Project 3 - Due Wednesday, December 6
CSE 461, Autumn 2000
Neil Spring

1 Overview

Project 3 will be a performance analysis of web transactions using a trace of network packets generated by a packet sniffer. Your group will enhance a packet sniffer to look for TCP retransmissions, congestion control behavior, flow control behavior, and estimate round trip time.

Since running a packet sniffer requires Administrator or root privileges, I will run the packet sniffer on my workstation, and it will trace only the transactions accessing the web server currently running there. The sniffer will generate files (one per day) with the headers from each of these packets. You will then download some of these files and apply the analysis code you write to the file.

For this assignment, you will *not* turn in your code. Instead, you will turn in your analysis, including graphs, performance numbers, insightful observations, and *unique* insights that the particular traces you look at suggest. Expectations for, and formatting of, the write up are discussed below.

Additional background on packet sniffing can be found in the Sniffer FAQ [3] and the tcpdump home page [1]. You may also find the pcap(3) and tcpdump(8) man pages helpful.

2 Phase One

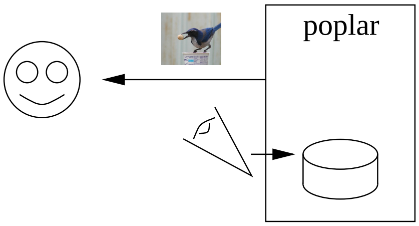

Figure 1: Phase One: Download the Western Scrub-jay from an interesting web client.

The first phase of the project consists of generating data to analyze later. The goal is to download from the web server from as many different places as possible in order to generate diverse network traces. Your task in Phase One is to find some reasonably interesting web client and follow the link from the course web page to download a picture of a Western Scrub-jay. This will generate a trace of network packets sent between poplar and that machine.

All you have to do is download the picture completely, once, and then make note of the following client attributes:

- Client's IP address and/or host name. If you get only the host name, you may run "host [hostname]" on sumatra or the like to find the IP address
- Client's Operating System. This will affect the default advertised window size.
- Location. On campus or off? Seattle? Did you get your friend in Japan or France to download the file?
- Expected performance. Is this your 3 megabit cable modem? a 144 kilobit DSL? your 33.6 modem connected to the modem pool?
- The date and time. One trace file will be generated per day. To find the packet capture trace you just generated, you'll have to remember the date.

You can get the first two attributes even on machines you know nothing about by visiting a script like `http://www.uaf.edu/cgi-bin/printenv.pl`. Copy down `REMOTE_HOST`, `REMOTE_ADDR` and `HTTP_USER_AGENT`.

Please try to have Phase One complete before Friday the 24th. It should only take a few minutes to download the file and collect this information. You can do this while you're bored at your parents house over thanksgiving.

3 Phase Two

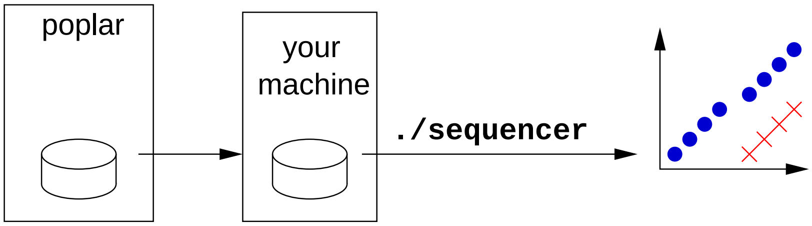

Figure 2: Phase Two: Download the network traces and run `sequencer` to generate a sequence number plot.

Phase two is the initial analysis of these traces. You will analyze both the trace you generated and one other trace, chosen as randomly as you can. The intent is that you can calibrate your analysis skills on the trace you already know something about, then apply these skills to a trace you don't know anything about. If all goes well, the trace you generated and the trace you chose will be noticeably different.

Download and compile `461-pa3-0.1.tar.gz`[1]. This procedure build `./sequencer`, in much the same way pa2 built `./net`. Download one or two of the traces. They have the form [date].tcpd, eg. Nov18.tcpd.

