages that are exchanged.

User 3 knows that User 1 will ask the local DNS server `ns.epfl.ch` for the IP address of `www.example.com`.

User 3 continuously sends DNS responses to User 1 for name `www.example.com`. These responses contain a faked record of type `A` with the IP address of `user3.epfl.ch`, and appear to be coming from `ns.epfl.ch` (the source IP address of these responses is spoofed).

If any of the fake responses reaches User 1 before the correct response from `ns.epfl.ch` arrives, User 1 will obtain and use the fake IP address for `www.example.com`. Thus, User 1 will connect to `user3.epfl.ch`, instead.

Question 4 (4 points):

User 3 wants to intercept the web traffic between all the computers inside EPFL (not just User 1) and the website `http://www.example.com/`.

For this purpose, User 3 wants them to connect to `user3.epfl.ch` instead of `www.example.com`.

Explain how User 3 can achieve this, and describe the DNS messages that are exchanged.

User 3 asks the local DNS server `ns.epfl.ch` for the IP address of `www.example.com`. If we assume that `ns.epfl.ch` does not have the response cached, User 3 knows that `ns.epfl.ch` will have to make further DNS queries to obtain the answer.

User 3 also knows that `ns.epfl.ch` resolves DNS queries iteratively. Therefore, it expects that in the process, `ns.epfl.ch` will eventually send a DNS query to the DNS server responsible for `example.com` (let's say that this is `ns.example.com`) in order to obtain the IP address of `www.example.com`.

Similar to the previous question, User 3 will try to make `ns.epfl.ch` accept a fake response. User 3 will continuously send fake responses to `ns.epfl.ch` for name `www.example.com`. These responses contain a faked record of type `A` with the IP address of `user3.epfl.ch`, and appear to be coming from `ns.example.com` (the source IP address of these responses is spoofed).

If any of these fake responses reaches the DNS server `ns.epfl.ch` before the correct response from `ns.example.com` arrives, the DNS server will accept the fake record and cache it. Now, whenever an EPFL user makes a DNS request for `ns.example.com`, they will be given the IP address for `user3.epfl.ch`, instead.

3 Network delays (30 points)

Suppose the following:

- End-nodes $1, 2, 3, 4, ..., N$ ($N \geq 5$) are part of a network, shown in Figure 1. The nodes are connected via switches S_1 and S_2, which perform packet switching using *store-and-forward*.
- The size of the packet buffer at every switch is infinite and processing delays can be ignored.
- Each of the physical links (solid lines) has length ℓ meters and propagation speed c.
- The transmission rate of the links that connect the end-hosts to the switches is R (in both directions).
- The transmission rate of the link that connects S_1 and S_2 is $\frac{R}{k}$, $k > 1$ (in both directions).
- All nodes are organized in a *one-way* circular DHT (shown with dashed lines in Fig. 1). Each node is aware only of its subsequent neighbor; e.g., Node 2 knows only about Node 3.
- The one-way circular DHT is implemented on top of UDP.
- Node 1 stores a *movie* of size F bits and Node 2 is responsible for storing information about this movie.

