Homework 1 (Due: Mon 10/17)

1. **NetActivity Tool**

   Your job is to draw on your understanding of protocol layering and write a tool that reports the activity on your ISP access link. Code up your program in C, working on Mac/Linux/Windows as you prefer, and run it to produce reports on both a test trace we give you and a trace you take in a home. You will also write more code to send your home reports to our class Web server. *We recommend that you do this part with a partner (only two people per team).*

**Motivation**

If you are like many people then you have a home network that is connected to the Internet. But what traffic goes over your ISP access link – what computers and applications are sending and receiving how much traffic? If you are like us then you don’t really know what is happening. So you don’t really know if you bought too much bandwidth from your ISP, or if some of your computers are infected and sending massive amounts of spam, or if everything is fine. Your mission is to fix this by creating a tool that lets people see a simple, high-level summary of the Internet activity in their own homes.

**Setup**

We assume that your home network looks like the one below, or that you can find a friend with this kind of network. The home has a wireless AP (Access Point) that is connected to an ISP and the Internet, usually by DSL or cable. There is at least one computer in the home, likely more, and they send and receive traffic over the access link to and from other Internet hosts.

![Diagram of a home network with an Access Point (AP) connecting to an ISP and the Internet, with computers sending and receiving traffic](image)

We are interested in the traffic that the home computers send and receive over the access link. The way we learn what is happening is from a trace taken at one of the computers called the monitor by wireshark (or a similar program). A trace is simply a log of all the packets that the monitor computer can see. Since wireless is used in the home, the trace will include packets that all the computers send to and from the AP.

**Program Outputs**

When netactivity program you write is run on a trace it will work out how much traffic of what kinds are incoming (to the home) and outgoing (to the Internet) over the access link by each
computer in the house. Computers are called hosts and identified by a link layer or “Ethernet” address like “002564D5108B”. The kinds of traffic are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>TCP to/from port 80 (over IP/Ethernet)</td>
</tr>
<tr>
<td>Secure Web</td>
<td>TCP to/from port 443 (over IP/Ethernet)</td>
</tr>
<tr>
<td>Other TCP</td>
<td>TCP with other ports (over IP/Ethernet)</td>
</tr>
<tr>
<td>UDP</td>
<td>UDP (over IP/Ethernet)</td>
</tr>
<tr>
<td>Other IP</td>
<td>IP under other protocol (not TCP/UDP) (over Ethernet)</td>
</tr>
<tr>
<td>Other Ethernet</td>
<td>Ethernet under other protocol (not IP)</td>
</tr>
</tbody>
</table>

Your program is to provide a count of the number of bytes of traffic of each kind in each direction and for each host. Your report should give this information broken down first by host, then incoming and outgoing, then kind. It is your job to use your understanding of protocols and layering to work out how turn a trace into this report. You will need to look up the packet formats of common protocols (Ethernet, IP, UDP, TCP) as given in your textbook.

