Problem Set #2

CS 352 Internet Technology

Released: October 27, 2022; Due: November 14, 2022 at 8 pm Eastern Time

Instructions

Please read and follow these instructions carefully.

1. You must work on this problem set individually.
2. Please complete your answers to the questions below and upload your responses to Canvas as a single PDF file.
3. Your answers (preferably typed up rather than handwritten) must be clear, legible, and concise. If we cannot understand your answer with reasonable effort, you will get zero credit.
4. If you leave a question out or clearly state "I don't know", you will receive 25% of the points for that question.
5. We care not just about your final answer, but also how you approach the questions. In general, if you show us your reasoning for your answers, we will do our best to provide partial credit even if your final answer is incorrect. However, if you provide no reasoning, and your answer turns out to be incorrect, you will typically receive zero points. So, please explain yourself.
6. You are free to discuss the problem set on Piazza or through other means with your peers and the instructors. You may refer to the course materials, textbook, and resources on the Internet for a deeper understanding of topics. However, you cannot lift solutions from other students or from the web. Do not post these problems to question-answering services like Chegg. All written solutions must be your own. We run sophisticated software to detect plagiarism and carefully monitor student answers.
7. There is a due date/time but no "time limit" on problem sets. That is, you may take as long as you need to work on problem sets, as long as you submit them on time. Further, you may make an unlimited number of submissions on Canvas.
8. As a response to the last question of this problem set, please specify who you collaborated with, and also include all the resources you consulted to answer questions in this problem set, including URLs of pages you visited on the Internet. Also specify which question and aspect you got help with. Please be as thorough and complete as possible here. It is mandatory to answer this question.
9. If you have any questions or clarifications on the problem set, please post them on Piazza or contact the course staff. We are here to help.

Questions

There are 13 questions in this problem set totalling to 50 points.

(1) TCP vs. UDP (2 points). When is it beneficial for an application to use TCP as its transport, and when UDP?

TCP is connection-oriented. So if we want applications where reliability is required, like email, and http, then we should use TCP. Because TCP will provide reliable data delivery by using RTO, ACKs, and sequence number.

The benefit part of UDP is that UDP is best-effort service and connectionless. In this case, it is easy to build and low overhead. Suitable for one-off request/response, time-sensitive and loss tolerant situations

(2) Computing the checksum (2 + 2 = 4 points). In this problem, we will get practice with computing the one's complement of the one's complement sum over a sequence of data values. We will use the exact same algorithm used to compute TCP/UDP checksum, but over a sequence of 4-bit chunks rather than 16-bit chunks.

(a) Consider the following data bits. What is the (4-bit) checksum of this (8-bit) data? (2 points)

```
 0101 1100

 0101
 1100
10001
 0010
And then we do one's complement
 1101
```

(b) Consider the following data bits. What is the 4-bit checksum of this (12-bit) data? (Note: you may be able to reuse calculations from part (a).) (2 points)

```
0101 1100 1001
```

```
We know the sum of 0101 and 1100 is 0010

So
0010
1001

1011
And then we do a one’s complement
0100
```

(3) Listing and identifying sockets. (5 points) As we saw in a demo in class, the `ss` command may be used to display the listening connections and the established connections on a machine. Suppose you see the sample output shown below.

```
Netid  State    Local Addr:Port    Peer Addr:Port    Users
udp    UNCONN   0.0.0.0:8003       0.0.0.0:*         (pid=29604,fd=3)
tcp    LISTEN   0.0.0.0:8003       0.0.0.0:*         (pid=28832,fd=3)

Netid  State    Local Addr:Port    Peer Addr:Port    Users
tcp    ESTAB    127.0.0.1:8003     127.0.0.1:47468   (pid=28832,fd=4)
tcp    ESTAB    127.0.0.1:47468    127.0.0.1:8003    (pid=29910,fd=3)
```

The output above lists all the connections on the machine where the `ss` command is run. Helpfully, the last column of the output, entitled *Users*, also shows the process ID (`pid`) of the application process associated with the socket and the number of the file descriptor (`fd`) associated with the socket.

