# CS422 Lab1: `traceroute` and network analysis via `wireshark`

**Due: 23:59:59 PM, Fri Sep 12, 2025**
Total points: 40 points

## 1 Goal

In this lab, you will begin with a simple task to learn how to use `wireshark` (or `tcpdump`) to capture data packets (pcap files) while performing a **traceroute** to a specific host. Afterwards, you will use `wireshark` (or `tcpdump`) to analyze the captured packets to understand how the **traceroute** was performed. Finally, you will create Python programs to analyze both the pcap files you collected and the provided one, which differs from the **traceroute**.

## 2 Before you start

1. Please download and install `wireshark` (`https://www.wireshark.org`). If your computer does not support `wireshark`, you can use `tcpdump`. The difference is that `wireshark` supports UI while `tcpdump` not. Both can be used to capture and analyze packets for this lab.
2. Please learn how to use `wireshark` (or `tcpdump`). You can find a lot of online resources useful, for example, `wireshark` Cheat Sheet (e.g, https://www.comparitech.com/net-admin/wireshark-cheat-sheet/) and `tcpdump` cheat sheet (or `man tcpdump`).
3. You must work individually on this assignment. You can work on your own computer (recommended) or on the XINU machines (xinu1.cs.purdue.edu, xinu2.cs.purdue.edu, etc.) in HAAS 257 or other lab machines (e.g., amber01.cs.purdue.edu, amber02.cs.purdue.edu, etc) in HAAS G050, which all are remotely accessible.

## 3 Part I: `traceroute` with `wireshark` (25 points)

### 3.1 Data Collection (5 points)

Please run `traceroute` to the following destination (www.cs.purdue.edu) while turning on `wireshark` (tcpdump) to capture packets. You need to start packet capturing before you run `traceroute` and stop packet capturing when `traceroute` stops.

Please run `traceroute` as follows:

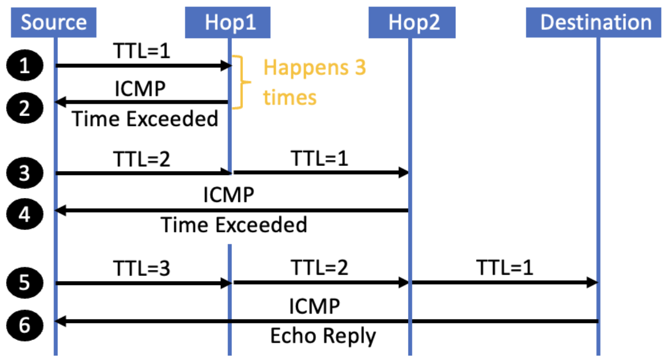
```
traceroute -I www.cs.purdue.edu
```

*Tip: You can run "man traceroute" to learn why we want to use "-I".*

Please save two files: (1) the printout of your `traceroute` as **trace.txt**, and (2) its corresponding packet traces captured by `wireshark` as **trace.pcap**.

Note: please reduce the volume of pcap files if they are too big (> 2MB.) You need to figure out how to filter out some packets which are not relevant to traceroute. Please make sure each pcap file < 2MB.

Note: you are encouraged to test with more destinations other than www.cs.purdue.edu. Hope that you have fun to learn how `traceroute` gives a glimpse of the Internet topology.



## 3.2 Packet Trace Analysis (10 points)

Please Write a **Python** program to analyze the captured pcap file and extract RTT (Round Trip Time) values for each hop in the previous `traceroute` experiment.

1. Please first generate hop.txt based on the output of traceroute. Each line is
   hop k: ip-of-router-at-hop-k
   Please start with hop k = 1 till the last hop. ip-of-router-at-hop-k is the IP address of the router at hop k. Please feel free to write a program or manually generate **hop.txt** from **trace.txt**.

2. Please write a program called **pcapTraceroute.py** to analyze the previously captured pcap file. Please run the program as follows:
   `python pcapTraceroute.py input.pcap hop.txt`
   Here, **trace.pcap** is the pcap file captured by `wireshark` in §3.1 and **hop.txt** is the above output hop results along with the same `traceroute` experiment.

**Expected Output:** The output of the program should print each row as follows
hop k: ip-of-router-at-hop-k, RTT-1, RTT-2, RTT-3
Start with hop = 1 and continue until the last hop. ip-of-router-at-hop-k represents the IP address of the router at hop k. RTT is the time interval (in milliseconds) between when the ICMP packet is sent by the source node and when the ICMP response is received by the source. Note that this procedure will repeat three times for each hop, and all three RTTs should be returned.
**Special Case:** Please note that sometimes, for hop i, there might be two (or even three) different IP addresses. However, the total number of packets sent to those IPs will still sum up to 3. In this case, the output would be:
hop i: ip-1, RTT-1
hop i: ip-2, RTT-2, RTT-3
Please make sure that your program **pcapTraceroute.py** can handle this special case.

**Hints:** Use **ICMP** to locate the packet records to/from the routers involved in the traceroute. By examining the **source IP, destination IP, and TTL** values, you can determine the RTT for each hop.

Note: you will be unable to get the delay if it returns "* * *". In this case, just tell us what you can measure and what you not.

Note: your program will be tested with more pacp files captured with `traceroute` experiments to other destinations.

## 3.3 Answer Questions (10 points)

Based on your `traceroute` experiment in §3.1 (trace.txt and trace.pcap), please answer the following questions:

- (1 point) What is the maximum TTL value observed in the pcap file, and how does it relate to the hop count? If you can't, please explain why not.
- (1 point) Please go to its pcap file, locate the records as the response from the second-hop router (hint: use the IP address of the routers on the way to locate the traces). Print only these records (or print the screenshot using `wireshark`).

