Exercise V: Analyzing the network traffic

December 3, 2007

Abstract

This work provides basic knowledge of network measurements. It has three part: use of basic tools, use of data provided by measurement platform and analysing network traces.

1 Introduction

This work has two different tasks. In the first one, you will experiment with basic tools studying network performance and its changes. While you are collecting data for the analysis of the first part, you may work for the other analysis task: to analyse data collected by a measurement grid.

2 Basic tools

Majority of operating systems provide a set of utility tools to study network status and performance. These provide easy insigth to study if network problems are local or if the probem is elsewhere in network.

There are differences between usage of commands in different operating systems and utility versions. Examples here are using program versions listed below on GNU/Linux system (those in classroom). If receive errors or dissimilar function, check usage for version you are using. Examples are designed so that no features special to particular version are used.

`dig` 9.2.2 (prints out in a query)

`ping` netkit-ping 0.10-9 (dpkg -S ‘which ping‘; dpkg -l netkit-ping)

`traceroute` Version 1.4a12 (version printed out if run without arguments)

`wget` GNU Wget 1.7 (-V)

In examples used here the charactes you type are printed in **bold**.

2.1 Round-trip latency and connectivity

A very basic test is to check if a host is available. For this purpose, for a part of IP functionality ICMP Echo server was designed. When a host receives ICMP Echo request message, it would respond it by sending ICMP Echo response and copying data part of received message to reply message.[1]

```
prompt> ping -c 3 www.iana.org
PING www.iana.org (192.0.34.162): 56 data bytes
64 bytes from 192.0.34.162: icmp_seq=0 ttl=48 time=190.8 ms
64 bytes from 192.0.34.162: icmp_seq=1 ttl=48 time=191.0 ms
64 bytes from 192.0.34.162: icmp_seq=2 ttl=48 time=190.5 ms

--- www.iana.org ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 190.5/190.7/191.0 ms
```

In example above, `ping` command was instructed to send three ICMP Echo requests to host `www.iana.org`. A line is printed out for each response received including sequence number, time-to-live field value[2], and time between sent message and reply.

For statistics it is printed number of packets transmitted and received with loss percentage and total time for command. For round-trip time minimum, average, and maximum are calculated.

[1] This helps for detection of data-related errors in transmission.

[2] Can be used to estimate number of routers between as initial values for TTL are 64, 255, 128, or 32 commonly.

2.1.1 Nobody home but services still working

For (pseudo-)security reasons many hosts and sites use firewall rules to drop ICMP messages[3] and then one cannot use ICMP echos to test. It is possible to check connectivity for ICMP-blocking servers by contacting the service the server is providing.

An examples of such site one of root servers, `a.root-servers.net`. As shown below, we do not receive any resonse to ICMP messages but when we query for name servers serving `fi`-ccTLD we got reply in 110 ms. This is quite close to last time provided by traceroute. While one must account for some delays in end systems even with ICMP messages, this is especially important for application tests. However, because root servers are serving only name service info for top-level domains (gTLD like `com`, `net` and ccTLD like `fi`, `fr`, `cn`) but know nothing about particular domains like `example.com`, the response time should be comparable to one of ICMP messages.

```
prompt> ping -q -c 3 a.root-servers.net
PING a.root-servers.net (198.41.0.4): 56 data bytes

--- a.root-servers.net ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
prompt> traceroute -f 15 a.root-servers.net
traceroute to a.root-servers.net (198.41.0.4), 30 hops max, 40 byte packets
15  sl-gw20-rly-9-0.sprintlink.net (144.232.14.30)  103.953 ms  103.947 ms  103.920 ms
16  sl-veris4-4-0.sprintlink.net (144.228.188.122)  107.433 ms  107.449 ms  107.412 ms
17  r1.shared-fo.brn1.verisign.com (65.205.32.170)  107.162 ms  107.227 ms  107.204 ms
18  r1.core-fo.brn1.verisign.com (65.205.32.42)  113.043 ms  112.976 ms  113.119 ms
19  r1-a-root.brn1.verisign.com (198.41.3.237)  113.288 ms  113.626 ms  113.947 ms
20  198.41.0.29 (198.41.0.29)  111.637 ms  111.799 ms  112.028 ms
prompt> dig @a.root-servers.net fi. ns


; <<>> DiG 9.3.4 <<>> @a.root-servers.net fi. ns
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59849
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 5

;; QUESTION SECTION:
;fi.                            IN      NS

;; AUTHORITY SECTION:
fi.                     172800  IN      NS      D.fi.
fi.                     172800  IN      NS      A.fi.
fi.                     172800  IN      NS      B.fi.
fi.                     172800  IN      NS      C.fi.

;; ADDITIONAL SECTION:
A.fi.                   172800  IN      A       193.229.4.44
B.fi.                   172800  IN      A       194.146.106.26
C.fi.                   172800  IN      A       128.214.4.29
D.fi.                   172800  IN      A       192.36.144.116
D.fi.                   172800  IN      AAAA    2001:698:9:302::53

;; Query time: 113 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Mon Dec  3 13:38:33 2007
;; MSG SIZE  rcvd: 176
```

For other services, such as http, there is more processing at end system needed so response time may be longer and more variable.