(a) Suppose a TCP packet enters the machine, destined to TCP port 8003 corresponding to an established connection. Can you identify the `pid` and socket `fd`, if any, corresponding to the socket where this packet is demuxed to? (1 point)

pid=28832, fd=4

(b) Suppose a TCP packet enters the machine, destined to TCP port 8003, corresponding to a fresh connection just being initiated by a client through a `connect()` call. Can you identify the `pid` and socket `fd`, if any, corresponding to the socket where this packet is demuxed to? (1 point)

pid=28832, fd=3

(c) Suppose a TCP packet enters the machine, destined to TCP port 47468, corresponding to a fresh connection being made by a client through the `connect()` call. Will this client's `connect()` succeed? Why or why not? (2 points)

No, because there is no listen socket connections at port 47468

(d) Suppose a UDP packet enters the machine, destined to UDP port 8003. Can you identify the `pid` and socket `fd`, if any, corresponding to the socket where this packet is demuxed to? (1 point)

pid=29604, fd=3

(4) When are retransmission timeouts useful? (0.5 * 4 = 2 points) Say YES or NO to each part of the question below. Do retransmission timeouts (followed by retransmission) help a sender ensure that a receiver was delivered a piece of data when:

(a) the data was lost?

yes

(b) the data was corrupted?

yes

(c) the ACK was lost?
yes

(d) the ACK was delayed?

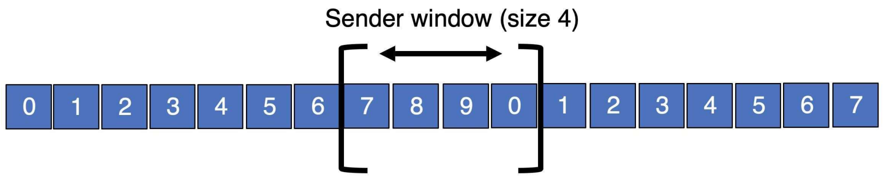
yes

(5) Efficiency gains with pipelined reliability protocols (2 points). Why do pipelined reliability protocols provide more throughput than stop-and-wait reliability protocols?

For stop-and-wait reliability, Sender sends a single packet, then waits for an ACK to know the packet was successfully received. Then the sender transmits the next packet. In this case, Sender sends only one packet per RTT, which is not efficient, because it makes the data transfer rate limited by the time between the endpoints, rather than the bandwidth.
For pipelined reliability, we keep many packets in flight. In this case, new packets are sent at the same time as older ones are still in flight. New packets are sent at the same time as ACKs are returning. Thus, pipelined reliability improves throughput by moving more data in the same time

(6) Sender and receiver view in sliding window protocols (8 points). Suppose a sender and receiver agree to use a window size of 4. Further, assume that the set of available sequence numbers goes from 0 to 9, and then rolls back to 0.

The sender's view of the current window of in-flight sequence numbers is shown below.



Assume there are no packet losses in the current window.

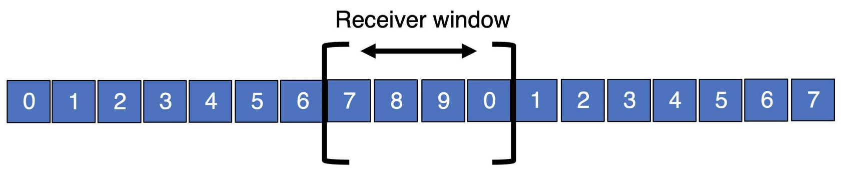
(a) What was the latest sequence number for which the sender has received an ACK from the receiver? (1 point)

6

(b) What is the latest sequence number that the sender has transmitted? (1 point)

0

For parts (c) and (d) below, consider the receiver's view of the current window of in-flight sequence numbers.



(c) What was the latest sequence number whose ACK was sent by the receiver? (1 point)

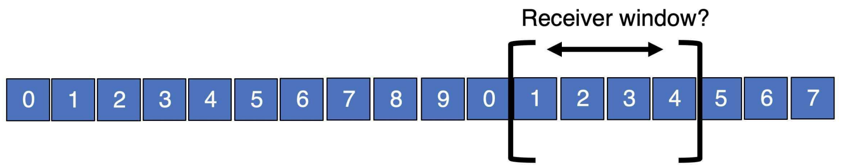
6

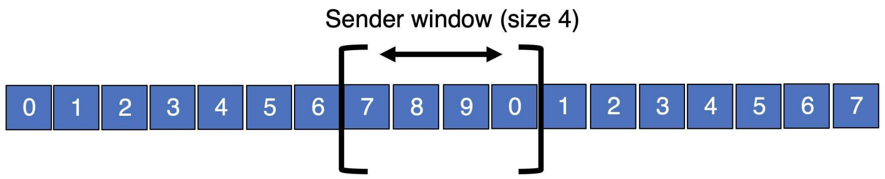
(d) What is the latest sequence number that will be accepted to be buffered by the receiver? (1 point)

0

For parts (e) and (f), assume the sender's view of in-flight sequence numbers is the same as the one shown at the beginning of this question.

