COMP 535 - COMPUTER NETWORKS, Winter 2025
ASSIGNMENT 1 (due date: Feb. 10th, 2025) HARD deadline
Instructor: Dr. Elsaadawy

Notes:

- Submit your solutions in **"WA1.pdf"** to **WA1** submission folder on MyCourses by **Feb. 10, 2025**. No submission will be accepted by email.
- You **must** include your name and ID number on the **1st page** of your homework.
- Corresponding TA: mohammad.ghadaksaz@mail.mcgill.ca

Part A: Written Assignment

[10] **Problem 1:** Suppose a 1-Gbps point-to-point link is being set up between the Earth and a new lunar colony. The distance from the moon to the Earth is approximately 385,000 km, and data travels over the link at the speed of light - 3×10^8 m/s.

[3] (a) Calculate the minimum RTT for the link.

[2] (b) Using the RTT as the delay, calculate the delay × bandwidth product for the link.

[2] (c) What is the significance of the delay × bandwidth product computed in (b)?

[3] (d) A camera on the lunar base takes pictures of the Earth and saves them in digital format to disk. Suppose Mission Control on Earth wishes to download the most current image, which is 25 MB. What is the minimum amount of time that will elapse between when the request for the data goes out and the transfer is finished?

[2] **Problem 2:** Show the 4B/5B encoding, and the resulting NRZI signal, for the following bit sequence:

1110 0101 0000 0011

[10] **Problem 3:** Suppose we want to transmit the message 010110110110 and protect it from errors using the CRC polynomial $x^4 + x^2 + x + 1$.

[5] **(a)** Use polynomial long division to determine the message that should be transmitted.

[5] **(b)** Suppose 101010 is the error that occurs in the transmission of the code word to the remote host due to noise on the transmission link. What is the result of the receiver's CRC calculation? How does the receiver know that an error has occurred?

[10] **Problem 4:** Suppose that if a host detects a transmission while it is transmitting a frame, then: (i) if the host has already transmitted the 64 bit preamble, the host stops transmitting the frame and sends a 32 bit jamming sequence; (ii) else the host finishes transmitting the 64 bit preamble and then sends a 32 bit jamming sequence. For simplicity, assume a collision is detected as soon as an interfering signal first begins to reach a host. Suppose the packets are 512 bits long, which is the minimum length allowed. Hosts A and B are the only active hosts on a 10 Mbps Ethernet and the propagation time between them is 15 μS, or 150 bit durations. Suppose A begins transmitting a frame at time t = 0, and just before the frame reaches B, B begins sending a frame, and then almost immediately B detects a collision.

[5] **(a)** Does A finish transmitting the frame before it detects that there was a collision? Explain.

[5] **(b)**What time does A finish sending a jamming signal? What time does B finish sending a jamming signal

[5] **Problem 5:** Consider a selective repeat Protocol with a sender window size of 4 and a sequence number range of 32. Suppose that at time t, the next in-order packet that the receiver is expecting has a sequence number of k. Assume that the medium does not reorder messages. What are the possible sets of sequence numbers inside the sender's window at time t? Justify your answer.

[5] **Problem 6:** Suppose you are designing a sliding window protocol for a 1-Mbps point-to-point link to the moon, which has a one-way latency of 1.25 seconds. Assuming that each frame carries 1 KB of data, what is the minimum number of bits you need for the sequence number?

[5] **Problem 7:** Suppose a router has built up the routing table shown in the following table. The router can deliver packets directly over interfaces 0 and 1, or it can forward packets to routers R2, R3, or R4. Describe what the router does with a packet addressed to each of the following destinations:

