CSE3300/CSE5299: Computer Networking

Homework 3

Due Date: **10/01/2025, Wednesday**. Submission through HuskyCT.
Full score: 100 for CSE3300 students; 120 for CSE5299 students (will be normalized to 100 when entering the grade in HuskyCT).

1. **Checksum (30 points).**

   A. Suppose you have the following 2 bytes: 01011100 and 01100101. What is the 1s complement of the sum of these 2 bytes? (5 pts)

   Sum = 11000001

   1's Complement = 00111110

   B. Suppose you have the following 2 bytes: 11011010 and 01100101. What is the 1s complement of the sum of these 2 bytes? (5 pts)

   Sum (Warp around and carryback) = 01000000

   1's Complement = 10111111

   C. For the bytes in part (a), give an example where one bit is flipped in each of the 2 bytes and yet the 1s complement doesn't change. (5 pts)
   bit number counting from R to L
   - Flip bit 5 in both:

     $01011100 \rightarrow 01111100, \quad 01100101 \rightarrow 01000101$

   - Flip bit 4 in both:

     $01011100 \rightarrow 01001100, \quad 01100101 \rightarrow 01110101$

   - Flip bit 3 in both:

     $01011100 \rightarrow 01010100, \quad 01100101 \rightarrow 01101101$

   - Flip bit 0 in both:

     $01011100 \rightarrow 01011101, \quad 01100101 \rightarrow 01100100$

D. UDP and TCP use 1s complement for their checksums. Suppose you have the following bitstream: `0101001101100110011110100`. What is the checksum? (It is the 1s complement of the sum of this bitstream. Note that although UDP and TCP use 16-bit words in computing the checksum, for this problem you are being asked to consider 8-bit sums.) Show all work. (15 pts)

Sum = 01010011 + 01100110 + 01110100 = 100101101 (9 bits)

Wrap-around and carry back: 00101101 + 1 = 00101110 (8 bits)

1's complement = 11010001

2. **Go-Back-N and Selective Repeat (20 points).** Consider Fig. 1 that shows the current sender window status at the sender at time t. For each of the following statements, decide whether it is correct or wrong for time t. Briefly justify your answer.

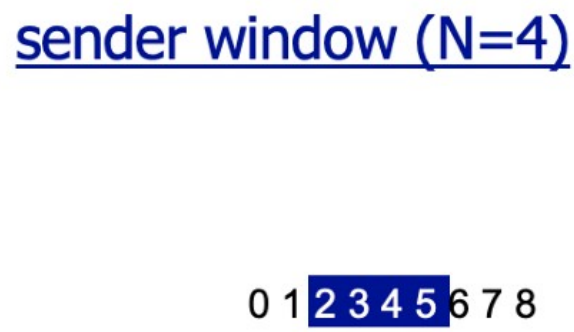


Figure 1: Illustration of the sender window at time t.

A. If Go-Back-N is used, the sender can receive an ACK for pkt 0. (4 pts)
Wrong/Correct. Pkt already acked, sender can't receive it at time t. A rare case is that an ACK0 was sent by the receiver to the sender, but was delayed for a long time, and then comes back after the sender window is as shown in Fig. 1. This is a case where the sender receives ACK0. The grading is flexible for this problem. If a student answered Yes with a well justified scenario, that's fine.

B. If Go-Back-N is used, the sender can receive an ACK for pkt 2. (4 pts)
Correct. Pkt is in the sender window.

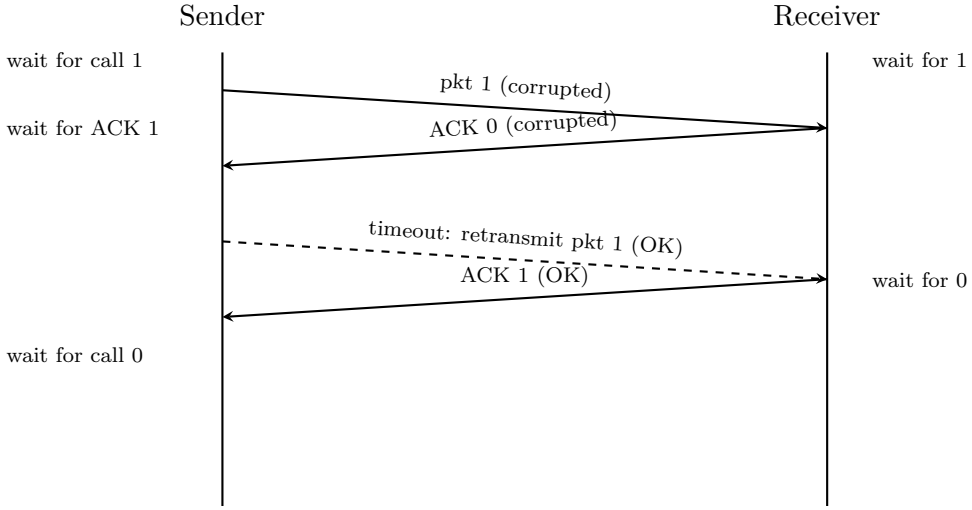
C. If Selective Repeat is used, the sender can receive an ACK for pkt 0. (4 pts)
Wrong/Correct. Pkt already acked, sender can't receive it at time t. For similar reason as for question A, the grading is flexible. If a student answered Yes with a well justified scenario, that's fine.