(e) Is it possible that the following view of the window *at the receiver* can happen simultaneously with the sender's view shown above? Why or why not? (1 + 1 = 2 points)

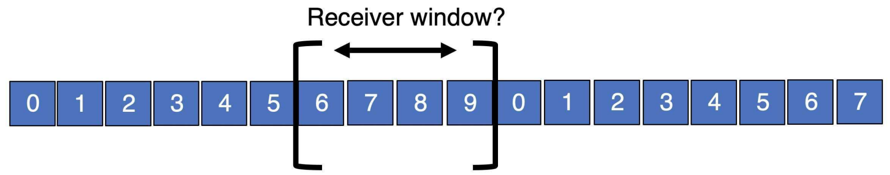


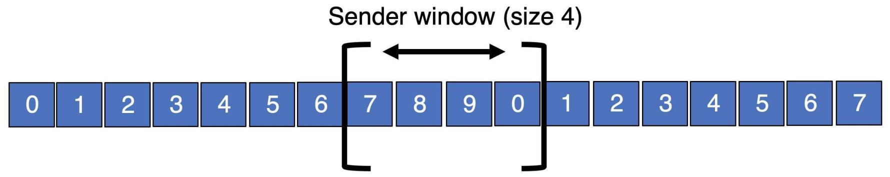


It is possible when the sender's window is 7890, the receiver has ACK all of them, but ACKs are still in flight. So the receiver has slid the window forward by 4, and the sender has not.

During this process, it will result in the picture above.

(f) Is it possible that the following view of the window at the receiver can happen simultaneously with the sender's view shown above? Why or why not? (1 + 1 = 2 points)





No, it is not possible. The receiver's window can be more advanced compared to the sender's but not behind. Because the sender should only slide forward when he receives an ACK from the receiver, and at that time the receiver should already slide forward since the receiver already sends an ACK to the sender. So the receiver should not be behind the sender.
In this question, the receiver window is behind, so it is not possible.

(7) TCP byte-based sequence numbers (8 points). Consider the TCP segment structure shown in slide 15/16 of lecture 13.

Although most of the discussion in class assigned sequence numbers to each *packet*, TCP uses a distinct sequence number for each *byte* from the application. The sequence number header on the TCP packet merely represents the sequence number of the *first byte* of application data contained in the packet.

Suppose the sequence number field on a TCP packet is 46005. The packet carries 102 bytes of (application-layer) data. The TCP header size on this packet is 20 bytes.

(a) What is the sequence number of the *last* application byte contained in the packet above? (2 points)

46005 (first byte) + (102 – 1) = 46106
The reason we need 102-1 is because we want the last application byte.
So the sequence number of the last application byte contained in packet is 46106.

(b) What is the value of the *acknowledgment number* field on the ACK that the receiver generates for the packet above? (2 points)

46005 + 102 = 46107
So the value of acknowledgment number field on the ACK is 46107

(c) Do you have sufficient information in this question to determine the value of the *sequence number* field of the ACK packet that the receiver generates for the packet above? If so, what is the sequence number? If not, why not? (2 points)

We don't have enough information, because TCP transmission is bi-directional, so the connection between A to B and B to A are independent. We don't know the ACK value (expected data for the sender from the receiver) in the sender's packet, so we don't know the sequence number in the ACK packet. So we do not have enough information to determine the value of the sequence number field.

(d) If the size of the TCP header were larger (e.g., 24 bytes, due to additional TCP options), would your answers to parts (a)–(c) change? Why or why not? (2 points)

For a and b, the answer will not change, because application data are still same.
For c, they will not change, because the size of TCP header have nothing to do with the sequence number. And since TCP transmission is bi-directional, the size of header will also have nothing to do with c.

(8) TCP retransmission strategies (5 points).