(a) 128.96.39.10

(b) 128.96.40.12

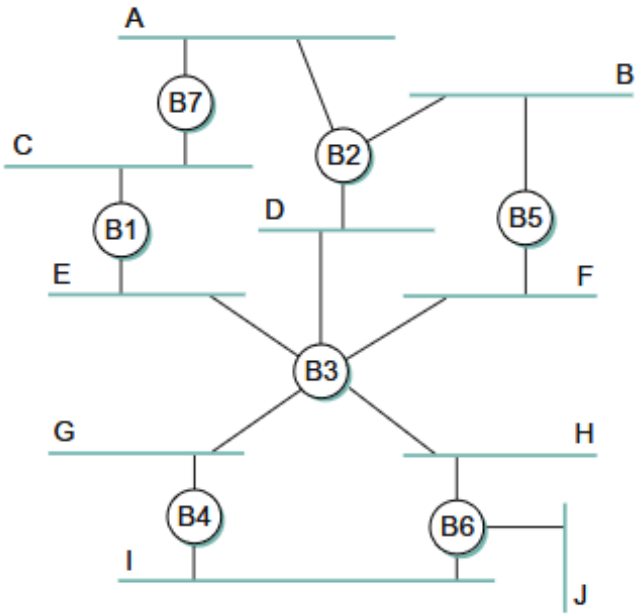
(c) 128.96.40.151

(d) 192.4.153.17

(e) 192.4.153.90

SubnetNumber	SubnetMask	NextHop
128.96.39.0	255.255.255.128	Interface 0
128.96.39.128	255.255.255.128	Interface 1
128.96.40.0	255.255.255.128	R2
192.4.153.0	255.255.255.192	R3
default		R4

[10] **Problem 8:** Given the extended LAN shown in the figure below, indicate which ports are not selected by the spanning tree algorithm, and discuss how any ties are resolved.



[10] **Problem 9:** Suppose a TCP message that contains 1024 bytes of data and 20 bytes of TCP header is passed to IP for delivery across two networks interconnected by a router (i.e., it travels from the source host to a router to the destination host). The first network has an MTU of 1024 bytes; the second has an MTU of 576 bytes. Each network's MTU gives the size of the largest IP datagram that can be carried in a link-layer frame. Give the sizes and offsets of the sequence of fragments delivered to the network layer at the destination host. Assume all IP headers are 20 bytes.

Part B: Wireshark Exercises

In this lab, we'll investigate the celebrated IP protocol, focusing on the IPv4 datagram. This lab has two parts. In the first part, we'll analyze packets in a trace of IPv4 datagrams sent and received by the `traceroute` program (explained in class). We'll study IP fragmentation in Part 2 of this lab.

Prerequisites

For this lab, we'll use the Wireshark[1] packet sniffer. Wireshark is a free/shareware packet sniffer that runs on Windows, Linux/Unix, and Mac computers. Wireshark installation instructions can be found at `https://www.wireshark.org/download.html`, and documentation on how to use Wireshark to capture network data can be found at `https://www.wireshark.org/docs/wsug_html_chunked/ChCapCapturingSection.html`.

Capturing packets from an execution of `traceroute`

In order to generate a trace of IPv4 datagrams for the two parts of this lab, we'll use the `traceroute` program to send datagrams of two different sizes to www.mcgill.ca. Recall that traceroute operates by first sending one or more datagrams with the time-to-live (TTL) field in the IP header set to 1; it then sends a series of one or more datagrams towards the same destination with a TTL value of 2; it then sends a series of datagrams towards the same destination with a TTL value of 3; and so on. Recall that a router must decrement the TTL in each received datagram by 1 (actually, RFC 791 says that the router must decrement the TTL by at least one). If the TTL reaches 0, the router returns an ICMP message (type 11 – TTL-exceeded) to the sending host. As a result of this behavior, a datagram with a TTL of 1 (sent by the host executing `traceroute`) will cause the router one hop away from the sender to send an ICMP TTL-exceeded message back to the sender; the datagram sent with a TTL of 2 will cause the router two hops away to send an ICMP message back to the sender; the datagram sent with a TTL of 3 will cause the router three hops away to send an ICMP message back to the sender; and so on. In this manner, the host executing `traceroute` can learn the IP addresses of the routers between itself and the destination by looking at the source IP addresses in the datagrams containing the ICMP TTL-exceeded messages.

