CAIDA CoralReef Exercises

Traffic Statistics: where did all those bits come from?

(Unix Help Version)

Level

Introductory.

Prerequisites

To get the most from this exercise a student should have:

1. an ability to perform simple tasks using Unix
2. an ability use use gnuplot, or another plotting tool under Unix.
3. some familiarity with TCP and IP including having seen the headers of these protocols.

System Resources

1. CoralReef or WWW access.
2. gnuplot

Preparation

Obtain the following traces:

- `ODU-926843183.crl.enc`
- `ODU-943099484.crl.enc`
- `ACK-dag-19990708-121553-0-160000-161000.crl`
- `ACK-dag-19990708-121553-1-160000-161000.crl`

A large number of "packet header traces", containing the first 48 bytes[1] of each IP packet are available at http://moat.nlanr.net/Traces/ In addition there are some traces that have been selected specifically for these exercises. Your instructor may have made these available locally. They are also available at: http://www.caida.org/~tonym/traces/

Typically the traces include data from two interfaces, one collecting data from each direction.

The names of these traces consist of a three letter code, a Unix timestamp, and an extension indicating the format of the trace. The three letter code identifies the location of the monitor. For example `ODU-926843183.crl.enc` refers to the Old Dominion University vBNS link collected at 20:26 on Sunday May 16 1999.

If you need to convert the Unix timestamp to a date and time try `date -r time_value` or

```
perl -e 'print scalar(localtime(time_value));'
```

Note that these commands will give date and time in *your* local time zone.

The Waikato University WAND group also collects traces using their DAG hardware. These are named differently to the NLANR traces. The DAG traces start with a three letter code identifying the trace, followed by `-dag-` identifying them as DAG traces, followed by the date and time of the trace, followed by the interface.

For example ACK-dag-19990708-121553-0-160000-161000.crl was collected at 12:15 on Jul 8 1999 on interface 0 of the monitor.

[1] Why 48 bytes?

Background

The single most striking feature of the Internet is that it is large and rapidly getting larger. Every second many hundreds of thousands of millions of bits are transfered from place to pace in the Internet. A natural question that arises is: "What is all this data and who is requesting it." Given the billions of dollars of spending related to the Internet it is surprising that the answers to simple questions like these are not known.

It is part of the mission of the Cooperative Association for Internet Data Analysis (CAIDA) to answer questions of this type. The answers are difficult to find for a number of reasons including:

- The quantities of data involved are very large. Just capturing the TCP/IP headers from a single link for a few minutes may generate a gigabyte or more of data to be stored and/or analysed.
- The decentralised nature of the Internet means there is no single place at which to make measurements. It also means there is no single organisation responsible for coordinating and controlling the Internet.
- Many of the organisations that own resources that make up the Internet do not want to be measured. This might be because they do not have the resources to support measurement or because the do not perceive it to be in the commercial interests of the company to be measured.

Despite these difficulties some progress has been made towards the goal of making meaningful measurement of the Internet. In this exercise we look at some of the measurements that can be made at a single site using CoralReef. Although we will be working with pre-recorded trace files try to keep in mind that CoralReef will also work in real time, analysing data while it is being captured from a network link.

crl_hist

One of the application programs provided with CoralReef is `crl_hist`. This program processes data from a trace file and produces summary output. The output is designed to be easy to manipulate and feed into a plotting program.

The output from `crl_hist` contains a number of sections with different summaries in each section. Run `crl_hist` on the trace `ODU-926843183.crl.enc` and redirect its output into a file. Use the command `crl_hist ODU-926843183.crl.enc > some_file_name`. The `>` symbol means send the output of this command to the file which is named next.

Have a quick look through the output file. There are comment lines at the beginning of each section. These start with a `#`. A full description of `crl_hist` can be found in

Gnuplot

You will need to use a plotting package to complete the exercises below. This exercise guide is written assuming you use gnuplot but you may use any plotting package that you have access to.

Gnuplot is a simple to use plotting program from the GNU project. Grace (also known as xmgr) is an alternative which some people prefer.