(a) What is the advantage of selective repeat over a go-back-N retransmission strategy? What is a disadvantage? (2 points)

The advantages of a selective repeat are receiver can store all the correct frames that arrive following the bad one, while go-back-N will simply discards all data with greater sequence numbers. Selective repeat also reduces the number of duplicate transmissions.
The disadvantage of selective repeat the packet header space is large compared to go-back-N. Also, it is complex and hard to build. And maybe the data will transmit out of order

(b) Among the selective repeat strategies, what is an advantage of selective acknowledgments over cumulative acknowledgments? What is a disadvantage? (3 points)

The advantage of selective acknowledgments is that it has more precision of information of the missing data, so it significantly reduce the number of duplicate transmissions compared to cumulative ACK.
The disadvantage is that since the information of the missing data is more precise, it require more packet header space. And it is also more complex and harder to build.

(9) Which data will be retransmitted? (1 + 2.5 + 2.5 = 6 points)

Suppose a sender transmits data with sequence numbers 500 through 510 back-to-back to a receiver (inclusive). The network drops data with sequence numbers 503, 506, and 508, while the rest of the data reaches the receiver's TCP stack. Suppose the round-trip time (RTT) of the connection is T (fixed), which includes negligible transmission and queueing delays. The sender has no more data to transmit beyond sequence number 510.

At time $t = 0$, the retransmission timeout (RTO) corresponding to sequence number 503 elapses at the sender. (In general, the RTO value is configured to be larger than the measured RTT of the connection.)

(a) Suppose the sender and receiver use a go-back-N retransmission strategy. Which sequence numbers are retransmitted by the sender after the RTO is triggered?

503

(b) Instead, suppose the sender and receiver use a cumulative ACK retransmission strategy. Assume that the sender receives (correspondingly-numbered) ACKs from the receiver for each sequence number successfully delivered. Further, assume that after $t = 0$, there are no drops in the network. At what time does the sender know that it has successfully recovered from all the losses? Explain your answer.

No drops after time t means that all the ACKs will be delivered and all the following retransmission will success. In this case, we only need to re-send 503, 506, and 508. Since we are using cumulative ACK, so we can only send each dropped data at a time. Therefore, we need 3RTO, so 3T.

(c) Instead, suppose the sender and receiver use a selective ACK retransmission strategy. Assume that the sender receives (correspondingly-numbered) ACKs from the receiver for each se- quence number successfully delivered. Further, assume that after $t = 0$, there are no drops in the network. Note that the sender may use the information in all the selective ACKs so far to retransmit more eagerly, after an RTO. At what time does the sender know it has successfully recovered from all the losses? Explain your answer.

We know that selective ACK will send all the missing data information to the sender in one-shot, which means that sender will know all the missing data at once, and re-send them all. Therefore we only need 1 RTO, which is T. The SACK will be 500-502, 504-505, 507, 509-510. So sender knows 503, 506, 508 are missing.

(10) The impact of packet reordering (2 points). Why might heavy packet reordering in the network reduce the throughput of TCP connections?

From the application's view, less data will be delivered from transport layer to application layer, because data need to be stored and ordered before passing to the application. Therefore, heavy packet reorder may cause the buffer full. Once the buffer is full or close to full, then the sender will stop sending data or slowly sending data. In this case, it will reduce the throughput of TCP connections.

(11) Stream-oriented transfer (2 points). Suppose a TCP sender pushed two pieces of data through a socket `send()` call. Each piece of data is of size 100 bytes. The data is delivered reliably to the TCP receiver. Now the receiving application performs a `recv()` system call on its socket, and gets some data in return. What are the possible sizes (in bytes) of this returned data?

Value between 1 – 200 bytes, depends on the recv buffer size and the number of in-order data.

(12) Flow control and congestion control (4 points).

(a) Why is flow control necessary in TCP? (2 points)

Because we want to avoid overwhelming the receiving application, we don't want any data drops due to buffer fill. Receivers can only buffer so much before dropping subsequent data. So if we don't have flow control, then we will have data drops after the buffer fills. So if we have flow control, then we can change our window depending on advertised window.

(b) What is the difference between flow control and congestion control? (2 points)

The flow control is to avoid overwhelming the receiving application. Sender is managing the receiver's socket buffer.
The congestion control is to avoid overwhelming the bottleneck network link. Sender is managing the bottleneck link capacity and bottleneck router buffers.

(13) Collaboration and References (mandatory). Who did you collaborate with on this problem set? What resources and references did you consult? Please also specify on what questions and aspects of the problem set you got help on. If you did not consult any resources other than the lecture slides and textbook, just say "no collaboration".

TA, recitation, lecture slides, and my partner JiaJing, we kind of discuss through every questions.