D. If Selective Repeat is used, the sender can receive an ACK for pkt 2. (4 pts)
Correct. Pkt is in the sender window.

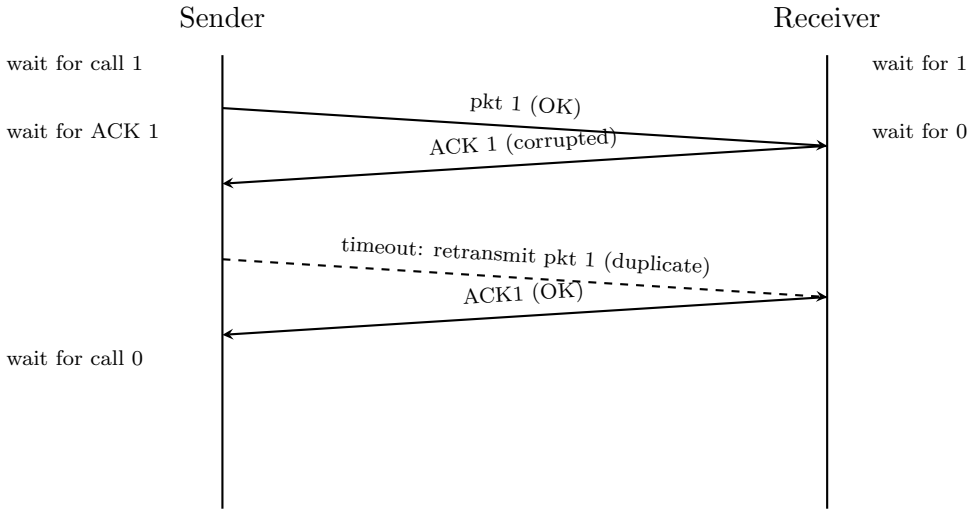
E. If Go-Back-N is used, the next ACK that the sender receives can be ACK 3. (4 pts)
Correct. For the scenario that the receiver receives pkt 2 then pkt 3, but ack 2 is lost later, then next ack the sender receives can be ack 3.

3. **rdt3.0 (30 points).** Consider the FSM of rdt3.0 shown in Fig. 2. Recall that rdt3.0 is designed for a channel that can corrupt and lose packets. Note that the receiver of rdt3.0 is the same as the receiver for rdt 2.2, which does not use NAK, rather only uses ACK with sequence numbers.

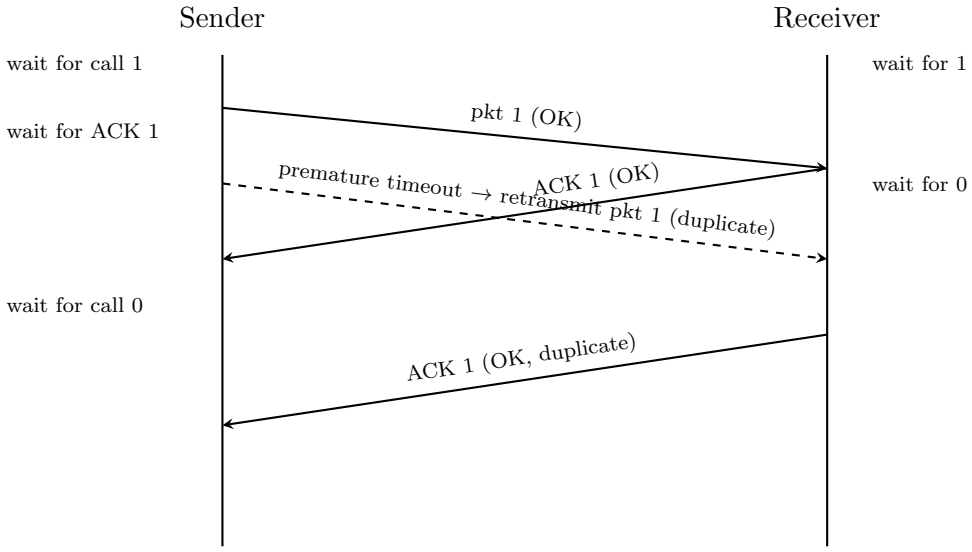
   a. (10 points) Suppose the sender is in the state of "wait for call 1 from above" and sends pkt1 to the receiver when receiving a packet from above (i.e., from the application layer). Suppose pkt1 is corrupted, but not lost. The receiver is in the state of "wait for 1 from below" and sends a "ACK, 0" after receiving pkt1. Suppose that the ACK is corrupted but not lost by the channel. Draw a time sequence graph to show the actions and state evolution at the sender and the receiver.



