Assignment 1 (Part 2)
Networking basics

Version: January 18, 2023

This assignment is mostly about setup and learning things you will need in the future: working on the command line, using Wireshark to analyze traffic, working with Gradle and the example repo.
You will need a PDF or markup document that you will submit on Canvas AND in your GitHub repo (see below - ser321-spring2023-C-asurite), I will call this "document" during the assignment description.
Your document needs to be formatted well, so we can navigate it easily and find your answers. If that is not fulfilled you might lose up to 5 points in this assignment (these points are not listed anywhere, these are additional deductions).

Part I.
Networking

1. Network traffic

This section is comprised of a collection of activities exploring networking protocols and tools for the data link, network (IP), and transport (TCP) layers of the network stack.
Advice: Do not just copy and paste the command from this document; this sometimes leads to issues when done from a PDF. I would also advise you to spend some time understanding what you are doing and not just blindly following the commands.
You will need to create a document in which you will add all screenshots, answers, etc. Please structure your document according to the tasks and activities in the assignment. If you want to you can of course remove information from your screen shots if you think they are things you want to keep private.

Learning outcomes

- Understand the nature of network traffic on your local computing device.

Objectives

- Understand the nature of network traffic at the data link, network, and transport layers.
- Gain basic competency using common command-line and GUI tools to analyze network traffic and routing.
- Explore the utility of the data link layer through the ARP protocol.
- Identify the structures in IP and TCP packets.
- Understanding packet routing (paths) across the open Internet.

Tools needed

- Ipconfig / ifconfig provide information regarding your network interfaces.
- The arp command has different flags on Windows and Linux, but allows you to see how data link and network layer addressing are bound on your local computer.
- The watch command will run a command repeatedly on a set interval (Unix).
- The netcat command gives you a command-line network tool to communicate over sockets using specific protocols and ports.
- The netstat command displays comprehensive network information, including socket states
- Wireshark is a packet sniffing application with mature facilities to filter and capture network traffic. There is an overview video on using Wireshark in the course shell.
- Traceroute and ping may be used to show routes taken by IP packets through different routers ("hops") using ICMP messages.

1.1. Explore the Data Link Layer with ARP (10 points)

Points are for the correct screencaptures and showing the right data in them. Each item is 1-2 points.

Step 1: Capture a Trace (6 points)

Proceed as follows to capture a trace of ARP traffic; alternatively, you may use a supplied trace. To gather ARP packets, we will cause your computer to send traffic to the local router when it does not know the router"s Ethernet address - your computer will then use ARP to discover the Ethernet address.

1 Find the Ethernet address of the main network interface of your computer with the ifconfig / ipconfig command. You will want to know this address for later analysis. Among the output will be a section for the main interface of the computer (likely an Ethernet interface) and its Ethernet address. Common names for the interface are "eth0", "en0", or "Ethernet adapter".

2 Find the IP address of the local router or default gateway that your computer uses to reach the rest of the Internet using the netstat / route command. You should be able to use the netstat command ("netstat -r" on Windows, Mac and Linux, may require ctrl-C to stop). Alternatively, you can use the route command ("route print" on Windows, "route" on Linux, "route -n get default" on Mac). In either case you are looking for the gateway IP address that corresponds to the destination of default or 0.0.0.0. An example is shown below.

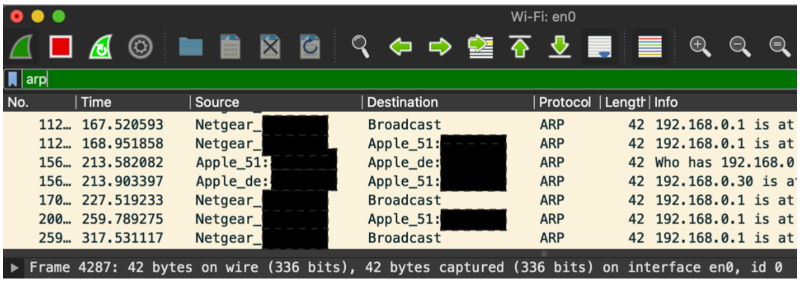
```
amankaushik->~$ netstat -r
Routing tables

Internet:
Destination        Gateway            Flags        Netif Expire
default            192.168.0.1        UGSc           en0
```

```
amankaushik->~$ route -n get default
   route to: default
destination: default
       mask: default
    gateway: 192.168.0.1
  interface: en0
      flags: <UP,GATEWAY,DONE,STATIC,PRCLONING>
 recvpipe  sendpipe  ssthresh  rtt,msec    rttvar  hopcount      mtu     expire
       0         0         0         0         0         0      1500         0
```