Let's run `traceroute` and have it send datagrams of two different sizes. The larger of the two datagram lengths will require `traceroute` messages to be fragmented across multiple IPv4 datagrams.

- **Linux/MacOS:** With the Linux/MacOS traceroute command, the size of the UDP datagram sent towards the final destination can be explicitly set by indicating the number of bytes in the datagram; this value is entered in the `traceroute` command line immediately after the name or address of the destination. For example, to send `traceroute` datagrams of 2000 bytes towards www.mcgill.ca, the command would be:

  %`traceroute` www.mcgill.ca `2000`

[1] Wireshark: `https://www.wireshark.org/`

- **Windows:** The `tracert` program provided with Windows does not allow one to change the size of the ICMP message sent by `tracert`. So it won't be possible to use a Windows machine to generate ICMP messages that are large enough to force IP fragmentation. However, you can use `tracert` to generate small, fixed length packets to perform Part 1 of this lab. At the DOS command prompt enter:

  ```
  >tracert www.mcgill.ca
  ```

  For the second part of this lab, you can download the captured packet trace file provided with this assignment[2].

Do the Following:

- Start up Wireshark and begin packet capture. (*Capture−>Start* or click on the blue shark fin button in the top left of the Wireshark window).
- Enter two `traceroute` commands, using www.mcgill.ca as the destination, the first with a length of 56 bytes. Once that command has finished executing, enter a second `traceroute` command for the same destination, but with a length of 3000 bytes.
- Stop Wireshark tracing.

If you're unable to run Wireshark on a live network connection, you can use the packet trace file, *ip-wireshark-trace1-1.pcapng*, referenced in footnote 1. You may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, as you explore the questions below.

Part 1: Basic IPv4

In your trace, you should be able to see the series of UDP segments (in the case of MacOS/Linux) or ICMP Echo Request messages (Windows) sent by `traceroute` on your computer, and the ICMP TTL-exceeded messages returned to your computer by the intermediate routers. In the questions below, we'll assume you're using a MacOS/Linux computer; the corresponding questions for the case of a Windows machine should be clear. Your screen should look similar to the screenshot in Figure 1, where we have used the display filter "udp||icmp" (see the light-green-filled display-filter field in Figure 1) so that only UDP and/or ICMP protocol packets are displayed.

[2]The packet capture trace file and the content of this lab are primarily based on the Wireshark labs provided with the optional textbook "Computer Networking: A Top-Down Approach", 8th ed., J.F. Kurose and K.W. Ross. The trace was made using Wireshark running on one of the author's computers, while performing the steps in this Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name.