   b. (10 points) Suppose the sender is in the state of "wait for call 1 from above" and sends pkt1 to the receiver when receiving a packet from above (i.e., from the application layer). Suppose pkt1 is not lost or corrupted. The receiver is in the state of "wait for 1 from below" and sends an "ACK, 1" after receiving pkt1. Suppose that the ACK is corrupted by the channel. Draw a time sequence graph to show the actions and state evolution at the sender and the receiver.



c. (10 points) Consider the scenario in [b]. The receiver sends an "ACK, 1" after receiving pkt1. Suppose that the ACK is not corrupted or lost by the channel. But the sender has a premature timeout, i.e., it resends pkt1 before receiving "ACK, 1". Suppose none of the packets is lost or corrupted, and there is no premature timeout afterwards. Draw a time sequence graph to show the actions and state evolution at the sender and the receiver, in particular show how the server treats the two "ACK, 1" (one received after resending pkt1, and the other is for the retransmission of pkt1), and what the receiver will do after receiving the duplicate pkt1.



4. **Socket programming (20 points).** The goal of this problem is to help you get familiar with socket programming.

   - **Set up environment.** You will need to have an environment that you can run python. Ideally, you want to run python3.
   - **Test sample code (the simple version). (10 points)** You can first test a simple version of echo-server and echo-client (attached to this homework) that does not have much in handling errors. Run the server in one terminal. Start a client in another terminal. After that, start another client. (1) Use screenshots to show what you have done and describe the clients' port numbers. (2) Modify the source code so that the client requires entering a phrase from the keyboard. Again use screenshots to show what you have done (include the source code of the client that you have modified).

     Answer may vary, give credit if the results are reasonable.
   - **Test sample code (the better version). (10 points)** Test echo-server-better and echo-client-better (attached to this homework) that include code in handling errors and has more elaborate code. The comment in the code suggests using python3, but you can run the code using python. Run the server in one terminal. Start a client in another terminal. (1) Run the client as `python echo-client-better.py`. Why is it not working? Please be very specific in explaining what is wrong. (2) Run the client in another way to make it talk to the server successfully. Describe the command line to run the client properly.

Wrong port number.
python3 echo-client-better.py 127.0.0.1 50008
first argument: IP address (we use loop back address for Localhost)
second argument: port number (should match the server port)

5. **(For CSE5299 students only) (20 points) DNS Privacy.** Read the article "User Expectations and Understanding of Encrypted DNS Settings" that is posted in HuskyCT with this homework. Answer the following questions. Feel free to look up other relevant information from the Internet.

   - (5 points) What are the main purposes of DNS-Over-TLS (DoT) and DNS-Over-HTTPS (DoH)? What benefits do they provide over the typical DNS over UDP port 53 (also referred to as *Do53*)? Provide a concrete example to show the benefits.

     Main purpose: prevent eavesdroppers from observing DNS queries.
     Benefits:

     – Privacy/Censorship resistance: DNS queries are encrypted.
     – Authenticity: TLS prevents message tampering.

     In an untrusted network, e.g., guest Wi-Fi at McDonald's, an attacker on the same network will not be able to learn which site we are visiting and therefore cannot perform related attacks.

   - (5 points) DoT and DoH solve an entirely different problem from DNSSEC. Explain why.

     DNS query encryption (mainly targets eavesdroppers) vs. DNS response authentication (ensures responses from resolvers are authentic).

   - (5 points) How can Oblivious DNS (ODNS) and Oblivious DNS over HTTPS (ODoH) provide additional advantages over DoT and DoH?

     By hiding query from recursive resolver and original ip from Oblivious DNS server

   - (5 points) What browser do you use as the default browser? For your browser, check whether you can find the setting on DNS-Over-HTTPS (DoH). The article at `https://www.zdnet.com/article/dns-over-https-doh-support-added-to-chrome-on-android/` might be useful. Note that some browsers refer to this as *Secure DNS*; *Private DNS* would be more appropriate, since this is more for privacy than security. If you use an Android or iPhone, check whether private DNS is enabled or not. You only need to do one of the above (i.e., either check your browser or phone).