[1] There may be updates, so if the download fails, try 0.11, 0.12, etc.

Run:

```
wget  http://www.cs.washington.edu/education/courses/cse461/00au/461-pa3-0.1.tar.gz
tar xvfz 461-pa3-0.1.tar.gz
cd 461-pa3-0.1
ncftp poplar.cs.washington.edu
  ls
  get Nov18.tcpd
  get Nov19.tcpd
  quit
./configure
make
./sequencer -r Nov18.tcpd -f "ip host 10.1.2.3" -o Nov18-123
```

ncftp is installed on sumatra: you may use your conventional ftp client, but ncftp is somewhat friendlier.

Substitute the date and IP address as appropriate. The filter (-f) passes only packets that are to or from 10.1.2.3 to the sequencer program. This means that even if several different connections are in the same trace, you'll look at only the connections to a particular IP address, and ideally there is only one such connection. The double quotes are important; they pass the three word phrase as a single argument to the program. The output basename option (-o) specifies the prefix to use for each of the output files. After execution, Nov18-123.seq and Nov18-123.ack will be created in the current directory.

To find a list of destinations in a trace,

```
tcpdump -nr Nov18.tcpd | awk '{print $3}' | sort -u | less
```

It may be necessary to substitute 2 for 3 in the previous line, if you use a system other than the instructional linux machines. The version of tcpdump installed there is somewhat unconventional and adds an extra column of information.

To resolve between different connections to the same destination, filter by port instead of IP address:

```
./sequencer -r Nov18.tcpd -f "tcp port 1633" -o Nov18-123
```

Please try to complete your choice of traces to analyze and generate the initial sequence number plots by Wednesday, November 29. This phase requires no coding, unless you attempt to port sequencer to windows or some other machine.

4 Phase Three

Augment the sequencer code to measure the things that need to be measured to fill in the write up. You should have some idea which attributes are important from an initial inspection of the sequence number plot.

The code is designed to output time value pairs, space separated. The Makefile includes some rules to generate encapsulated postscript (EPS) graphs using jgraph [4]. jgraph for Linux is installed in /cse/courses/cse461/00au. Though you are welcome to, I do not expect you to learn this program to generate your graphs for the write up. You are welcome to transfer

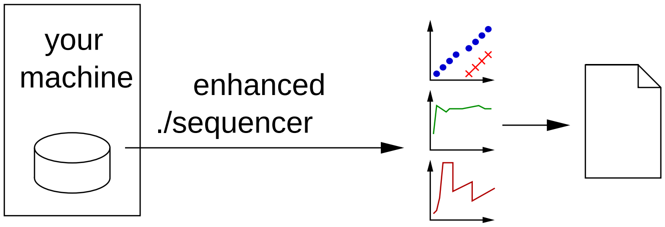

Figure 3: Phase Three: Enhance the sequencer program to generate graphs you will insert into a report.

these output files to a windows machine and use whatever graphing program is convenient for you.

To generate the sequence number plot using jgraph and view it using gv, run:

```
make Nov18-123.sp.eps
gv !$
```

"!$" is a shorthand for the last argument to the previous command that works under csh-based shells. Your default shell (tcsh) on sumatra is csh-based.

Note to Excel users: Please disable the gray background so that the graph is legible, and make sure the graph (not the legend and axis labels) occupies most of the space.

None of these programming tasks is supposed to be challenging - the most difficult is the round trip time estimation, which requires that you keep track of the time each packet was sent. I have included a basic hash table in the distribution for this purpose, and already insert entries into the hash table mapping each packet's sequence number to the time that packet was sent.

5 Write up

I expect a short write up, preferably formatted using two-column, single spaced, 11 point times roman. I expect at least three pages; if it takes more paper to display legible graphs and explain them, so be it. Do not attach your code as an appendix, though you may append additional, possibly zoomed-in graphs. Do not prepend a cover page, as I consider it a waste of paper.[2]

The write up should answer the following questions for each of the traces, providing all information available about the known trace:

- What was the overall performance of the connection: how long did it take to download, how many bytes were transferred, and what is the resulting throughput? You should be able to infer these numbers from the sequence number plot.

