Gliwice, April 19th 2006

Semester: 2
Group: 3
Section: Macrofaculty

Computer Programming Laboratory
ttmap – a TCP timestamp mapper

Author: Paweł Foremski
Tutor: dr inż. Marcin Ciura

1. Introduction

The task of my topic was to write a program detecting and analyzing remote TCP/IP port forwarding basing on the *TCP Timestamps Option*. Such program could later be used eg. to analyze remote IP load-balanced clusters.

TCP is a connection-oriented, stateful protocol, which is the main transmission protocol used in the Internet (typically around 95% of packets). It has been standardized in year 1981, in *RFC 793,* and since then it has received various enhancements, one of which was the *"TCP Extensions for High Performance"*, published as *RFC 1323* in May 1992.

The mentioned document introduces new TCP option - the "*TCP Timestamps Option"* (p. 11), which enables simple RTTM (Round-Trip Time Measurement) to Internet hosts. The idea behind this option is that the sender host places current value of it's "timestamp clock" into outgoing packet (this value is called *TS Value - TSval)* and the remote host should include this value in acknowledgement packets sent back (this value is called *TS Echo Reply - TSecr*). Symbolical definition of the TCP Timestamps Option is as follows:

```
TCP Timestamps Option (TSopt):

   Kind: 8

   Length: 10 bytes

    +-------+-------+---------------------+---------------------+
    |Kind=8 |  10   |   TS Value (TSval)  |TS Echo Reply (TSecr)|
    +-------+-------+---------------------+---------------------+
        1       1              4                     4

   The Timestamps option carries two four-byte timestamp fields.
   The Timestamp Value field (TSval) contains the current value of
   the timestamp clock of the TCP sending the option.

   The Timestamp Echo Reply field (TSecr) is only valid if the ACK
   bit is set in the TCP header; if it is valid, it echos a times-
   tamp value that was sent by the remote TCP in the TSval field
   of a Timestamps option.  When TSecr is not valid, its value
   must be zero.  The TSecr value will generally be from the most
   recent Timestamp option that was received; however, there are
   exceptions that are explained below.

   A TCP may send the Timestamps option (TSopt) in an initial
   <SYN> segment (i.e., segment containing a SYN bit and no ACK
   bit), and may send a TSopt in other segments only if it re-
   ceived a TSopt in the initial <SYN> segment for the connection.
```

Then, the RFC describes how values of the "timestamp clock" should be obtained. It states that they should be at least approximately proportional to real time. However, it does not state on what should be the proportionality factor and how the first value should be obtained.

Analysis of TCP stacks of modern operating system shows what follows:

- value of proportionality factor is the duration of one tick of the system timer interrupt - ie. the *jiffy,*
- the timestamp clock counts time since system start-up.

Such information can be further used for passive remote machine and operating system detection.

These observations, in authors opinion, may be regarded as results of a design flaw in the TCP Timestamps Option standard. The problem could be minimized, for example, if the RFC document recommended using just the Unix time, with given accuracy. Taking into account that many of modern Internet hosts are capable of synchronizing their system clocks with global NTP servers, such improvement could make mentioned detection impossible or hard to conduct.

2. The ttmap program

After successful initialization, ttmap starts analysis of packets received on selected network interface. For this, it uses the *libpcap* library, which injects captured packets to the *ttmap_callback()* function.

Next, the program checks if received packet is a TCP one and whether it has TCP Timestamps Option. If yes, then ttmap reads essential data from it and passes it to the *process_packet()* function. However, if the packet has *RST* or *FIN* flag set, then a special procedure is called, which removes any data regarding the connection being closed, if any.

The *process_packet()* function matches a single packet to a TCP connection. It checks whether number of packets collected in a single connection is enough, and if it is, the control is passed to the *identify_connection()* function.

Now, ttmap has enough sample of packets which were received from a single remote machine to find the proportionality factor (the jiffy), let it be the *a* parameter, and system start-up time, let it be the *b* parameter. For best results, the program uses linear regression method from the *GNU Scientific Library.* Provided that the quality of obtained values is good enough, what is discussed later, an internal database holding information about already identified machines is queried for calculated remote system characteristics. If nothing matches, a new remote machine is detected; if there is a match, then machine's *a* and *b* parameters are corrected by mean value.

