Inverted Capacity Extended Engineering Experiment

CFTStat: A tool for passive TCP analysis, statistics and user's web experience gathering.

Claudio Favi[1]
Centre for Advanced Internet Architectures, Technical Report 040227C
Swinburne University of Technology
Melbourne, Australia

***Abstract*-This paper documents CFTStat, a passive TCP analysis tool based on an existing software called Tstat [1]. The aim of this work is to assess the current user's web experience and network characteristics. The global scope is set by the ICE[3] [2] project in which the data provided by CFTStat will be simulated in an inverted capacity network.**

Keywords- Networks, Traffic Characterization

I. Introduction

TStat was started as an evolution of tcptrace [3] created by Shawn Ostermann at Ohio University. Tcptrace was created to analyse network trace files and outputs information on each TCP connections seen.

TStat adds a few more statistics to Tcptrace already impressive set of statistics. Instead of simply recording values, means and standard deviations of the measured quantities, TStat also records histograms representing the distribution of these quantities during a specified period of time. A total of 79 different histogram types are available, comprising both IP and TCP statistics. They range from classic measures directly available from packet headers (e.g., percentage of TCP or UDP packets, packet length distribution, TCP port distribution, ...), to advanced measures, related to TCP (e.g., average congestion window, RTT estimates, out-of-sequence data, duplicated data, ...). A complete log also keeps track of all the TCP flow analyzed, and is useful for post-processing purpose. This TCP flow analysis allows the derivation of novel statistics, such as, for example, the congestion window size, out-of-sequence segments, duplicated segments, etc.

Our contribution is present in two areas. First, we added a per connection packet loss estimator based on Benko and Veres algorithm [4]. Secondly we added functionality to analyse the HTTP protocol and parse "on-the-fly" HTML documents. This was done to be able extract information on how the users use the Internet (which pages, how often) and assess how long they are waiting for the pages to load.

TStat development is maintained by the telecommunication network group of the Politecnico di Torino.

II. How to Use:

A. Installation

CFTStat is distributed as a patch to TStat and both can be found on the ice tools web page [5]. Download TStat and CFTStat archives from there.

Extract the contents of the CFTStat archive with

```
tar zxvf cftstat_v0.93.tgz
```

A new folder called cftstat_v0.93 is automatically created in the current directory. Copy TStat archive into the newly created directory.

```
mv tstat_v0.92.tgz cftstat_v0.93/

cd cftstat_v0.93

make

cd cftstat

./configure
```

check the Makefile, then do:

```
make

make install
```

You can also specify a different installation location by running configure with the –prefix=/new/path option.

B. Measurement scenario

The measurement scenario in which CFTStat is expecting to work in is presented in this section. We define an edge node to be a point in the network that "see" all the exchanged packets between one subnetwork and the rest of the network. Figure 2 present an example of edge node. CFTStat supposes it is processing traces from an edge node.

An edge node can be set up in at least two different ways. First this can be achieved by using a bridge machine which is transparent at the network layer level. Secondly we can use port redirection of some switching equipment to redirect the traffic to the analyser machine.

1This work was performed while working for Swinburne University of Technology. Claudio can be contacted at claudio.favi@epfl.ch

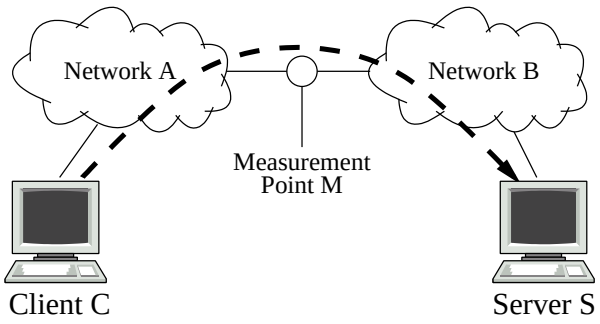

Figure 1: Measurement scenario

Note that tstat does NOT do packet capture from the NIC (Network Interface Card). You'll have to use tcpdump or some other capturing tool for this. Tstat supports several dump file formats (tcpdump, snoop, etherpeek, netmetrix, ns, Dag).