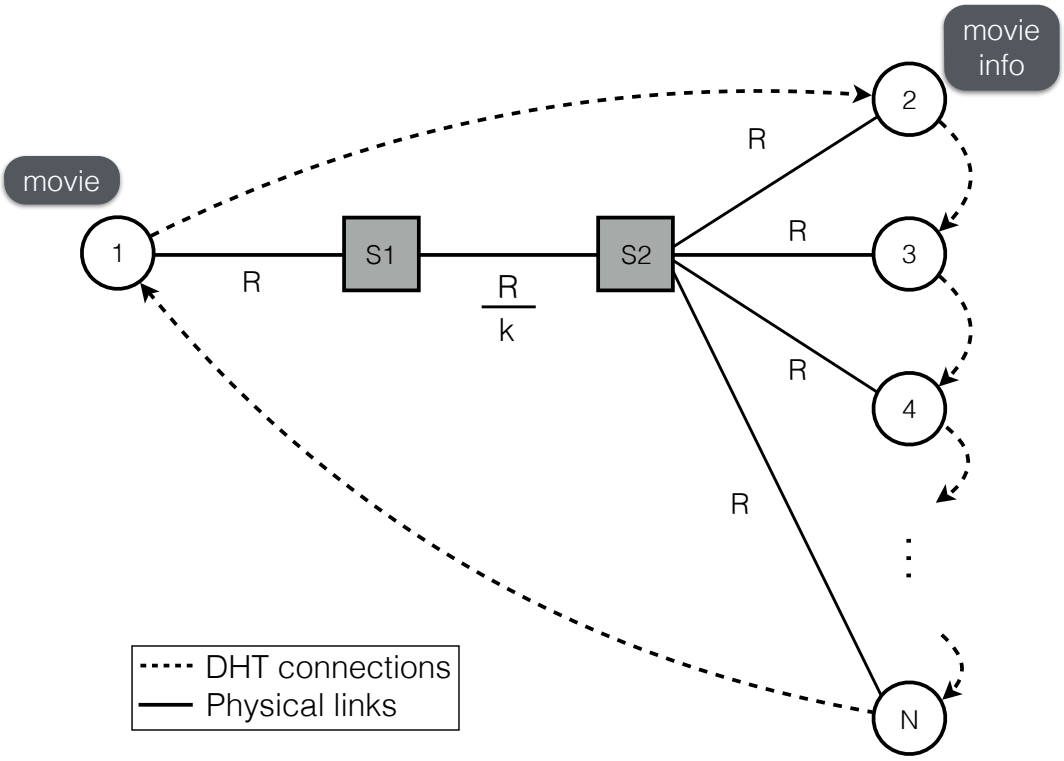

Figure 1: The setup for the exercise on network delays.

Question 1 (5 points):

Suppose that any DHT query/request message fits into one packet of size q bits. Compute the total time that a node i needs to learn where the movie is stored, for the following cases:

a. $i = N$

b. $i = 3$

The propagation delay is $d_{prop} = \frac{\ell}{c}$ for every link.

a. Node N has to ask node 1, then node 1 has to ask node 2 and finally node 2 must reply back to node N.

So,

$$d_{N\to 1} = \frac{q}{R} + \frac{q}{R/k} + \frac{q}{R} + 3 \cdot d_{prop}$$
$$d_{1\to 2} = \frac{q}{R} + \frac{q}{R/k} + \frac{q}{R} + 3 \cdot d_{prop}$$
$$d_{2\to N} = \frac{q}{R} + \frac{q}{R} + 2 \cdot d_{prop}$$

Thus

$$d_{total_1} = (6 + 2k) \cdot \frac{q}{R} + 8 \cdot \frac{\ell}{c}$$

(2 pts)

b. The query has to be propagated clockwise from node 3 to node 2 and then node 2 has to reply back to node 3: $d_{i\to(i+1)} = \frac{q}{R} + \frac{q}{R} + 2 \cdot d_{prop}$, for $3 \le i \le N-1$.

So,

$$\begin{aligned} d_{total_2} &= \sum_{i=3}^{N-1} d_{i\to(i+1)} + d_{N\to 1} + d_{1\to 2} + d_{2\to 3} \\ &= (N-3)\left[\frac{q}{R} + \frac{q}{R} + 2 \cdot d_{prop}\right] + \left[(4+2k)\frac{q}{R} + 6 \cdot d_{prop}\right] + \left[\frac{q}{R} + \frac{q}{R} + 2 \cdot d_{prop}\right] \\ &= 2(N+k)\frac{q}{R} + 2(N+1)\frac{\ell}{c}. \end{aligned}$$

(3 pts)

Comment about grading: *Students who considered that node 1 may also directly reply back to the query "Who has the movie?" and made it clear, received full credit. This assumes that the resulting delay calculation was done correctly.*

Question 2 (3 points):

What mechanism would you use to reduce the average delay experienced by the nodes in learning where the movie is stored? Give a mechanism example and explain how would it affect the delay experienced by Node 3?

