Homework 2 (Due: Mon 10/31)

1. WirelessActivity Tool
Having successfully tackled the access link, we’re now moving to the wireless network in your home! Your job is to draw on your understanding of wireless protocols and 802.11 in particular and write a tool that reports the activity on and health of your wireless network. Code up your program in C, working on Mac/Linux/Windows as you prefer. Run it to produce reports on the test traces we give you (for turn-in), and if possible on a trace you take in a home (for upload to our server). We recommend that you do this part with a partner (only two people per team).

Motivation
Sometimes poor Internet performance can be caused by a problems on your home wireless network rather than problems out in the Internet. There might be lots of competing traffic, the signals to or from your computer may be weak, or there could be interference or collisions caused by other devices. Your job is to analyze traces of the local wireless network to determine how well it is working and print out a summary report with high-level conclusions.

Setup
The setup is the same as before, but the focus is now on your local wireless network rather than the access link.

The trace you are given and should take in a home with 802.11 is in “monitor mode” selecting “AVS radio information”, not promiscuous mode. This trace will give you all packets the computer’s radio could receive on the current 802.11 channel from any device transmitting on that channel. They may be packets to/from your AP or packets to/from another AP that uses the same channel. In monitor mode, the packets captured in WireShark have a real 802.11 header (not a pseudo-Ethernet header), and they also have an AVS pseudo-header that contains parameters such as the radio transmission rate and wireless signal strength. Unfortunately, monitor mode requires the right hardware and operating system support to work. Those of you on Mac/Linux should find that it works out-of-the-box. For those of you on Windows, it probably does not work at all, so just use the standard traces we provide.
Program Outputs

Your program computes three measures for traffic to and from each host that is using the AP:

1. The fraction of airtime that the transmissions take up. For example, a 1250 byte packet sent at 1 Mbps takes 10 ms. If there are 10 such packets in a 1 second trace then the fraction of airtime is 100 ms of 1 second or 0.1. This tells you how much of the wireless capacity is taken up. To compute this you will need the transmission rate that is available in the AVS pseudo-header.

2. The time-averaged data rate of the transmissions. For example, consider three 1250 KB packets sent at 1 Mbps, 2 Mbps, and 5 Mbps. The transmissions take 10 ms, 5 ms, and 2 ms. The time-averaged data rate is \( \frac{1 \times 10 + 2 \times 5 + 5 \times 2}{10 + 5 + 2} = 1.76 \) Mbps. This tells you what data rate a host is getting out of its airtime.

3. The fraction of retransmissions. For example, if there are 8 transmissions and 2 retransmissions then the fraction of retransmissions is \( \frac{2}{8+2} \) or 0.2. This roughly tells you the packet error rate over the wireless link. To compute this you will need to look at the 802.11 header. Retransmissions are identified by a bit in the header, and packets are also counted with a sequence number in the header.

These measures are computed for each host separately for packets send from the AP directly to the host (“fromAP”) and packets sent from the host directly to the AP (“toAP”). These groupings are very similar to the “incoming” and “outcoming” groupings of the previous homework. However, we were only interested in traffic that crossed the access link before. Now we are interested in traffic to and from the AP even if it does not go over the access link.

There is one other measure to compute:

4. The fraction of airtime for all other traffic in the trace. There might be packets sent to or from a different AP (your neighbor, for instance) or broadcast traffic sent from a host to all other hosts (using the special destination address ff:ff:ff:ff:ff:ff).

For all of the above measures, you need only consider 802.11 Data packets. 802.11 defines several packet types (Data, Ack, RTS, CTS, management types). You can ignore non-Data frames.

Your program prints out an XML report as before using the tags in the following example:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<report>
  <description>Wireless activity at home</description>
  <duration>120</duration>
  <bandwidth>3000000</bandwidth>
  <host>
    <address>33557799AACC</address>
    <fromAP>
      <airtime>0.22</airtime>
      <rate>5200000</rate>
      <retries>0.07</retries>
    </fromAP>
  </host>
</report>
```
Much of this is like the report in the previous homework: 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 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 new tags are as follows: <fromAP> replaces <incoming> and <toAP> replaces <outgoing> (because we care about all traffic in the home, not just the packets that go over the access link); <airtime> gives the fraction of airtime as a real number between 0 and 1; <rate> gives the time-averaged data rate as an integer in bits per second; <retries> gives the fraction of retransmissions as a number between 0 and 1; and <otheractivity> is for the airtime other traffic.

**Program Inputs**
Your turn-in program must be called wirelessactivity, 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 AP: -m <MAC-address>
```
For example: “wirelessactivity –t bob12.pcap –m 00:25:64:D5:10:8B > 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 traces 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. If possible, gather your own trace in your home, your partner’s home, or your friend’s home. You can do this with Wireshark capturing in monitor mode (with AVS radio information) on a non-Windows platform. Play around a bit – what happens when the microwave is on, or a computer is near the edge of coverage, etc. Then run your program on this trace to get a report and post it using your earlier program to:

   http://amlia.cs.washington.edu/cse461/wirelessactivity/

The last step will let us see how well people’s wireless is working! We realize that not everyone may be able to do this because of platform issues, so the second step will not be graded.

You will turn in your program source files, executable, report for our standard traces, and answers to these questions:

1. Suppose that you want your program to assess whether the home wireless is “healthy” or “working well”. By working well we mean that it is not limiting or slowing down your Internet experience. Give a procedure using only the parameters in the report to decide whether it is healthy or not. Explain the different ways in which it can fail to be healthy.

2. How much overhead does 802.11 have in your experience using the standard traces? You have computed measures for the Data packets above, and these packets have header information as well as a payload. Consider also non-Data packets for your answer. You may consider all non-Data packets to be pure overhead.

3. The “retries” measure you computed is an estimate of the packet loss rate for a host sending to an AP (or vice versa), not an exact measure. Why? Think about this and explain why it is not an exact measure and the conditions under which it is likely to be a good estimate.

Development Steps

Here is how we suggest that you proceed:

1. Look at the standard wireless trace we gave you in Wireshark and familiarize yourself with the new headers you will need to work with. The AVS header (format DLT_IEEE802_11_RADIO_AVS) is:

   ```c
   struct AVS_header
   {
     uint32 version;
   }
   ```


uint32 length;
uint64 mactime;
uint64 hosttime;
uint32 phytype;
uint32 channel;
uint32 datarate;
uint32 antenna;
uint32 priority;
uint32 ssi_type;
int32 ssi_signal;
int32 ssi_noise;
uint32 preamble;
uint32 encoding;
}

where “uint32” means unsigned integer 32 bits, etc. Thus the datarate is 32 bytes into the header, and it gives the rate in terms of multiples of 100 Kbps. For 802.11, you’ll need to work out the structure. Read the text (Section 4.4) to get an overview.

2. Find the packets for each host, starting from your earlier program. To do this you must work with the 802.11 header, instead of the Ethernet header, and identify the frames that are of Data type.

3. Access the rate information, and check that you have it using Wireshark. With this information you can compute measures 1, 2, and 4.

4. Access the retry bit, and check that you have it using Wireshark. With this information you can compute measure 3.

5. You should now have all the information you need to develop the report.

2. Textbook

3.34, 3.38, 4.23, 4.24, 4.25, 4.38

□□□