Due to various delays and fluctuations that packets traversing the Internet might be subject of, the obtained data might be of low quality, ie. there will not be any linear function matching collected *(time, TCP timestamp)* points. So, for best results, only the points lying close enough to the best-fit line should be accepted as meaningful. The ttmap program checks whether ratio of covariance (returned from GSL) and obtained *a* parameter is small enough. A similar situation appears when querying the internal database for matching machines - here the program user may configure acceptable "delta" for *a* and *b* parameters.

When a new remote machine is detected, an informational message is printed to the *standard output.* Such message contains machine's *a* parameter, with a corresponding remote operating system guess, and *b* parameter, with probable time when remote machine was turned on (in local timezone).

A summary containing statistical information about analyzed remote IP addresses, machines and TCP ports is printed when program ends. Same summary may be obtained at any time, by sending the *SIGUSR1* signal to ttmap process.

3. External specification

The Unix manual page format has been chosen to document program interface. Corresponding ttmap(1) manual page has been attached after this report.

4. Internal specification

The source code of ttmap has been documented using *Doxygen* system and corresponding documentation has been attached to this report, after ttmap(1) manual page.

5. Source code

Source code of ttmap has been attached after Doxygen documentation.

6. Testing

In this section, the most interesting results the author obtained while testing the program are presented.