We can use "shortcuts", so that each peer not only keeps track of its immediate successor, but also of a relatively small number of shortcut peers scattered about the DHT circle. Shortcuts are generally used to expedite the routing of the query messages.

***Example:** Assume a DHT where each node keeps track of its two subsequent nodes. In that case, in order to resolve node 3's query, we only require half the number of messages to be exchanged. (compared to the number of messages needed to resolve the same query in the original DHT of Fig. 1). Hence, the delay experienced by node 3 will be generally lower. (3 pts)*

Question 3 (12 points):

After querying the DHT, all Nodes $2, \ldots, N$ are aware of where the movie is stored and they request it from Node 1. We are at the point where Node 1 has collected all the requests. Now, Node 1 splits the movie into P packets and then starts sending the packets using the *Client-Server* approach.

For this question, you may ignore propagation delay, but no further approximations are accepted.

a. Suppose all P packets have the same size.

   i. How long will it take for the movie to be delivered only to one node (e.g. Node 2)?

   ii. How long will it take for the movie to be delivered to *all* nodes $2, \ldots, N$?

b. Suppose now that all P packets have random sizes. What would be the best sending strategy for Node 1 to improve its delay performance: send them in a descending order (i.e. send the biggest packet first) or an ascending order (i.e. send the smallest packet first)? Does it make a difference? Justify your answer.

Since $k > 1$, link $(S_1 \to S_2)$ is the "bottleneck" link.

a. Let $f_j = \frac{F}{P} = f$ be the size of the j-th packet. All P packets have therefore the same size.

i. The total delay for transmitting P packets from node 1 to node 2 is:

$$\begin{aligned} d_{1\to 2} &= \frac{f_1}{R} + \frac{f_1}{R/k} + \left(\sum_{j=2}^{P} \frac{f_j}{R/k}\right) + \frac{f_P}{R} \\ &= \frac{f}{R} + kP \cdot \frac{f}{R} + \frac{f}{R} \\ &= (kP+2)\,\frac{F}{P \cdot R} \end{aligned}$$

Explanation: *We compute the delay into account that:*

- *link $(S_1 \to S_2)$ is the "bottleneck" link*
- *each packet j (where $2 \le j \le P$) has to queue at S_1, and wait until all preceding packets are transmitted.*
- *by the time that the last packet $(j = P)$ arrives at S_2, all the previous packets j $(1 \le j \le P-1)$ have already reached the destination node 2.*

Another way to think about this is that: since the packets have the same size and links $(1 \to S_1)$ and $(S2 \to 2)$ have the same transmission rate, the total delay would be the same if the bottleneck link was the first link that the packets had to pass through. (5 pts)

ii. The $(N-1)$ copies of the movie can be sent back-to-back to the clients to minimize the total delay. When the last packet of the first movie copy leaves node 1, node 1 can already start transmitting the first packet of the second movie copy (destined to another client). We can also observe that all links connecting S_2 and each client equal transmission rates. Thus, the problem we are trying to solve is similar to $3a.(i)$: the difference is that node 1 now has to transmit $[(N-1) \cdot P]$ packets instead of P packets.

The total delay is: (5 pts)

$$\begin{aligned} d_{CS} &= \frac{f}{R} + (N-1) \cdot P \cdot \frac{f}{R/k} + \frac{f}{R} \\ &= ((N-1) \cdot k \cdot P + 2)\,\frac{F}{P \cdot R} \end{aligned}$$

b. No, it does not make any difference. The total delay is the same in both cases (ascending and descending order) because:

- *The switch buffers are infinite and there are therefore no packet drops.*
- *The link rates of the first and the last link in each path (i.e. (server $\rightarrow$ S_1) and (S_2 $\rightarrow$ client)) are the same. Since the topology is symmetrical, the ascending and the descending order of sending packets (also symmetrical to each other) will experience the same total delay.*

(2 pts)

Question 4 (10 points):

Consider now that Node 1 uses the *P2P approach*. It, again, splits the movie into P equally sized packets and distributes the packets with the help of its peers.