     Answer may vary.
     e.g. Microsoft Edge
     Privacy, search, and services → Security
     Use Secure DNS (on by default)

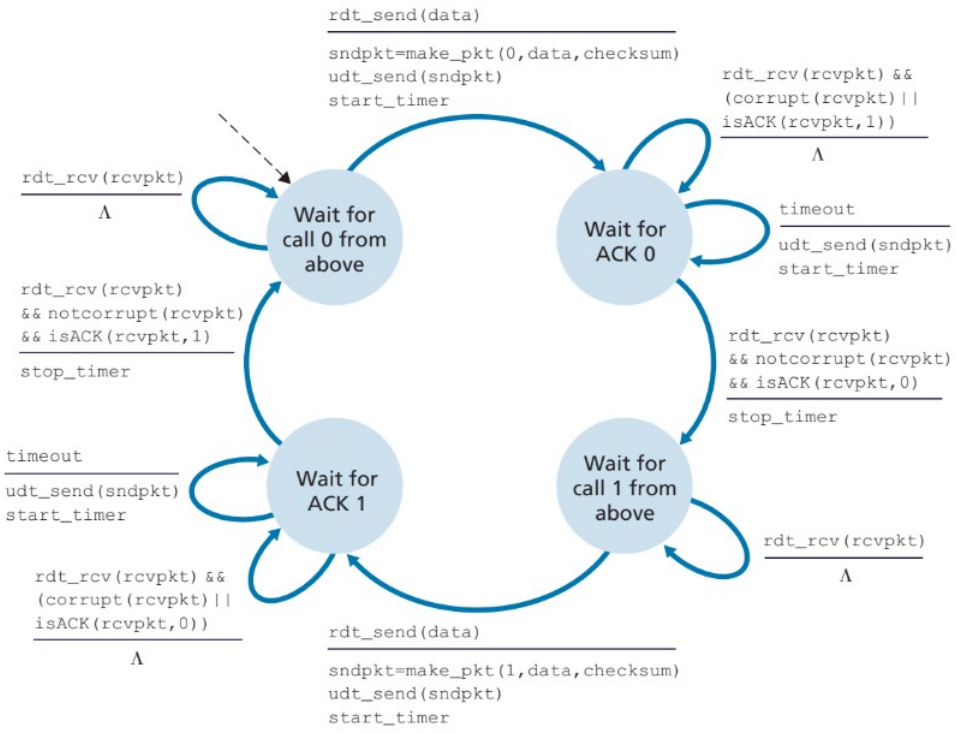


Figure 3.15 ♦ rdt3.0 sender

(a) rdt 3.0 sender from the textbook

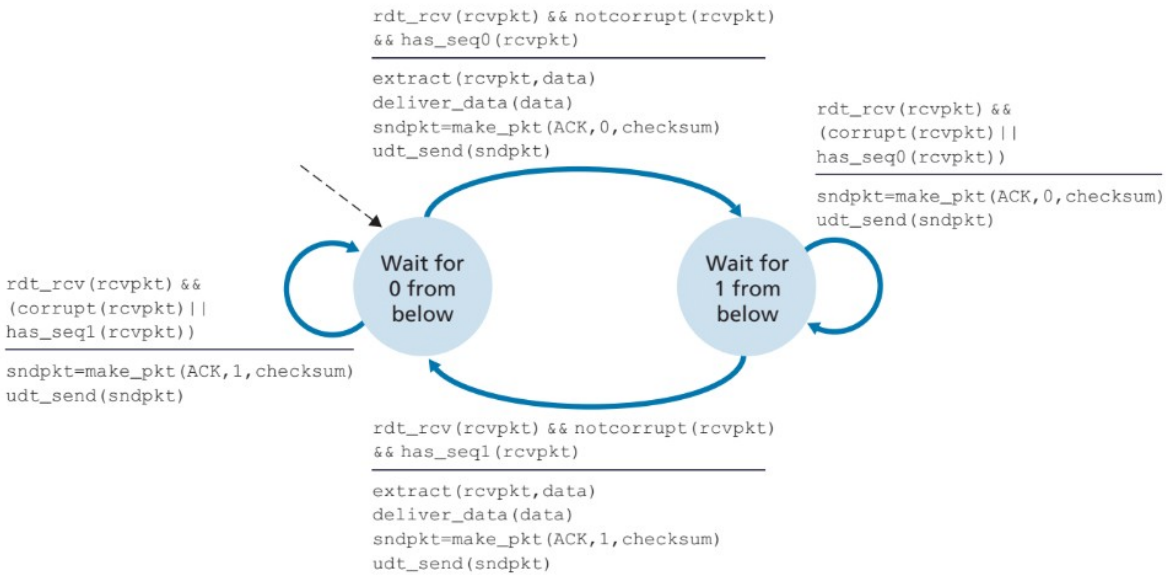


Figure 3.14 ♦ rdt2.2 receiver

(b) rdt 2.2/3.0 receiver from the textbook

Figure 2: rdt 3.0 protocol diagrams