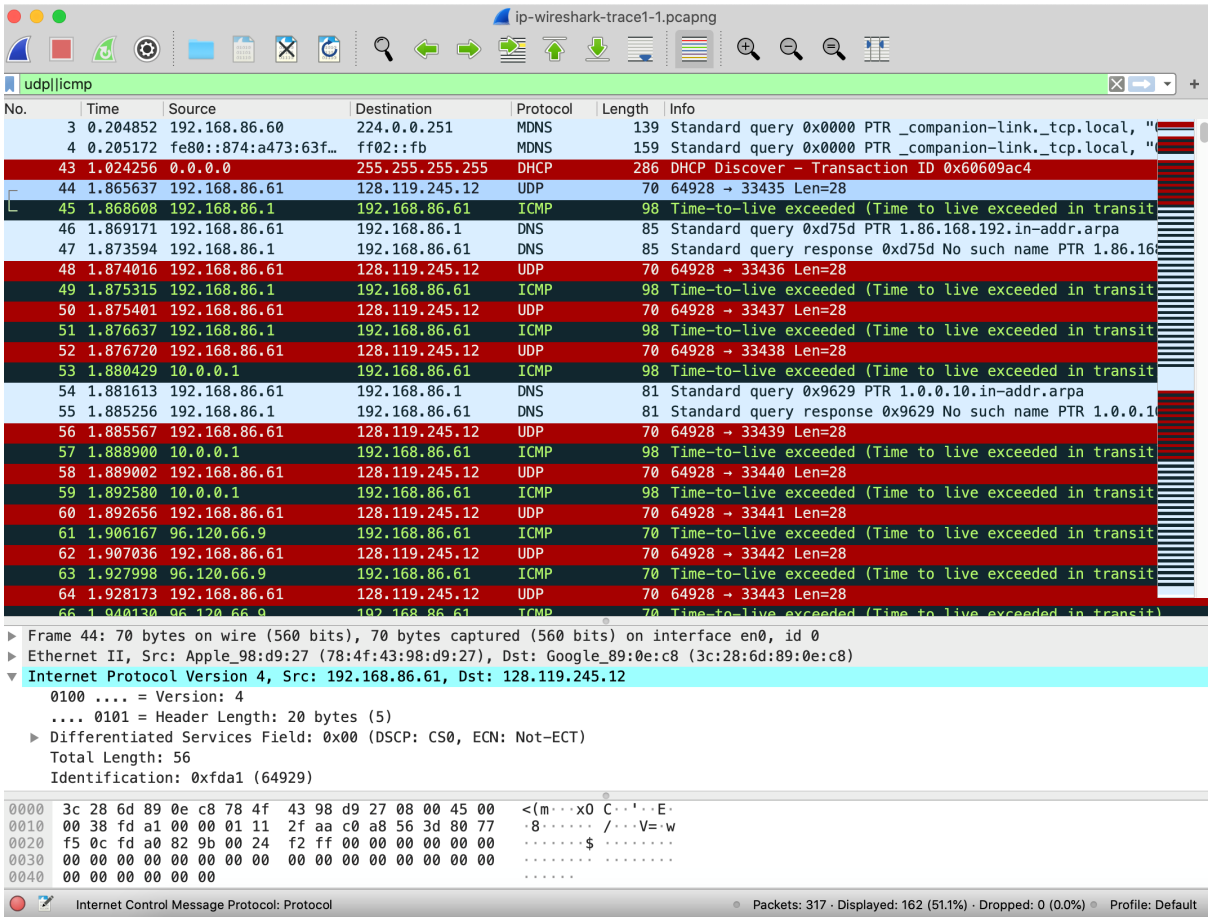


Figure 1: Wireshark screenshot, showing UDP and ICMP packets in the tracefile *ip-wireshark-trace1-1.pcapng*.

Answer the following questions and support your answers with screenshots that show where in the packet you've found the information that answers a question.

[1] 1. Select the first UDP segment sent by your computer via the `traceroute` command to www.mcgill.ca. (Hint: this is 44th packet in the trace file in the *ip-wireshark-trace1-1.pcapng* file in footnote 1). Expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?

[2] 2. What is the value in the time-to-live (TTL) field in this IPv4 datagram's header?

[2] 3. What is the value in the upper layer protocol field in this IPv4 datagram's header? [Note: the answers for Linux/MacOS differ from Windows here].

[2] 4. How many bytes are in the IP header?

[3] 5. How many bytes are in the payload of the IP datagram? Explain how you determined the number of payload bytes.

[2] 6. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.

Part 2: Fragmentation

In this section, we'll look at a large (3000-byte) UDP segment sent by the `traceroute` program that is fragmented into multiple IP datagrams.
Sort the packet listing from Part 1, with any display filters cleared, according to time, by clicking on the *Time* column.

[**1**] 7. Find the first IP datagram containing the first part of the segment sent to 132.216.177.157 by your computer via the `traceroute` command to www.mcgill.ca, after you specified that the `traceroute` packet length should be 3000. (Hint: This is packet 179 in the *ip-wireshark-trace1-1.pcapng* trace file which is designated to 128.119.145.12). What information in the IP header indicates that this datagram has been fragmented?

[**2**] 8. How many fragments are holding of the data of this UDP segment? Explain how you determined the number of fragments.

[**3**] 9. Inspect the IP datagrams containing the fragments of the fragmented UDP segment and list the fields change in the IP header between each fragment datagram. What information in the IP header for this packet indicates whether this is the first fragment, a latter fragment, or the last fragment of that segment?