Assume the following redistribution scheme: Node 1 sends packets to Node 2 only. Upon receiving a packet, Node 2 redistributes it to its subsequent neighbor (Node 3). Similarly, when the i^{th} node receives a packet it redistributes it *only* to the $(i+1)^{th}$ node. When a node has sent the last packet of the file it stops redistributing.

For this question, you may ignore propagation delay, but no further approximations are accepted.

a. How long will it take for the movie to be delivered to all the nodes?

b. Is there a value k for which the Client-Server approach has better performance than the P2P approach? Justify your answer.

a. From $3a.(i)$, we know that the time needed for the P-th packet $(j = P)$, to arrive at node 2 is $t_1 = (kP+2)\frac{f}{R}$. By that time, each of the packets before that will be either at S_2, or would have already arrived at one or more peers. In order to compute the total time needed for P packets to be pushed to all peers, we need to calculate the delay it takes for the last (P-th) packet to arrive at node N, after it has arrived at node 2. Then, we add that delay to t_1, which we have already computed, and we have our final answer.

Thus,

$$\begin{aligned} d_{P2P} &= t_1 + (N-2)\cdot 2\cdot \frac{f}{R} \\ &= (kP + 2N - 2)\,\frac{F}{P\cdot R} \end{aligned}$$

Note that term $\left(N-2)\cdot 2\cdot \frac{f}{R}\right)$ contains no delay attributed to queueing. This comes from the fact that packet P experiences no further delay due to queueing, after it reaches node 2. By the time packet P moves from node i to S_2, or from S_2 to node i, all preceding packets have already made a hop of their own as well. (e.g., when packet P arrives at S_2 coming from node 2, packet $(P-1)$ arrives at node 3; thus, there is no queue at S_2). (8 pts)

b.

$$\begin{aligned} &(N-1)kP + 2 < kP + 2N - 2 \Leftrightarrow \\ &\Leftrightarrow (N-1)kP < kP + 2N - 4 \Leftrightarrow \\ &\Leftrightarrow (N-2)kP < 2(N-2) \Leftrightarrow \\ &\Leftrightarrow k < \frac{2}{P} \end{aligned}$$

(2 pts)

4 Transport layer (30 points)

Question 1 (10 points):

End-host A communicates with end-host B using a Go-Back-N protocol with sender window size $N = 3$. The link may drop packets, but it can neither reorder nor corrupt packets. A is trying to send a file to B. It does so by splitting the file in 8 packets with sequence numbers from 0 to 7.

In the entire duration of the file transfer, end-host A receives *only* the following ACKs (and in that order): $ACK1$, $ACK1$, $ACK4$, $ACK6$, $ACK7$

Show which packets have been transmitted by A and B, including the ones which have been dropped. Answer this question by completing the sequence diagram in Figure 2.

Your answer should include the following:

- All packet transmissions, including ACKs.
- Lost packets and ACKs.
- Timeout events.
- Changes to the sender and receiver window, when packets (including ACKs) are received by either end-host.

We have already completed some of the information to help you get started. End-host A has sent data packets 0 and 1, which have reached end-host B. B sends $ACK0$ and $ACK1$, but only $ACK1$ reaches A.