Gnuplot plots data in a file. In the simplest case the data should be lines with pairs of xvalue yvalue separated by a space or a comer. If there is only one value per line it is assumed that the x values start at 0 and increase by 1 each line.

To plot data start `gnuplot` and use a command line: `plot "filename" with lines` Don't forget the quotes around the file name. If you prefer you can specify `with points` or `with linespoints`. Extra lines can be added by including more filenames (with or without a `with` clause) separated by a comer. For example:

```
plot "foo", "bar" with linespoints
```

To print a guplot graph set the output type to the type of printer with the set printer command and send the output to a file with the set output command. For example:

```
set terminal postscript color
set output "ts.ps"
```

There is a lot more gnuplot can do for you if you are adventurous. Try the online help. E.G. `help plot`

Exercises

In the sections that follow you will work with some of the sections of the output from `crl_hist` to extract some of the characteristics of some traffic.

Protocol types

The fourth section of the report (titled `#Traffic breakdown by protocol`) lists the number of packets that were found in each of the IP protocol types.

1. Plot a histogram of the total number of packets for each protocol. To do this you must first extract the first and second columns from this section of the report. You can use a command like:

   ```
   crl_hist tracefile | awk '/Traffic breakdown by protocol/,/TCP Traffic by
    { print $1 " " $2}' > tmpfile
   ```
   [2]

   This selects all the lines from `/Traffic breakdown by protocol/` to `/TCP Traffic by source port/` and returns the first (`$1`) and second (`$2`) fields from each line. The `|` character joins two Unix commands so the the output of the first command becomes the input of the second command. The `awk` command above specifies that all lines from the

[2]This command must be entered as a single line, it has been broken here to fit on the page.

one that contains `/Traffic breakdown by protocol` to the one that `contains TCP Traffic by source port` should have the command in the { and the } executed on them.

You can then use `gnuplot`, with style impulse rather that lines or points, to get an approximation to a histogram.

Print the graph and label (by hand if you wish) each column with the name of the protocol. You will probably need to look up the IP protocol number in RFC1700.[3]

2. The average packet size is different for different protocols so the protocol with the largest number of packets is not necessarily the one that carries the most data. Column 4 of this part of the `_crl_hist` output shows the number of bytes carried by each of the protocols.

   Plot a similar plot to the one you did for question 1 for the number of bytes by protocol.

   It is difficult to compare these two graphs because the scales are percentages of the total rather than as absolute values. The percentages are given in columns 3 and 5 of the output of `_crl_hist`. Plot a single graph with the percentages for both bytes and packets. (You can use `gnuplot` to plot two, or more, sets of data using a single plot command and separating the plot specifications with a comma.) For example `plot ''x'' with imp, plot ''y'' with imp`

   To be able to see both impulses you may need to add a small offset (say 0.25) to the protocol number for one set of percentages.

3. One reason to study traffic mixes is to see if the traffic patterns are changing over time. Plot a graph of the protocol percentages from the `ODU-926843183.crl.enc` (from May 16 1999) and the `ODU-939721940.crl.enc` (from Nov 21 1999) traces.

   Do you think the differences between these graphs indicate a significant change in traffic patterns or might they be the result of significance?

   If you have time carry out the check or test.

[3]The RFCs are a set of documents that include standards and other information about common practice in the Internet. They can be found at many web sites including the

Traffic by Port

Neither TCP or UDP are used directly by users. Instead users use applications like a web browser, telnet or ftp. These applications make use of an application protocol (HTTP for example) and that protocol is carried inside TCP or UDP packets.

When a TCP or UDP packet arrives at its destination a decision needs to be made as to which application protocol the packet is passed to for processing. The destination port number field of the TCP or UDP packet indicates what application protocol is being used. For example port 80 indicates http. The port numbers below 1024 are reserved for well known protocols, such has http and telnet. The port numbers above 1023 are used for local protocols and for unique identification of the client as described below.