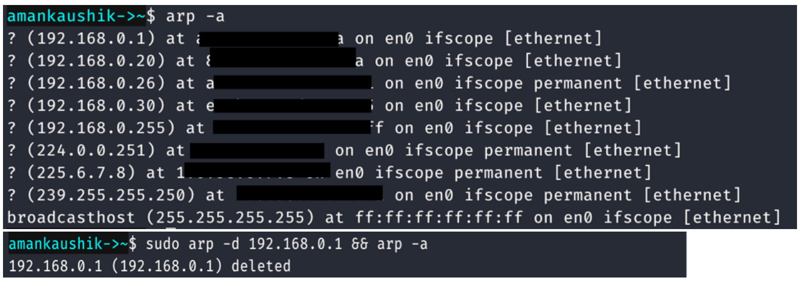
Deliverable: Provide a screen capture of your calls to identify your network interface and gateway. Similar to the above screenshots.

3 Launch Wireshark and start a capture with a filter of "arp". Your capture window should be similar to the one pictured below other than our highlighting. Select the interface from which to capture as the main wired or wireless interface used by your computer to connect to the Internet. If unsure, guess and revisit this step later if your capture is not successful. Uncheck "capture packets in promiscuous mode". This mode is useful to overhear packets sent to/from other computers on broadcast networks. We only want to record packets sent to/from your computer. Leave other options at their default values. The capture filter, if present, is used to prevent the capture of other traffic your computer may send or receive.



Deliverable: Provide a screen capture of your Wireshark instance with the appropriate filters also being visible.

4 When the capture is started, use the "arp" command to clear the default gateway from the ARP cache. Using the command "arp -a" will show you the contents of the ARP cache as a check that you can run "arp". You should see an entry for the IP address of the default gateway. To clear this entry, use the arp command ("arp -d" on Windows). This usage of arp will need administrator privileges to run, so you should login as root on your machine. Note that the command should run without error but the ARP entry may not appear to be cleared if you check with "arp -a". This is because your computer will send ARP packets to repopulate this entry as soon as you need to send a packet to a remote IP address, and that can happen very quickly due to background activity on the computer. Note some flags may be different on Windows. Run "arp -help" to get info on the flags.

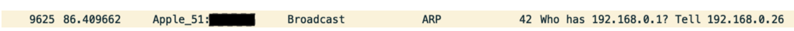


Deliverable: Provide screen captures of your arp -a and arp -d commands. After running the arp -d be sure to run arp -a again to demonstrate the node successfully deleted.

5 Clear your entire ARP cache using arp -d (you need to lookup the proper flag). Also clear your Wireshark entries.

Then fetch a remote page with your Web browser (just go to a webpage). This will cause ARP to find the Ethernet address of the default gateway so that the packets can be sent. These ARP packets will be captured by Wireshark. You might clear the ARP cache and fetch a document a couple of times. Hopefully there will also be other ARP packets sent by other computers on the local network that will be captured. These packets are likely to be present if there are other computers on your local network. In fact, if you have a busy computer and extensive local network then you may capture many ARP packets. The ARP traffic of other computers will be captured when the ARP packets are sent to the broadcast address, since in this case they are destined for all computers including the one on which you are running Wireshark. Because ARP activity happens slowly, you may need to wait up to 30 seconds to observe some of this background ARP traffic.

6 Once you have captured some ARP traffic, stop the capture. You will need the trace, plus the Ethernet address of your computer and the IP address of the default gateway for the next steps.