Statistic are collected distinguishing between clients and servers, i.e., host that actively open a connection and host that replay to the connection request. Also Tstat identifies internal and external hosts, i.e., hosts located inside or outside the edge node used as measurement point. Thus incoming and outgoing packets/flows are identified.

Instead of dumping single measured data, for each measured quantity CFTStat builds a histogram, collecting the distribution of that given quantity. Every 15 minutes, it produces a dump of all the histograms it collected.

C. Running CFTStat

First we must create a file that contains the definitions of the internal networks. The file describing the internal networks (let's call it intnet.txt) must have pairs of lines where the first line contains an IP address and the second a netmask.

For example:

```
136.186.229.0
255.255.255.0
```

This describes one internal network. Several other similar pair of lines can also be added.

We now have the choice to do either a live analysis or analyse an already captured traffic trace.

To do a live capture we must pipe tcpdump output to CFTStat in the following way:

```
tcpdump -s0 -w - tcp | tstat -Nintnet.txt -slivetrace stdin
```

To process an existing trace file:

```
tstat -Nintnet.txt tracefile.dmp
```

We create trace files for later analysis with tcpdump:

```
tcpdump -s0 -w tracefile.dmp tcp
```

The trace file can be compressed, if so CFTStat will decompress it on the fly. Note that the -s0 (capture of the entire packets) is required if we want to analyse the HTTP web objects interdependencies. Also, with this option, the dump file size grows particularly fast.

III. The Output

In this section we mainly describe the modification made to the original TStat. Though we describe some of the unmodified output the original documentation is still the reference.

CFTStat creates a directory with the name specified from the -s switch or if the switch was not specified it will use the trace file name extended with ".out". In this new directory it creates another directory with a name representing the date and time when the first packet of the trace file has been captured. Example of output hierarchy:

```
tracefile.dmp.out/
tracefile.dmp.out/15_45_12_Jan_2004.dump.gz.out/
tracefile.dmp.out/15_45_12_Jan_2004.dump.gz.out/log_complete
tracefile.dmp.out/15_45_12_Jan_2004.dump.gz.out/log_nocomplete
tracefile.dmp.out/15_45_1 ... 04.dump.gz.out/log_complete_seglist
tracefile.dmp.out/15_45_12_Jan_2004.dump.gz.out/001/
tracefile.dmp.out/15_45_12_Jan_2004.dump.gz.out/LAST/
tracefile.dmp.out/15_45_12_Jan_2004.dump.gz.out/html_dl
tracefile.dmp.out/15_45_12_Jan_2004.dump.gz.out/html_dep.db
```

In each of the subdirectories (LAST or 001-999) many files are created. They represent histograms of gathered data (for each 15min time window). As already presented, CFTStat makes a difference between internal and external hosts, but also statistics are aggregated according which host started the connection. Four different suffixes in the statistics files are used:

a2b:	stats about packets flowing from the initiator of the connection
b2a:	stats about packets flowing back to the initiator of the connection
usc:	stats about packets going out of the internal network (Italian: uscente)
ent:	stats about packets entering the internal network (Italian: entrante)

Table 1: File suffixes used in CFTStat and TStat.

For each measured quantity TStat builds a histogram. Table 2 describes the files containing the different histograms generated.

File Name	Description
IP Level:	
protocol_*	Protocol Used
ip_tos_*	Type Of Service
ip_ttl_*	Time To Live value
ip_len_*	Packet Length
TCP Level:	
tot_time	Flow duration
tcp_cl_p_*	Flow length Packets
tcp_cl_b_l_*	Flow length Bytes [Large bins]
tcp_cl_b_s_*	Flow length Bytes [Small bins]