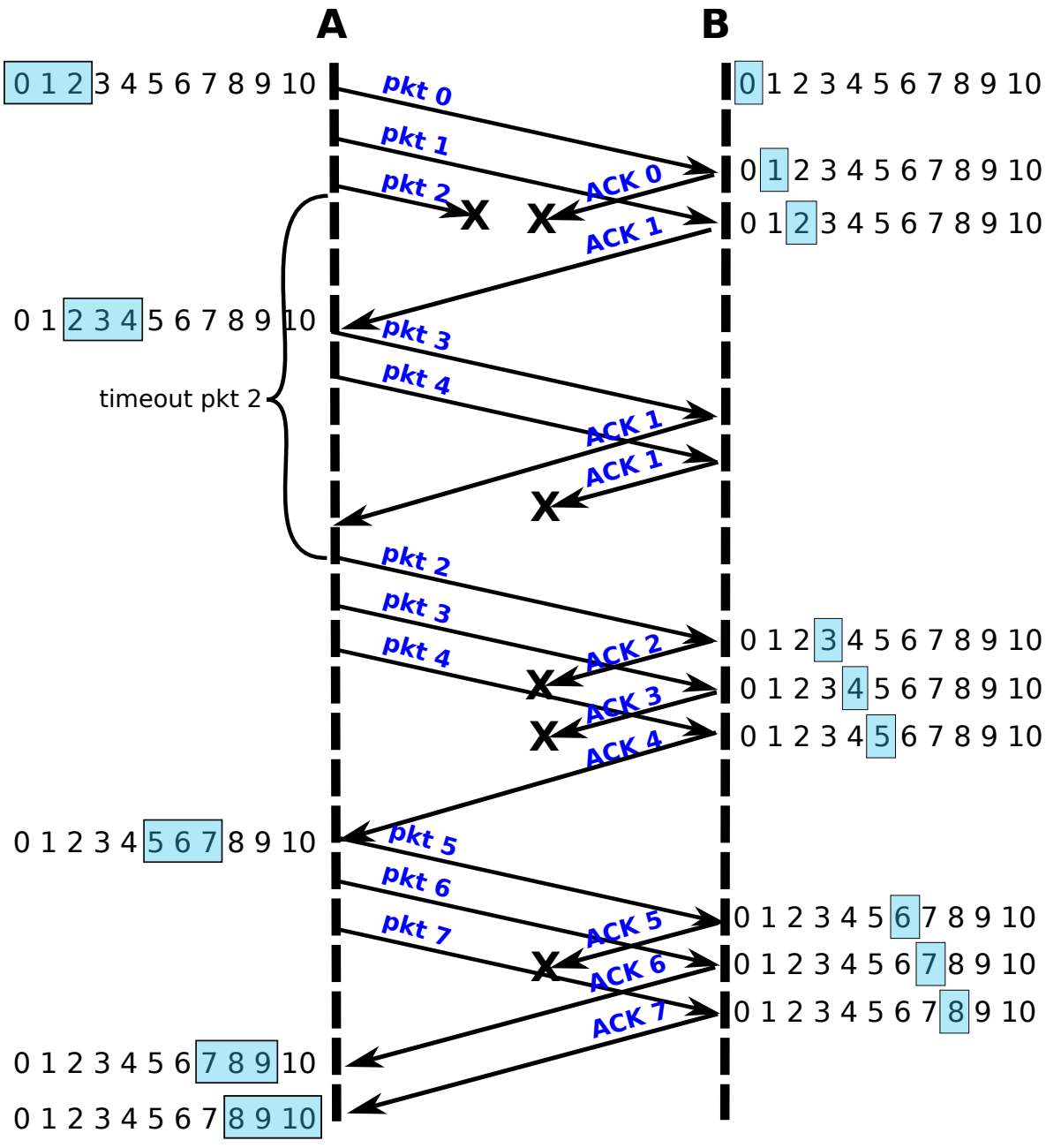

Figure 2: Sequence diagram to be completed for Question 1.