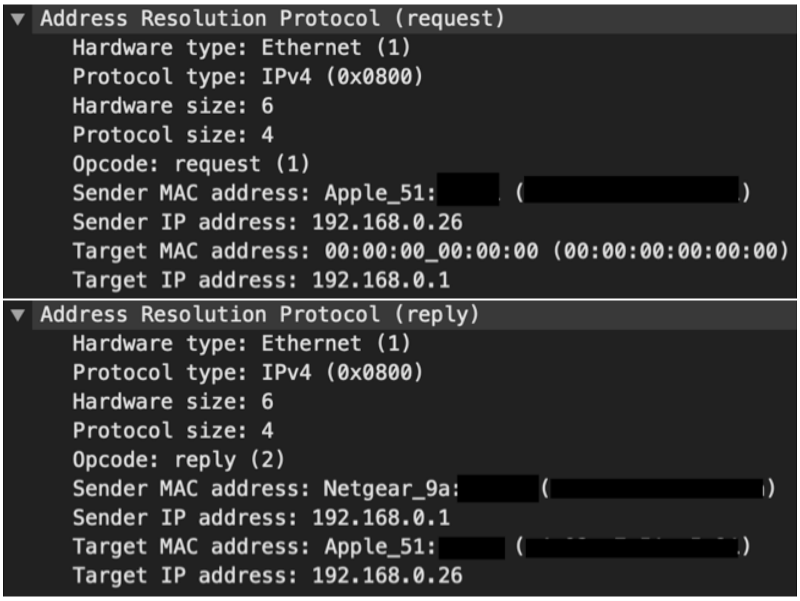
Deliverable: Screen capture the updated trace in Wireshark, add this to your document.

Step 2: Inspect the Trace (2 points)

Now we can look at an ARP exchange! Since there may be many ARP packets in your trace, we"ll first narrow our view to only the ARP packets that are sent directly from or to your computer.

1. Find and select an ARP request for the default gateway and examine its fields. There are two kinds of ARP packets, a request and a reply, and we will look at each one in turn. The Info line for the request will start with "Who has ...". You want to look for one of these packets that asks for the MAC address of the default gateway, e.g., "Who has xx.xx.xx.xx ..." where xx.xx.xx.xx is your default gateway. You can click on the + expander or icon for the Address Resolution Protocol block to view the fields:
   - Hardware and Protocol type are set to constants that tell us the hardware is Ethernet and the protocol is IP. This matches the ARP translation from IP to Ethernet address.
   - Hardware and Protocol size are set to 6 and 4, respectively. These are the sizes of Ethernet and IP addresses in bytes.
   - The opcode field tells us that this is a request.
   - Next come the four key fields, the sender MAC (Ethernet) and IP and the target MAC (Ethernet) and IP. These fields are filled in as much as possible. For a request, the sender knows their MAC and IP address and fills them in. The sender also knows the target IP address it is the IP address for which an Ethernet address is wanted. But the sender does not know the target MAC address, so it does not fill it in.
2. Next, select an ARP reply and examine its fields.
   - The Hardware and Protocol type and sizes are set as before.
   - The opcode field has a different value that tells us that this is a reply.

     Next come the four key fields, the sender MAC (Ethernet) and IP and the target MAC (Ethernet) and IP just as before. These fields are reversed from the corresponding request, since the old target is the new sender (and vice versa). The fields should be filled in since both computers supplied their addresses.



```
▼ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Apple_51:[redacted] ([redacted])
    Sender IP address: 192.168.0.26
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.0.1
▼ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Netgear_9a:[redacted] ([redacted])
    Sender IP address: 192.168.0.1
    Target MAC address: Apple_51:[redacted] ([redacted])
    Target IP address: 192.168.0.26
```

Deliverable: Screen capture the ARP request and reply from this step and add to your document.