File Name	Description
tcp_port_*	Port used
tcp_port_syn_*	Port used for pure SYN segments only
tcp_opts	TCP negotiated options
tcp_mss_*	Maximum Segment Size
tcp_cwnd	"In flight" number of bytes
tcp_rwnd_*	Advertised congestion window estimate
dupb_*	Duplicated bytes ["normal" data transfer]
dupb_sf_*	Duplicated bytes after closing sequence [FIN+ACK or RST]
dupb_rwndz_*	Duplicated bytes when receiver windows was announced zero
ooob_*	Out Of Order Bursts
ooob_zm_*	Out of Order Bursts small bins
ooob_sf_*	Out of Order Bursts when closing sequence already started
ooob_rwndz_*	Out of Order Bursts when receiver windows announced zero
rtt_*	Round Trip Time
Other:	
addresses	IP addresses seen
flow_number	Number of flows seen
not_id_p	Number of non identified packets

Table 2: CFTStat statistics files

The logcomplete file contains the statistics for all the individual completed TCP connections. It is at TAB separated file with a header explaining the meaning of each column.

The lognocomplete contains the same information as logcomplete but for the non completed TCP connections. That is those connections that CFTStat recorded the beginning of but for some reason weren't terminated by a normal FIN or a RST (typically connections ending after the trace finishes).

The logcomplete_seglist contains for each TCP connection two lines (one for each flow direction) with some information about the segments (=packets) in the TCP flow (time when a segment arrived, sequence number, ip_id value, packet_length). This file also contains all the data used by the packet loss estimator algorithm.

We also gather HTTP protocol related statistics. These are closely related to the web users' perception. We keep track of the objects transferred from the servers to the clients. Their size and download time are stored in plain text in the html_dl file. Dependencies between objects are computed by parsing the HTML objects and the resulting list of dependencies is store in hashed format in the html_dep.db file.

IV. Post processing tools

This section presents the scripts we wrote to process CFTStat output. All the post processing scripts are located in the scripts directory in tstat main folder. You need Python 2.3[6]to run them.

- aggregate_pktloss: This script computes the "aggregated" packet loss rate for all the connections in a specified trace file.
- dltime: This script computes the cumulative download time of web object that approximates the real user

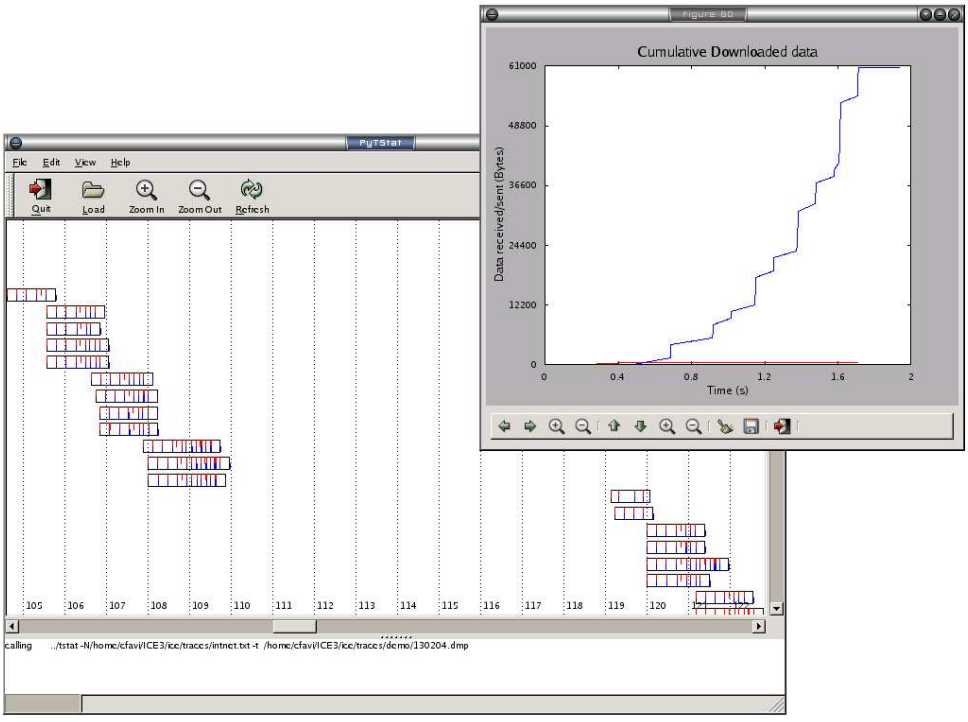

Figure 2: pytstat

wait time. It uses the data stored in the files html_dl and html_dep.db.