6.1 SMTP load balancer

```
[root@demiurg ~]# ttmap -i eth0 --no-promisc "src smtp.wp.pl"
Listening on eth0
New machine found behind 212.77.101.1 port 25 (1 so far)
  a=100         (OS guess: UNIX, eg. Linux 2.4)
  b=430711062   (up since +- Fri Mar  3 21:35:50 2006)
New machine found behind 212.77.101.1 port 25 (2 so far)
  a=100         (OS guess: UNIX, eg. Linux 2.4)
  b=1150321374  (up since +- Sat Dec 10 10:32:50 2005)
New machine found behind 212.77.101.1 port 25 (3 so far)
  a=100         (OS guess: UNIX, eg. Linux 2.4)
  b=21253481    (up since +- Thu Apr 20 08:12:21 2006)
New machine found behind 212.77.101.1 port 25 (4 so far)
  a=100         (OS guess: UNIX, eg. Linux 2.4)
  b=1082681793  (up since +- Sun Dec 18 07:55:29 2005)
New machine found behind 212.77.101.1 port 25 (5 so far)
  a=100         (OS guess: UNIX, eg. Linux 2.4)
  b=1833463671  (up since +- Thu Sep 22 18:14:22 2005)
New machine found behind 212.77.101.1 port 25 (6 so far)
  a=100         (OS guess: UNIX, eg. Linux 2.4)
  b=10844794    (up since +- Fri Apr 21 13:01:44 2006)
New machine found behind 212.77.101.1 port 25 (7 so far)
  a=100         (OS guess: UNIX, eg. Linux 2.4)
  b=1797067024   (up since +- Mon Sep 26 21:49:14 2005)
New machine found behind 212.77.101.1 port 25 (8 so far)
  a=100          (OS guess: UNIX, eg. Linux 2.4)
  b=96987572     (up since +- Tue Apr 11 13:51:06 2006)
New machine found behind 212.77.101.1 port 25 (9 so far)
  a=100          (OS guess: UNIX, eg. Linux 2.4)
  b=490163813    (up since +- Fri Feb 24 22:58:05 2006)
New machine found behind 212.77.101.1 port 25 (10 so far)
  a=100          (OS guess: UNIX, eg. Linux 2.4)
  b=9718679      (up since +- Fri Apr 21 16:11:34 2006)
New machine found behind 212.77.101.1 port 25 (11 so far)
  a=100          (OS guess: UNIX, eg. Linux 2.4)
  b=3117191175   (up since +- Tue Apr 26 23:31:32 2005)
New machine found behind 212.77.101.1 port 25 (12 so far)
  a=101          (OS guess: UNIX, eg. Linux 2.4)
  b=1727380708   (up since +- Fri Oct  7 02:41:10 2005)
New machine found behind 212.77.101.1 port 25 (13 so far)
  a=100          (OS guess: UNIX, eg. Linux 2.4)
  b=430711043    (up since +- Fri Mar  3 22:18:01 2006)
New machine found behind 212.77.101.1 port 25 (14 so far)
  a=100          (OS guess: UNIX, eg. Linux 2.4)
  b=1082681746   (up since +- Sun Dec 18 13:54:07 2005)
New machine found behind 212.77.101.1 port 25 (15 so far)
  a=100          (OS guess: UNIX, eg. Linux 2.4)
  b=1366272896   (up since +- Tue Nov 15 16:41:09 2005)
New machine found behind 212.77.101.1 port 25 (16 so far)
  a=100          (OS guess: UNIX, eg. Linux 2.4)
  b=1082681923   (up since +- Sun Dec 18 05:15:53 2005)
New machine found behind 212.77.101.1 port 25 (17 so far)
  a=100          (OS guess: UNIX, eg. Linux 2.4)
  b=1082681742   (up since +- Sun Dec 18 12:13:01 2005)

Analyzed IP addresses:
212.77.101.1 - 17 machines, 206 connections
  machine #1:
    a=100, b=559871915
    OS guess: UNIX, eg. Linux 2.4
    probably up since: Thu Feb 16 21:26:20 2006
    percentage load: 7.28% (15 connections)
    handling ports:
      25 (100.00%, 15 connections)
  machine #2:
    a=100, b=1833462073
    OS guess: UNIX, eg. Linux 2.4
    probably up since: Thu Sep 22 10:47:50 2005
    percentage load: 11.17% (23 connections)
    handling ports:
      25 (100.00%, 23 connections)
```

```
machine #3:
  a=100, b=21253557
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Thu Apr 20 08:06:52 2006
  percentage load: 4.37% (9 connections)
  handling ports:
    25 (100.00%, 9 connections)
machine #4:
  a=100, b=1782762098
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Wed Sep 28 17:11:06 2005
  percentage load: 8.25% (17 connections)
  handling ports:
    25 (100.00%, 17 connections)
machine #5:
  a=100, b=1833463671
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Thu Sep 22 18:14:22 2005
  percentage load: 0.49% (1 connection)
  handling ports:
    25 (100.00%, 1 connection)
machine #6:
  a=100, b=10844904
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Fri Apr 21 13:02:04 2006
  percentage load: 12.62% (26 connections)
  handling ports:
    25 (100.00%, 26 connections)
machine #7:
  a=100, b=1797067020
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Mon Sep 26 22:30:17 2005
  percentage load: 0.97% (2 connections)
  handling ports:
    25 (100.00%, 2 connections)
machine #8:
  a=100, b=96987592
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Tue Apr 11 13:42:39 2006
  percentage load: 7.28% (15 connections)
  handling ports:
    25 (100.00%, 15 connections)
machine #9:
  a=100, b=490163791
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Sat Feb 25 00:35:48 2006
  percentage load: 9.22% (19 connections)
  handling ports:
    25 (100.00%, 19 connections)
```

```
machine #10:
  a=100, b=9718810
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Fri Apr 21 16:09:50 2006
  percentage load: 5.34% (11 connections)
  handling ports:
    25 (100.00%, 11 connections)
machine #11:
  a=100, b=3117204804
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Tue Apr 26 22:43:07 2005
  percentage load: 3.88% (8 connections)
  handling ports:
    25 (100.00%, 8 connections)
machine #12:
  a=100, b=1727381154
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Tue Oct  4 21:50:37 2005
  percentage load: 2.43% (5 connections)
  handling ports:
    25 (100.00%, 5 connections)
machine #13:
  a=100, b=430710998
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Fri Mar  3 22:09:08 2006
  percentage load: 4.85% (10 connections)
  handling ports:
    25 (100.00%, 10 connections)
machine #14:
  a=100, b=1645198107
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Fri Oct 14 09:13:05 2005
  percentage load: 10.68% (22 connections)
  handling ports:
    25 (100.00%, 22 connections)
machine #15:
  a=100, b=1366272796
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Tue Nov 15 18:24:18 2005
  percentage load: 2.43% (5 connections)
  handling ports:
    25 (100.00%, 5 connections)
machine #16:
  a=100, b=1150288348
  OS guess: UNIX, eg. Linux 2.4
  probably up since: Sat Dec 10 14:42:46 2005
  percentage load: 5.83% (12 connections)
  handling ports:
      25 (100.00%, 12 connections)
    machine #17:
      a=100, b=1082681850
      OS guess: UNIX, eg. Linux 2.4
      probably up since: Sun Dec 18 09:18:43 2005
      percentage load: 2.91% (6 connections)
      handling ports:
        25 (100.00%, 6 connections)
  Average uptime: 11699477 sec = 3249.85 h = 135.41 days
```