Step 3: Details of ARP over Ethernet (2 points)

ARP packets are carried in Ethernet frames, and the values of the Ethernet header fields are chosen to support ARP. For instance, you may wonder how an ARP request packet is delivered to the target computer so that it can reply and tell the requester its MAC address. The answer is that the ARP request is (normally) broadcast at the Ethernet layer so that it is received by all computers on the local network including the target. Look specifically at the destination Ethernet address of a request: it is set to ff:ff:ff:ff:ff:ff, the broadcast address. So the target receives the request and recognizes that it is the intended recipient of the message; other computers that receive the request know that it is not meant for them. Only the target responds with a reply. However, anyone who receives an ARP packet can learn a mapping from it: the sender MAC and sender IP pair.
To look at further details of ARP, examine an ARP request and ARP reply to answer these questions in your document:

1. What opcode is used to indicate a request? What about a reply?
2. How large is the ARP header for a request? What about for a reply? You will need to research this (hint: some sources define what belongs to the header differently, name which source you base your answer on)
3. What value is carried on a request for the unknown target MAC address?
4. What Ethernet Type value indicates that ARP is the higher layer protocol?

Deliverable: Add your answers to the questions above to your document.

1.2. Understanding TCP network sockets (12.5 points)

Steps:

1. The network should be observed for a period of at least 10 minutes during which a decent amount of network activity takes place. Network activity in this context would mean basic web browsing - emails, social media, streaming. Ideally a mix of the mentioned activities. Another option would be, to set this up in a terminal window and then do activities 1.3 and 1.4 and then go back to this section.

2. This network activity observation should be done via netstat. You'll need to continuously monitor the output of the netstat command. This must be automated/scripted.
   a) The watch, grep, and netstat commands could be one way to do this.
   b) Grab readings every 30 seconds and filter for socket connections that are in state ESTABLISHED or LISTEN. You will need to find out the command to use here. Tip: print the current time every 30 seconds before the data that you are grabbing so you can distinguish the different time steps.

3. Once you have the required data points, import them into Excel or a suitable graphing tool and make a line chart of each socket state count over the 10-minute period. So show how many Sockets are listening and established at every 30sec mark (two lines).

Deliverable: Include the command/script you used and the graph of socket states over the 10 minutes in your document.

1.3. Sniffing TCP/UDP traffic (11 points)

Step 1 (TCP) (5.25 points)

1. Setup Wireshark to intercept and log network traffic.
   a) Start a trace on the Loopback: lo0 interface with a filter on the tcp port, tcp.port=3333. The port here could be anything between 1K and 48K, but let"s use 3333.

2. To generate traffic we are going to use netcat. netcat is a utility program for reading from and writing to network connections using TCP or UDP.
   a) To read, run on the command line: nc -k -l 3333. Keep the command window open after running this. Please make sure that port number here (3333) matches the port number set as a filter in step 1a and the port number in 2b.
   b) Open a second command window: To write, run on the command line: nc 127.0.0.1 3333. After running the command, in this window, type in

      SER321

      Rocks!
   c) Make sure that you type the above as it is, with a newline in between

3. To stop both the commands from 2, press Ctrl + C on the terminal windows running the commands.

4. Stop the Wireshark recording and take a screenshot of your data with the appropriate filter.
5. Answer the following questions (0.75 points for each)
   a) Explain both the commands you used in detail. What did they actually do?
   b) How many frames were send back and forth to capture these 2 lines?
   c) How many packets were sent back and forth to capture only those 2 lines?
   d) How many packets were needed to capture the whole "process" (starting the communication, ending the communication)?
   e) How many bytes is the data (only the data) that was send?
   f) How many total bytes went over the wire (back and forth) for the whole process?
   g) How much overhead was there. Basically how many bytes was the whole process compared to the actually data that we did send.

Step 2 (UDP) (5.75 points)

Basically, do the same thing as above just with UDP, I will only give the basics here.

1. Setup Wireshark to intercept and log network traffic.
   a) Start a trace on the Loopback: lo0 interface with a filter on the tcp port, udp.port=3333. The port here could be anything between 1K and 48K.
2. To generate traffic:
   a) Run: nc -k -l -u 3333.
   b) Run: nc -u 127.0.0.1 3333. After running the command, in the same window, type

      SER321

      Rocks!
   c) Make sure that you type the above as it is, with a newline in between
3. Stop the client and server.
4. Stop the Wireshark recording and take a screenshot of your data with the appropriate filter. 0.75 points for each:
   a) Explain both the commands you used in detail. What did they actually do?
   b) How many frames were needed to capture those 2 lines?
   c) How many packets were needed to capture those 2 lines?
   d) How many packets were needed to capture the whole "process" (starting the communication, ending the communication)?
   e) How many total bytes went over the wire?
   f) How many bytes is the data (only the data) that was sent?
   g) Basically how many bytes was the whole process compared to the actually data that we did send.?

   h) What is the difference in relative overhead between UDP and TCP and why? Specifically, what kind of information was exchanged in TCP that was not exchanged in UDP? Show the relative parts of the packet traces.