You can use `traceroute` to check how long is the common path to two destination. For example running two traceroutes below on `t.ns.verio.net` and `ns.uu.net` one can notice that up to 16th router the routes are same.

```
 prompt> traceroute T.NS.VERIO.NET
```

[3] Either all message types that results severe hard-to diagnose connectivity problems or just “useless” types (including echo).

```
 removed lines..
14  sl-bb21-msq-10-0.sprintlink.net (144.232.19.29)  100.766 ms
15  sl-bb20-msq-15-0.sprintlink.net (144.232.9.109)  100.439 ms
16  sl-bb20-nyc-11-3.sprintlink.net (144.232.9.102)  101.779 ms
17  sl-gw39-nyc-1-0.sprintlink.net (144.232.13.62)  101.394 ms
 removed lines..
prompt>  traceroute ns.uu.net
 removed lines..
14  sl-bb21-msq-10-0.sprintlink.net (144.232.19.29)  100.850 ms
15  sl-bb20-msq-15-0.sprintlink.net (144.232.9.109)  101.626 ms
16  sl-bb20-nyc-11-3.sprintlink.net (144.232.9.102)  101.845 ms
17  204.255.174.225 (204.255.174.225)  105.584 ms  105.624 ms
 removed lines..
```

3 Passive measurements

Passive measurements can be often very compicated, especially if you want to make in-depth flow-level measurmenents real-time. However, we do only basic measurements using simple text-based tools.

3.1 Tcpdump

The very basic tool for network traces is `tcpdump`[4] that is available for multiple operating system. For simple analysis, you can work with text-based tcpdump output.

Short summary of important options:

-n do not resolve addresses (should be used here everytime as packet trace has addresses changed and thus names are totaly useless and non-informative).

-q provides more condenced output. It is useful here as it prints out tcp and udp user data lengths.

-v provides more information about packets, no useful here as most of information is removed.

-t removes timestamp from lines

-tt prints timestamp as seconds.microsecond. Usefull if you want to do some time-based calculations

-r file read packets from file instead of network device

Tcpdump has an expression language to select packets from network trace. For example expression `udp port 53` selects those packets that have UDP protocol and either source or destination port is 53 (name service). Expression `tcp src port 80 or tcp src port 8080 or tcp src port 8888` selects those tcp packets that have one of typical web server port numbers as source port.

Text output is easy to analyse using `awk` program. With following line (combine to one line) one can find out number of bytes in tcp traffic, number of packets and average packet size.

```
tcpdump -r /p/edu/s-38.180/trace/capture.pcap -n -q  tcp
— awk ’{i+=$7+40;c++}END{print i, c, i/c}’
```

Note that code above assumes that IP+TCP header length is 40 bytes that may not be true because of use of TCP options. If you calculate for other protocols, you need to check those header lengths.

3.2 Timing commands

If you do not want to sit beside computer to run commands at specified times, you can utilise `at` command. Easiest way is to create command script that should be run at specified times:

```
date >> log
ping -c 4 m.root-servers.net >> log
dig @m.root-servers.net fi. ns | grep time: >> log
```

[4] There is no system-installed tcpdump, you can use one in `/p/edu/s-38.180/Linux/bin`. Append the directory in question to your `$PATH` variable. An graphical tool (`ethereal`) is also installed in the same directory. It may provide more user-friendly interface for initial analysis. Because of anonymisation of trace data, the Ethereal program gives wrong ansfers for some queries.

If you save framgment above file as `ping-dig`, you can then give following commands to run script after one, two, three, and four hours:

```
prompt>  at -f ping-dig now + 1 hour
prompt>  at -f ping-dig now + 2 hour
prompt>  at -f ping-dig now + 3 hour
prompt>  at -f ping-dig now + 4 hour
```

Output is recoded to file `log`. If there are errors, they will be mailed to you. You can forward emails to your own email account by creating `.forward` file:

```
prompt>  echo your.email@example.com ¿ .forward
```

4 Tasks

Answers should be returned by Thursday 2007-12-20 by 15 hours (but recommend returning as soon as possible).

4.1 Check latency for you home ccTLD name servers

To randomly distribute ccTLDs, there is a program at `/p/edu/s-38.180/trace/cc` you need to run to find out ccTLD assigned to you.[5] Compare rtt times measured using ICMP messages and DNS queries. If some of those does not respond to ICMP messages, use traceroute to locate last responding router and compare ping rtts to DNS queries (select some domain you know and ask its name servers).

Select three of those name servers and check round-trip time with dns query (as above) and ICMP messages. Repeat test at least three times with more than one hour intervals. Make some conclusions about stability of network delay. Was some of those servers different from the others?

4.2 Measurement platform studies

AMP do not anymore have web interface, but a selected set of measurements for one day are available at /p/edu/s-38.180/trace/amp. There are three kinds of data in each directory. The *-106.8.31 files contain ICMP echo roundtrip times: the first column is seconds since midnight (PDT time) and the second column is the RTT time in milliseconds or loss in case of loss. The other files of interest are *-106.8.31.trace files that contain traceroute run about every 10 minutes. You need to compare RTT and traceroute between amp-hutf and amp-tenet-s_af. Do you identify any anomalies? Are the problems specific for either TKK or Tenet?

4.3 Traffic analysis

There is an anonymised network trace in `/p/edu/s-38.180/trace/capture.pcap` (information about the file will be in `capture.txt` file). Based on that data, answer following:

1. What is proportition of each transport-layer protocol (tcp, udp, icmp, igmp) by packets and by bytes.

2. For each transport-layer protocol, give distribution of packet sizes.

3. Select 5 different TCP application (well-known port numbers) and report proportition for each of those[6]

4. Report top-10 hosts by packet count (senders) and by bytes

5. How much there are ACK-only TCP segments (i.e. packets that do not carry TCP data)

References

[5] The program checks that there is at least 5 name servers for given ccTLD and tries another one until it finds one with enough name servers.

[6] You can check from `/etc/services` for applications corresponding to port numbers.