Question 2 (20 points):

Table 3 describes 5 different network channels between a sender A and a receiver B. E.g., in channel (3), there may be packet corruption only on the path from A to B, and we need to do pipelining. There is no packet reordering in any of the channels.

channel ID	errors in $A \rightarrow B$	errors in $B \rightarrow A$	support for pipelining?
(1)	none	none	yes
(2)	corruption	none	no
(3)	corruption	none	yes
(4)	corruption &loss	none	yes
(5)	corruption & loss	corruption & loss	yes

Table 3: Network channel types for Question 2.

For each channel, specify which of the mechanisms in Table 4 are necessary for providing reliable data delivery (no data loss and no data duplication).

You have the following constraints:

- Specify a mechanism only if it is necessary. E.g., if one can provide reliable data delivery over a given channel without checksums, you should not specify checksums for that channel.
- Do not specify timeout-based retransmissions if NACK-based retransmissions are sufficient.

Fill in Table 4 by writing "yes" or "no" in each cell. Then justify your choice for each channel by explaining why the mechanisms you picked are necessary and sufficient.

channel ID	sequence numbers	checksums	NACK-based retransmissions	timeout-based retransmissions
(1)				
(2)		X	X	
(3)	X	X	X	
(4)	X	X		X
(5)	X	X		X

Table 4: Reliability mechanisms to be selected for Question 2.

(10 pts)

As a general principle:

- *We need checksums to detect and drop corrupted packets. (2 pts)*

  If the channel corrupts a packet, and the protocol does not include checksums, there is no way for a receiver to know whether to accept or to discard the corrupted packet.

- *We need sequence numbers to detect and drop duplicate packets.*

  A protocol's retransmission mechanism may create duplicate packets. This is particularly the case when we use timeouts to counter packet loss in channels with unpredictable delay. We could have a scenario where the sender experiences a timeout, even though the packet correctly reaches the destination (the timeout interval was too short, or the ACK was lost).

Another use for sequence numbers is to identify exactly which packets the receiver is still missing.

This is particularly the case when there can be multiple packets in-flight (e.g., due to pipelining), and any of them could be lost. Depending on the mechanism we use, it is either the receiver that explicitly specifies the missing packets (NACKs), or it is the sender that infers the missing packets (the packets it has not received ACKs, when a timeout occurs). (3 pts)

(1) No errors occur in this channel; there is never any need to retransmit packets.

(2) NACK-based retransmissions are sufficient for this channel: the receiver can explicitly notify the sender about whether a packet is corrupted or not, since we know that all NACKs will reach the sender (no corruption or loss on the $B \rightarrow A$ direction). (2 pts) Since there can ever be only one packet in-flight at any given time, we do not require sequence numbers. When the sender receives a NACK at any point in time, it knows that the NACK corresponds the last packet that was transmitted.

(3) This is similar to the previous channel, with the difference that we now require sequence numbers. Since any of the transmitted packets could have been dropped, the receiver will have to specify which of the packets it is still missing.

(4) A NACK-based retransmission mechanism will not be sufficient for this channel. If the network drops a packet, the receiver does not have enough information to discover that a packet has been dropped and notify the sender. (2 pts)

We could argue that the receiver can discover that a packet has been lost, when it receives a packet with a higher sequence number. Such a mechanism would be sufficient for applications which constantly generate new data (e.g., video streaming applications).

However, we cannot rely on this mechanism to counter packet loss when it is the last packet during a transfer that has been lost. For that final packet, we still require a timeout-based retransmission mechanism.

Since we could also use a timeout-based retransmission mechanism to recover from all other errors in the channel, we could switch to only using a timeout-based retransmission mechanism. In this case, a NACK-based retransmission is no longer needed. (1 pt)

We need sequence numbers to detect duplicate packets; since delays are unpredictable, our timeout-based retransmission mechanism may unnecessarily retransmit certain packets (e.g., if the timeout is too short).

(5) A timeout-based retransmission mechanism is sufficient to also recover from errors on the $B \rightarrow A$ direction. The reasoning is identical to the one we used for the previous channel.