Deliverable: Put the following in your document

- TCP: Screenshot of Wireshark capture with appropriate filter
- TCP: Answers to all the questions in Step 1
- UDP: Screenshot of Wireshark capture with appropriate filter
- UDP: Answers to all the questions in Step 2

1.4. Internet Protocol (IP) Routing (5 points)

Step

1. Download and install Open Visual Traceroute OR you can use traceroute (*nix), tracert (windows) commands – I only used traceroute since I did not feel like installing anything.
2. Open "OpenVisualTraceroute" (or use traceroute) and on your home network, do a traceroute to www.asu.edu. Export the results from OVT to a CSV.
   a) If using traceroute/tracert, run traceroute www.asu.edu
   b) The results of the above commands would have to be manually exported to a CSV file (copied).
3. Switch to a different network. Possibilities are local coffee shops, libraries (if they do not block traceroute traffic), ASU, or mobile hotspot. Repeat #2
4. Now compare the 2 routes and answer the following questions
   a) Which is the fastest?
   b) Which has the fewest hops?

If for some reason you cannot test a different network please reach out to me directly for an alternative.
Deliverable:

- Route 1 (ASU Network) - screenshot into your document
- Route 2 (Non-ASU Network) - screenshot into your document
- Answers to the questions in 2

1.5. Running client servers in different ways (15 points)

In this section I want you to run the JavaSimpleSock2 example from the repo. You will also need to have Wireshark open to capture the traffic between client and server.

1.5.1. Running things locally (6 points)

Run your client and server locally on your computer (so both of them on localhost). Setup Wireshark so that it shows the traffic for your client/server connection. Send a couple of Strings and numbers to the server and wait for the response.
Take a short screencast (max. 2min) and add the link to this video to your document. In this video talk us through the Wireshark traffic and show us where the Strings and Ints are send to the server and where to find the response on Wirehshark.
In your document add screenshots of your command line windows showing all the command you called and the output in the console.

1.5.2. Server on AWS (5 points)

Now, run your server on AWS and your client locally on your computer. Make sure you use the correct IP and port and that the port on AWS is opened for traffic.
Setup Wireshark to capture the traffic again and send over some data as before.
Add screenshots of your command line windows to your document.
Also describe what changed compared to when just running things locally, what did you need to change on Wireshark, what did you need to change in your Gradle calls (where there changes?)?.

1.5.3. Client on AWS (2.5 points)

Consider the case you want to run your server locally (so on your home computer) and your client on AWS (you do **not** have to do this but you can try). Does this work without issues? Can you do it in the same way as in 1.5.2? Why or why not? What is different?

1.5.4. Client on AWS 2 (3 points)

In this context also explain how the differences in local IP addresses, how your router plays into all of this. Why can you easily reach your server on AWS with a client running in your local network but not as easily go the other direction? And what can you do to reach your server in your local network if you want to reach it from outside your network (you do not have to do that)? What is the "issue" if you want to run your server locally and reach it from the "outside world"?
Deliverable:
Answer all questions from above, non of them are rhetorical in this section. Remember the link to your screen cast, your screenshots and explanations.

2. Submission

On Canvas submit the link to your GitHub repo "ser321-spring2023-C-asurite ". In the main/master branch of this repo you should now have 6 directories (one for each assignment). Make sure you invited ser316asu as collaborator. If you did not you will receive 0 points.
In the Assignment1 directory on GitHub you should have:

- Assign1-2.pdf (or markup) document with all the required information, see assignment. Remember this needs to be well formatted (this will be graded)