- (1 point) What is the average of the round trip delays to the first-hop router towards this destination? If you can't, please explain why not.
- (2 points) Explain how the router receives a packet with TTL=1, and how this is reflected in the captured packets.

Based on the **given test.pcap** file (which was captured during one `traceroute` experiment), please answer the following questions:

- (1 point) Please locate the packets with ICMP Echo Reply in the pcap file. Print only these records (or print the screenshot using wireshark). Check their source IP. What can you conclude from them?
- (2 points) Check the source and destination IP addresses of all traceroute *request* packets in the pcap file. Are they always the same? Why is traceroute designed this way?
- (1 point) What is the average of the round trip delays to the first-hop router towards this destination? If you can't, please explain why not.
- (1 point) What is the average of the round trip delays to the final destination? If you can't, please explain why not.

## 4 Part II: Analyze non-traceroute pcap files (15 points)

### 4.1 Implementation (10 points)

1. **Step 1: Data Processing (2 points)**
   In the second part, please first write another python program called **dataProcess.py** to process the two given pcap files. Please run the program as follows:
   `python dataProcess.py sender.pcap receiver.pcap`

   **Expected Output:** we expect you to extract important information of each packet sent/received by the sender/receiver. Please generate a **senderPackets.txt** file where each line follows this format (similarly, please also generate **receiverPackets.txt** for **receiver.pcap**):

   `sequence number, next sequence number, timestamp`

   We will use these two files as inputs for the following questions.

   **Step 2: Data Analysis (6 points)**
   Please create **getTimePerPacket.py** to analyze the generated two **.txt** files to compute the RTT per packet (more precisely, per packet of interest). Please run the program as follows:
   `python getTimePerPacket.py senderPackets.txt receiverPackets`

   You need to pair the packets sent by the sender with those corresponding ones received by the receiver. For each pair, please calculate the consumed time per packet. Please note that some packets sent by the sender might not be received by the receiver (these are called missing packets) or some packets might be received more than once (these are duplicate packets or retransmitted packets). Please also identify and report all missing and duplicate packets.

   **Expected Output:** we expect you to report the status of each packet sent by the sender. Please generate a **result.txt** file where each line follows this format:

   `sequence number, next sequence number, status, consumed time`

   If the packet is missing, please set the status value to be 0 and set the consumed time to a very large large number (say, 5000 ms). If the packet is received and is not a duplicate packet (received once), please set the status value to be 1 and record the correct consumed time (ms). If the packet is received but received more than once, please set the status value to be 2 and record the consumed time using the largest receiving timestamp - the initial sending timestamp (the longest time). Please sort the lines in result.txt in the increasing order of the sequence number of packets sent by the sender.

**More details:** When you check the sequence number, the next sequence number, and the timestamp extracted from the pcap files, you may notice the following cases.
**Case 1**: the sender sends a single packet (e.g., (100, 200, t1)), but on the receiver side, this packet may be split into multiple smaller packets (e.g., (100, 150, t2) and (150, 200, t3)). In this case, the consumed time should be calculated as (t3 - t1), and the status is 1. Therefore, you have to check both the sequence number and the next sequence number to match them.
**Case 2:** the sender sends a data packet (e.g., (100, 300, t1), but on the receiver side, data packets are not fully received, say, (100, 150, t2) and (200, 300, t3) are received. In this case, (150, 200) is missing and its status is 0. The consumed time for (100, 150) is t2 - t1, and the consumed time for (200, 300) is t3 - t1.
**Case 3:** the sender sends a data packet (e.g., (100, 300, t1), but on the receiver side, **at first**, data packets are not fully received, say, (100, 150, t2) and (200, 300, t3) are received, while (150, 200) is missing. **Later**, the sender **resends** (100, 200) at time t4. Then, the consumed time for (100, 200) is t4 - t1, and the status is 2. While for (200, 300) the consumed time is t3 - t2, and its status is 1.

Note: The given pcap files may not contain all of the above special cases. However, please test your code with as many cases (or even more than the ones above) as possible using fake input **.txt** files created by you. We will test your code with multiple hidden cases.

2. **Step 3: Plotting (2 points)**. Please write a python program called **plot.py** to plot the results generated in the above step. Please run the program as follows:

   ```
   python plot.py result.txt
   ```

   In this output plot, the x-axis represents the index of each packet sent by the sender (starting from 1). The y-axis represents the consumed time for each packet (in milliseconds). Note if the packet status is not 1, you don't need to plot the consumed time. Please submit both plot.py and your output plot **plot.png**.

## 4.2 Answer Questions (5 points)

Please analyze the given sender.pcap and receiver.pcap, and answer the following questions.

- (1 point) What is the sender's IP? What is the receiver's?
- (1 point) How many ***unique*** bytes does the sender send in total? For example, for a packet (100, 200), the total bytes will be 200 - 100 = 100 bytes.
- (1 point) How many bytes are missing? (Check those packets with status = 0.)
- (2 points) What is the average time consumed to send data packets (excluding lost, duplicate and retransmitted packets)? What is the minimal and maximal time?

## 5 Materials to turn in

You will submit your assignment on `gradescope`. You submission should zip all the files into "lab1_UID*.zip", including the following files

- **Program files**: pcapTraceroute.py, dataProcess.py, getTimePerPacket.py and plot.py.
- **Output Files**: trace.txt, trace.pcap, senderPackets.txt, receiverPackets.txt, result.txt and plot.png. Please, make sure that `the wireshark` packet traces (pcap files) is < 2MB.
- **Report (pdf or word)**: First, please include your name and UID on the cover page. In the report, Please answer the above questions and paste the required printouts in the right place.

## 6 Support

Questions about the assignment should be posted on Campuswire or asked during PSOs.