It is possible there there could be more than one connection between a particular pair of machines using the same application protocol. For example two users of a multiuser machine might be browsing pages on the same web server. When this occurs all the connections will have the same source and destination IP addresses and the same destination port number. Without further information is is not possible to tell which connection a packet belongs to. To avoid this confusion, when a TCP connection is established an unused port number is chosen by the TCP software that initiates the connection. This number is transmitted in the source port number field of the TCP packet. As a consequence any TCP packet can be associated to the correct TCP connection using the source and destination IP addresses and the source and destination port numbers.

Collecting the port number from TCP and UDP packets can give a valuable estimate of the proportion of traffic created by different applications. The three sections of the output of `crl_hist` following the breakdown by IP protocol contain summaries of this sort. They are:

- TCP Traffic by source port
  The top 10 source ports.
- TCP Traffic by destination port
  The top 10 destination ports

- TCP Traffic matrix
  The traffic for each pair of source and destination ports. Before this table is generated ports >= 1024 are replaced with 0. Because well known applications have port numbers < 1024[4] this heuristic, in most cases, classifies packets by application type.

4. In the complete Auckland trace (for both directions) what is the proportion of web traffic (by packets and by bytes)? Remember to include both interfaces and traffic toward and away from the web server.

5. In this trace what is the ratio of traffic entering web servers to traffic leaving web servers?

6. Give the name of protocol that has the largest average packet size. Why do you think this is the case?

Packet Size Distributions

When a packet is forwarded by a router there is a fixed overhead for processing each packet, irrespective of its size. As a consequence one of the things that router manufacturers are particularly interested in is the distribution of packet size and the maximum number of small packets that arrive in a row.

The sections of the output of `crl_hist` entitled `Packet and byte counts by IP length` and `Packet Run Length Histograms` provide this information.

7. What percentage of packets were part of a run of 20 or more packets of less than 45 bytes each were there in the Auckland interface 1 trace? Compare this with the ODU trace.

8. Plot a graph of packet size against the number of packets of that size for the two traces.

[4]Actually there are some ports above 1023 that have come to be used as well know ports. For example the game quake is often run on port 27920. `crl_hist` includes these ports with those below 1024.

Because of the way changes cluster together it is sometimes difficult to see the detail of a plot of the number of packets. As an alternative a cumulative percentage plot can be produced. In the cumulative plot each point shows the percentage of packets that are this size or smaller. The output of `crl_hist` has a column for producing this type of plot.

9. Plot the cumulate percentage of packets against packet size for each of the two traces.

10. Plot the cumulate percentage of bytes against packet size for each of the two traces.

11. Look carefully at the

Variability

So far in this exercise we have only looked at traces from two sources. It is not clear whether the results we have seen are typical of all Internet traffic, are characteristic of this site or perhaps are only characteristic of these traces.

12. To get a sense of the variability of the metrics, pick one of the studies we have investigated in this exercise. Fetch traces for 8 different days for one site for 8 different sites on one day. Repeat the analysis for each of these traces. Comment on the variability. Can you say anything for sure about the results you have?

    It is fairly simple to write a Unix command that will repeat an operation on each file in a group of files. The details depend a little on the command processor (a.k.a. shell) that you use. Most systems have the bash shell. The following example assumes that you are using bash, if it doesn't work for you try typing `bash` first and typing control-D when you have finished.

```
 for file in list-of-files-and-or-wild-cards
do
some-processing-on $file
done
```

For example to run crl_hist on all file in the current directory starting with ODU you might use:

```
 for odufile in ODU*
do
crl_hist $odufile > $odufile.out
done
```

It would be good to compare traces from different kinds of environment (e.g. commercial ISP traces compared with research and education traces, or backbone traces compared with user traces). Unfortunately a wide range of traces are not easily available.

13. What differences do you think there are likely to be between the traces you studied in this exercise and a trace collected from the backbone of a commercial ISP?

Conclusion

In this exercise we have worked through some of the traffic analysis that can be carried out directly from packet trace headers. In particularly we have investigated the output of `crl_hist` which processes packet headers into a number of output formats that give different summaries of the trace.

Many other types of analysis can be done using CoralReef either from `crl_hist` output, other CoralReef applications or by writing your own applications.