Your report is to be in XML format using the tags given in the following example:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<report>
  <description>Browsing the web from my laptop</description>
  <duration>120</duration>
  <bandwidth>3000000</bandwidth>
  <host>
    <address>33557799AACC</address>
    <incoming>
      <traffic>
        <kind>Secure Web</kind>
        <bytes>156000</bytes>
      </traffic>
      <traffic>
        <kind>UDP</kind>
        <bytes>3440</bytes>
      </traffic>
    </incoming>
    <outgoing>
      <traffic>
        <kind>Web</kind>
        <bytes>345230</bytes>
      </traffic>
      <traffic>
        <kind>Other TCP</kind>
        <bytes>3445553</bytes>
      </traffic>
    </outgoing>
  </host>
</report>
```
The first line simply declares it to be XML. If you save it in a file with an XML extension you can then view it in your browser and see if it is well formed. The description tag is a user-readable description of the activity, duration is the time period of the trace in seconds, and bandwidth is the access link speed in bits per second. You can find out your access bandwidth by running netalyzer (http://netalyzr.icsi.berkeley.edu/) or a similar tool. The host tag contains the entire report for a single host whose link layer address is given in the address tag; there may be multiple host tags in a report. The incoming and outgoing parts may have multiple traffic tags inside them, one for each kind of traffic. Do not include a traffic tag for kinds of traffic with zero bytes.

Program Inputs
Your turn-in program must be called netactivity, run with standard command line arguments and send only its report to stdout. This is for grading purposes. The required arguments are:

- Trace file: 
  -t <filename>
- MAC address of gateway: 
  -m <MAC-address>
- IP address of AP: 
  -i <IP-address>

For example: “netactivity -t bob12.pcap -m 00:25:64:D5:10:8B -i 128.208.2.151 > report.xml”.

Your program must be able to run with only these required arguments so that we can test it. You can add other optional arguments that give different behavior as are convenient for you.

Running your program and turn-in
We want you to run your program to generate reports in two ways:

1. Run it on the standard trace we give you. This gives us a known report that you will turn in and we will use as part of grading your program.
2. Gather your own trace in your home, your partner’s home, or your friend’s home, ideally when there is something interesting going on. You can do this with wireshark capturing in promiscuous mode. Then run your program on this trace; working out the IP and MAC address of the AP is a puzzle for you to solve. This gives us a report that you will put up (as described in the next part) for everyone in the class to see the variety of activity that is going on in different homes.

You will turn in your program source files, executable, and report for our standard trace. Submit these items to the dropbox with the rest of your assignment.

Development Steps
Here is how we suggest that you proceed:

1. Pick your development platform on which you write and run code and your monitoring platform on which you gather traces. Ideally these will be the same so that you can get a trace and generate a report on the same computer. The development platform could be Windows, Mac or Linux, and you might use the CSE labs or your laptop. Note that the reference platform on which we test our solutions and can support you is Mac/Linux, so
you should use Windows for development only if you are comfortable with it. The monitoring platform could be Windows, Mac or Linux, as wireshark will run on all of them. This means that if you need to then you can gather a trace on a Windows laptop in your home and process it on a Linux program in the CSE lab.

2. Install wireshark on your monitoring platform, run it to see packets on the network, and capture a trace. See http://www.wireshark.org/ for Windows/OSX and your package manager for Linux. Also run it on our trace file to see the contents; you can try different filters to check whether your program is finding the right Ethernet addresses, etc.

3. Begin with the starter program that we give you. This program will open a trace file, read each packet, and count them. Try it on a short trace that you capture. Your job is to extend this program to analyze the trace and spit out a traffic report.

4. Add functionality to the starter program incrementally. You might begin by going over the packets in the trace and keeping a list of hosts by their Ethernet address; to check this you would run wireshark on the trace and see if it is showing packets with the addresses that you are finding. You might then count packets by the kind of traffic, working your way up from low layer (Ethernet) to high layer (TCP/UDP) protocols. Wireshark will help you check that your report is sensible at each stage. There are programming considerations such as using structs to access packet fields and mapping between host and network byte order that we will cover in recitations.

5. Once you’ve developed your program to have the right inputs and outputs you can run it as instructed above and be ready for turn-in. But we hope you won’t stop there. There are many cool additions you might make to build a better tool: include a visual representation of the access link traffic (e.g., see our report Web server); run live and update the report as time passes; bin traffic in more categories that are more relevant to typical activities, e.g., remote desktop, Skype, Netflix, iTunes, BitTorrent, etc.

2. Post Tool

Your job is to write a program to send the report for your own trace to our Web server so that everyone in the class can see the variety of activity in people’s homes. To do this your program will use the socket API and implement an HTTP POST operation, which is commonly used for form submission and file upload on the Web. You need to learn about sockets; we will tell you what to send to do a POST operation. **We recommend that you do this part with a partner (only two people per team).**

To POST some parameters and a file to the URL amlia.cs.washington.edu/cse461/netactivity/ you connect to the Web server and send the following:

```
POST /cse461/netactivity/ HTTP/1.1
Host: amlia.cs.washington.edu
Content-Type: multipart/form-data; boundary=AsEpArAtOr
Content-Length: xxx
```
The blank lines are important – including the last line which is also blank. All lines end with CR LF (“\r\n”). There are sometimes two dashes before and/or after the boundary value. The values “yourusername” and “yourtracefile” should be replaced with ones specific to your situation. The length shown as xxx must give the length in bytes of the POST that follows after the next blank line. And of course your XML report should go in. The URL we used above is the one to use to upload your report. After sending this, receive the response to find out what happened. For more information consult the HTTP/1.1 specification in RFC 2616.

We have set up our server so that you can POST multiple times, with the newer report for a given username and tracefile updating the older report. POSTing an empty report also will cause the report to be deleted on the server. You can check to see what made it to the server by doing a GET with the username and tracefile parameters, e.g., by opening a URL in your browser such as:

http://amila.cs.washington.edu/cse461/netactivity/?username=bob&tracefile=bob.pcap

You may need to view the page source to see the XML. Note that the above scheme has no security to keep it simple so we are depending on you to not abuse uploads by other students.

Finally, if you view the directory in your browser you will see a visual representation of all the reports that students have uploaded. Point your browser to:

http://amila.cs.washington.edu/cse461/netactivity/

**Development Steps**

Here is how we suggest that you proceed:

1. Pick a development platform as before. Note that our supported reference platform is Mac/Linux, so only use Windows if you are comfortable with it.
2. Learn about sockets. Read your text sections 6.1.3 and 6.1.4. There are also many Web resources. Note that the Windows version of sockets is called winsock and it is slightly different than the Mac/Linux version.

3. Develop a simpler program to GET a page first. Try sending:

```
GET /index.html HTTP/1.1
Host: www.cs.washington.edu
```

Don’t forget the blank line afterwards. If you get it right the response will include the HTML page.

4. Extend your program to do the POST. As well as looking at the response code (which will tell you success or an error) use your browser to GET and check the report you uploaded, and to view the directory to see all reports.

**Running your program and turn-in**

Use your program to upload the report for the trace you collected. Turn in your program source files to the dropbox. We do not need an executable, and we do not need your trace.

3. **Textbook**

1.1, 1.16, 2.4, 2.6, 2.16, 3.17, 3.18