- Was the connection's performance limited by the receiver's window? Graph the remaining space in the window left unfilled by the sender.

- Was the connection's performance limited by the congestion control policy? Graph the size of the window (the amount of unacknowledged data in the network at a time).

[2] yes, I am neurotic.

- If there is unfilled space in the receiver's buffer, perhaps the sender's window is the limit. What is the size of the sender's buffer?
- Did packets experience delay due to queuing? Graph the round trip time as a function of time. It is not necessary to implement Karn's algorithm to ignore samples during retransmission events. Compare the round trip delay graph to the window graph above.
- Can you tell what the sender's initial cwnd is? You may have to look at the raw tcpdump output here: it is not necessary that you augment `./sequencer` to handle this.
- Did the connection experience losses? Did these losses slow down the connection or were there enough packets in the queue to absorb the losses? Were there any timeouts? You should be able to answer these questions by interpreting the sequence number plot, looking for periods of slow start, idle periods, etc.
- If there were any losses, did they occur when the window was large or did the size of the window have little effect? I don't expect a statistical analysis, particularly since I don't expect you to see very many losses: a visual, qualitative guess is sufficient.
- Did the rate of transmission vary over time? Apply the TCP Vegas actual transmission rate calculation to generate a graph of transmission rate. It may be appropriate to compare to the expected rate from the related formula (The expected rate is just a different presentation of another graph in this section).

Use your judgement in how much text you need to explain the graphs and answer the questions listed above. Make sure you answer all (appropriate) questions, but please do not list them in your write up.

6 Grading

Grading for this assignment will be based on:

- did you generate and label the graphs correctly? Graphs include
  - Sequence number plot
  - Unused receiver buffer space plot
  - Congestion or actual window plot
  - Round trip time plot
  - Actual transmission rate plot
- did you calculate or estimate the numbers correctly? Including:
  - overall performance: time, bytes, throughput
  - losses, detection, correlation with window size
  - sender buffer size if appropriate
- did you explain graphs and numbers and their implications clearly?

- presentation, formatting, spelling, and grammar.
- anything extra - comparisons between your two traces, additional graphs, identification and explanation of anomalies in previous graphs, etc.

7 Deadlines

The project is due Wednesday, December 6, the last day of instruction. I expect the report to be turned in, in hard copy, at the end of class.

If you find that you desperately need the extra evening, I will accept electronic submission of PostScript (attach a Word document only if you think your PostScript didn't work) documents until midnight by email. I will be very unhappy if it takes me time to print out your document or find that it loads some evil Word virus.

Try to at least have a plan for how to simply measure each performance attribute before Monday the 4th. That will give you some time to come to office hours for clarification.

8 Windows and The Visual Studio

You may be able to do the entire project under windows, although you'll likely need winpcap [2]. If dealing with pcap in this way is a headache, you may instead process the pcap dumps into text output using tcpdump (eg. `tcpdump -nr Nov18.tcpd ip host 10.1.2.3 > file`), then parse the text file using whatever tools you find useful.

I have not tried to make this project work under windows, so there are no guarantees it will work. In particular, the getopt functions may represent a minor headache.

If I were a windows addict and wanted to do most of the programming under windows, I would use the unix sequencer program to generate an intermediate representation of the network trace, consisting of a list of (time, sequence number, data or ack, data length or advertised window size) tuples. I would then transfer this digested form to a windows machine, and write a windows program to fscanf() each line and process it. There may even be appropriate tools to go directly to a graph widget.

References

[1] TCPDUMP public repository. `http://www.tcpdump.org/`.

[2] winpcap. `http://netgroup-serv.polito.it/winpcap/`.

[3] R. Graham. Sniffing (network wiretap, sniffer) FAQ. `http://www.robertgraham.com/pubs/sniffing-faq.html`.

[4] J. Plank. Jgraph. `http://www.cs.utk.edu/~plank/plank/jgraph/jgraph.html`.