- view_depdb: This script can be used to visualize the contents of the html_dep.db file described above.

A. *Pytstat*

Not a post-processing tool per se pytstat is visualization tool on top of CFTStat. It isn't a full interface to CFTStat statistic gathering measurements but rather a tool to visualize when a TCP connection is made and where the different packets in the connection are. It also supports creating time-sequence graphs of the connections. PyTStat is included in CFTStat archive distribution. It requires Python [6], py-gtk [7], matplotlib [10] modules and gtk2 [8] libraries installed[2]. Figure 2 shows a screenshot of pytstat running the rectangular boxes represent tcp connections. When double clicking on a connection a graph representing the cumulative exchanged data is displayed.

V. Hacking the Code

All our modifications to the original TStat have been confined to a few files. Table 3 presents the main files in question with their description.

tstat.h	This is the main header file containing the definitions of the different structures used by CFTStat. Of particular importance is the struct tcb.
http.c	Contains all the HTTP protocol handling routines and html parsing code
pktloss.c	Contains all the packet loss estimation functions. Includes naive and advanced packet loss estimator algorithm.

Table 3: Main files added or modified and their description

Two other two important files are described in table 4.

trace.c	Where the tracing for each packet is done and hooks to the rests of the functionality are coded (particularly http and pktloss functions are called from here)
tstat.c	Contains the main procedure and declaration of global variables.

Table 4: Main CFTStat files

VI. Further work

We discuss here some of many improvements that can be added to our code.

On the HTTP protocol handling procedures, a better html parsing function (using an existing library possibly)can be implemented. Being able to reorder out of order segments that are likely to create problems with the parsing can be useful.

To be able to extract the information gathered by CFTStat in a very flexible way, it would be good to be able to insert that data into a database. This database would could then be queried in different ways.

A visualization tool such as pytstat can be very useful when analysing network traces. A more complete interface to CFTStat output statistics would be the next step into the development of pytstat. An interesting project called nprobe [9] seems to lead the way in this area. Unfortunately the source code for it is not available at the time of this writing.

VII. Conclusion

We presented here CFTStat our modified version of Tstat. Its prerequisites and output have been explained in some detail. We showed the main enhancements implemented and identified some directions in which this work can continue.

References

[1] [tstat] Tstat, "TCP Statistics and analysis Tool" (http://tstat.tlc.polito.it/) (as of February 2004)

[2] [ice] "Inverted Capacity Extended Engineering Experiment (ICE[3])", Centre for Advanced Internet Architectures,

[3] [tcptrace] S. Osterman, "TCPTrace, TCP traces analyser" (http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html) (as of February 2004)

[4] [benko]P. Benko and A. Veres, "A Passive Method for Estimation End-to-End TCP Packet Loss", Proc. IEEE Globecom 2002, Taipei, Taiwan

[5] [ice-tools]http://caia.swin.edu.au/ice/ice-tools.html (as of February 2004)

[6] [Python]Python, "Python Programming Language", (http://www.python.org)

[7] [pygtk]PyGTK, "Python bindings for the GTK widget set", (http://www.daa.com.au/~james/software/pygtk/) (as of February 2004)

[8] [gtk2]GTK 2.x, "The Gimp Toolkit", (http://www.gtk.org/) (as of February 2004)

[9] [nprobe] A. Moore, J. Hall, C. Kreibich, E. Harris and I. Pratt. "Architecture of a Network Monitor", In Passive & Active Measurement Workshop 2003 (PAM2003), Apr. 2003

[10] Matplotlib, "Matlab style python plotting", http://matplotlib.sourceforge.net/ (as of February 2004)

2 On FreeBSD 4.9 install the py-gtk2 package dependencies should take care of installing the rest.