6.2 HTTP load balancer

```
www.microsoft.com
pjf@pjf:~/projects/cp/ttmap/src$ sudo ./ttmap -i eth1 --no-promisc "src www.microsoft.com"
Listening on eth1
New machine found behind 207.46.19.60 port 80 (1 so far)
  a=10          (OS guess: Windows)
  b=28298677    (up since +- Fri Mar 17 09:17:03 2006)
New machine found behind 207.46.19.60 port 80 (2 so far)
  a=10          (OS guess: Windows)
  b=28298658    (up since +- Fri Mar 17 08:39:18 2006)
New machine found behind 207.46.19.60 port 80 (3 so far)
  a=10          (OS guess: Windows)
  b=28243652    (up since +- Fri Mar 17 11:21:50 2006)
New machine found behind 207.46.19.60 port 80 (4 so far)
  a=10          (OS guess: Windows)
  b=28243677    (up since +- Thu Mar 16 19:34:43 2006)
New machine found behind 207.46.18.30 port 80 (1 so far)
  a=10          (OS guess: Windows)
  b=4848376     (up since +- Thu Apr 13 02:05:53 2006)
New machine found behind 207.46.18.30 port 80 (2 so far)
  a=10          (OS guess: Windows)
  b=4843463     (up since +- Thu Apr 13 05:48:04 2006)
New machine found behind 207.46.18.30 port 80 (3 so far)
  a=10          (OS guess: Windows)
  b=4848120     (up since +- Thu Apr 13 06:07:15 2006)
New machine found behind 207.46.18.30 port 80 (4 so far)
  a=10          (OS guess: Windows)
  b=4844481     (up since +- Thu Apr 13 02:16:02 2006)
New machine found behind 207.46.18.30 port 80 (5 so far)
  a=10          (OS guess: Windows)
  b=4843624     (up since +- Thu Apr 13 03:20:08 2006)

Analyzed IP addresses:
207.46.18.30 - 5 machines, 41 connections
  machine #1:
    a=10, b=4876826
    OS guess: Windows
    probably up since: Thu Apr 13 04:06:38 2006
    percentage load: 29.27% (12 connections)
    handling ports:
      80 (100.00%, 12 connections)
  machine #2:
    a=10, b=4851163
    OS guess: Windows
    probably up since: Thu Apr 13 04:37:16 2006
    percentage load: 14.63% (6 connections)
    handling ports:
      80 (100.00%, 6 connections)
  machine #3:
    a=10, b=4848816
    OS guess: Windows
    probably up since: Thu Apr 13 04:05:53 2006
    percentage load: 17.07% (7 connections)
    handling ports:
      80 (100.00%, 7 connections)
  machine #4:
    a=10, b=4848221
    OS guess: Windows
    probably up since: Thu Apr 13 05:40:02 2006
    percentage load: 21.95% (9 connections)
    handling ports:
      80 (100.00%, 9 connections)
  machine #5:
    a=10, b=4846435
    OS guess: Windows
    probably up since: Thu Apr 13 01:41:27 2006
    percentage load: 17.07% (7 connections)
    handling ports:
      80 (100.00%, 7 connections)
207.46.19.60 - 4 machines, 10 connections
  machine #1:
    a=10, b=28298704
    OS guess: Windows
    probably up since: Thu Mar 16 19:39:41 2006
    percentage load: 50.00% (5 connections)
    handling ports:
      80 (100.00%, 5 connections)
  machine #2:
    a=10, b=28298658
    OS guess: Windows
    probably up since: Fri Mar 17 10:05:51 2006
    percentage load: 20.00% (2 connections)
    handling ports:
      80 (100.00%, 2 connections)
  machine #3:
    a=10, b=28271102
    OS guess: Windows
    probably up since: Thu Mar 16 00:12:30 2006
    percentage load: 20.00% (2 connections)
    handling ports:
      80 (100.00%, 2 connections)
  machine #4:
    a=10, b=28243677
    OS guess: Windows
    probably up since: Thu Mar 16 19:34:43 2006
    percentage load: 10.00% (1 connection)
    handling ports:
      80 (100.00%, 1 connection)
Average uptime: 2709595 sec = 752.67 h = 31.36 days
```

Comment: what is interesting here, is that it seems that the Microsoft company has their HTTP-serving cluster divided into two parts. What is even more interesting is why the most recently rebooted part is serving more HTTP requests? Further research might reveal something that the company does not want others to know.

7. Final notes

The whole ttmap source code and documentation has been released under GNU licenses, respectively: the GNU General Public License version 2, and the GNU Free Documentation License.

During development, the *Subversion* source code versioning system was used. The repository is accessible under following URL:

http://svn.asn.pl/misc/ttmap/trunk/

Further releases of ttmap will be available from:

http://pjf.asn.pl/ttmap/