A quick tutorial on using `tshark`

Ross Maloney

January 24, 2017

The network sniffing program `tshark` is the terminal oriented version of the GUI version `wireshark`. This GUI version was initially called `ethereal`. Wikipedia states in May 2006 `ethereal` was renamed `wireshark` due to trademark issues.

In this tutorial, `tshark` was compiled from version 2.2.1 of the `wireshark` open source distribution. The configuration used for this compile was:

```
./configure --disable-wireshark --with-extcap=no
```

followed by standard `make` and `sudo make install` terminal commands. But if `tshark` is available on the computer you are using, then use it for a new version is not necessary for this tutorial.

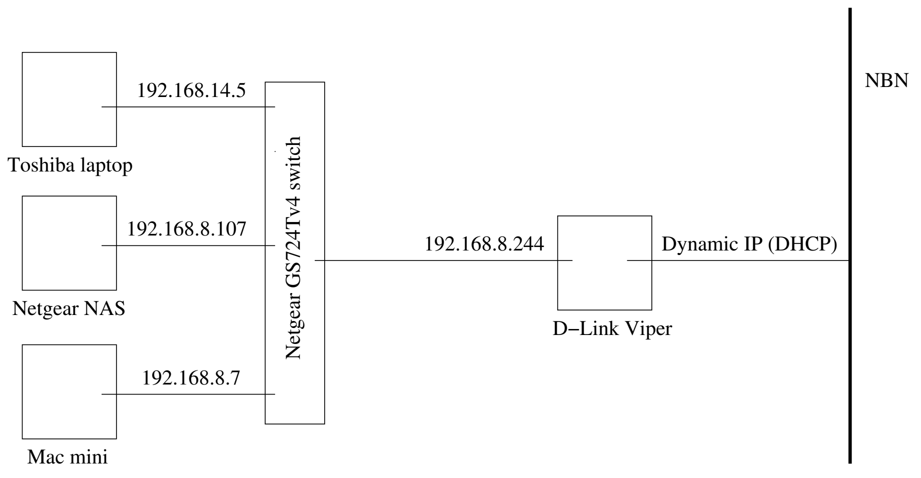

Figure 1: Network used in the examples

Figure 1 shows the network used in this tutorial. The Toshiba laptop and Mac mini each operated under the Linux operating system. The D-Link Viper was a model DSL 2900AL connected as a *Residential Gateway* to a RJ45 connection port on a NBN connection box and thus out to the Internet. Ethernet port 2 of the Viper was connected to the Netgear switch. The Netgear NAS is used later in this tutorial.

A simple network maybe. But appropriate for this tutorial. Figure 1 also demonstrates that a diagram of the network being investigated is helpful.

1 Getting started

`tshark` operates in one of two modes. It can either capture network activity or examine previously captured data. Before considering each of those activities, some preliminaries first.

1.1 Insufficient permission

The command:

```
tshark
```

gave the output:

```
Capturing on 'enp2s0f0'
tshark: The capturing session could not be initiated on interface 'enp2s0f0'
(You don't have permission to capture on that device.  Please check to make
sure you have suffieient permissions, and that you have the proper interface
or pipe specified.
0 packet captured
```

This indicates to be able to read packets on the network, `tshark` should be run as root. Alternatively, the `tshark` command could be prefixed by `sudo`.

1.2 Showing all of the computer's network interfaces

To look at the network interfaces on the current computer, the command:

```
sudo tshark -D
```

when executed on the *MAC mini* computer of Figure 1 gave the output:

```
Running as user "root" and group "root". This could be dangerous.
1. enp2s0f0
2. any
3. lo (loopback)
4. nflog
5. nfqueue
6. usemon1
7. usbmon2
8. usbmon3
9. usbmon4
```

The first entry indicates an ethernet interace (en), on bus 2 (p2), slot 0 (s0), and function 0 (f0). This is in line with the new systemd/udev naming convention replacing the eth0, wlan0, etc. convention previously used. The `any` indicates all network devices connected, while `lo` is the standard loopback device used with each network interface. Both `nflog` and `nfqueue` are part of the `netfilter.org` project for developing network hooks within the Linux kernel. For this computer there are also four usb network interfaces available.

Only the first of those devices will be used in the following.

1.3 All the current network activity

The command:

```
sudo tshark
```

will output on the terminal screen the current activity on the network to which the computer executing the command is connected. In most instances it is a repidly changing tabulation too difficult to comprehend. At least network activity is shown.

This is the default `tshark` behaviour.

2 Saving to a file and then reading from it

There are three ways of assisting the comprehension of network activity which `tshark` can provide. One is to watch all activity on the network as it happens. Another is to focus on certain activity or activities by capturing those activity. This is the subject of Section 3.2. The remaining way is to store all activity, or a selection of activities, in a file and later read it for analysis.

The command:

```
sudo tshark -w mycaptures.pcap
```

will capture all packets on the network, storing then in the file here called `mycaptures.pcap` until it is told to stop by giving a Ctrl-C from the keyboard. A better command would be to nominate the number of packets to capture. For example, the command:

```
sudo tshark -c 500 -w mycaptures.pcap
```

takes the next 500 packets and stores them in the `mycaptures.pcap` file.

Both these commands give the terminal output:

```
capturing on 'enp2s0f0'
```

with a number on the following line which indicates the current number of packets captured and stored in the file. The file `mycaptures.pcap` is of binary content.

By contrast, the command:

```
sudo tshark -r mycaptures.pcap
```

reads the file containing the packet captures. If there are no switches refining how the file is to be processed (as in the above command), then the contents of the file containing the packet capture data is printed on the terminal.

3 Modifying default `tshark` behaviour

The default behaviour of `tshark` can be adapted to suit purpose. These changes result from switches specify filters on the command line which are applied to the execution of `tshark`. There are switches are directed at pachet capture, what is displayed on the terminal screen while packet capture is preceeding, and what is printed on the screen when the captured data is being analysed.

3.1 Basic rules

Some basic rules can assist in using `tshark`. These rules are:

1. No switches on command line gives default behaviour
2. If it is not captured, then the packet cannot be examined
3. A packet is captured in its entirity
4. Parts of a captured packet can be selected for displayed

The contents of the next three sections expand upon these basic rules.

3.2 Fetch filters

If not told otherwise, `tshark` fetches all packets on the network.

A reduction in packets fetched, and either displayed on the terminal or stored in a capture file, can be reduced by giving fetch filters on the `tshark` command line. These filters are given as `-f` switches. Table 1 contains some of those fetch switch options. As shown in Table 1 such parameters can be combined and qualified by using the three logical operators below the line at the base of the table. Each option is required to be included in double quotes around associated address or number.

An example of a `tshark` command using fetch filtering is:

```
sudo tshark -f "net 192.168.8.0/24"
```

or written another way:

```
sudo tshark -f "net 192.168.8.0 mask 255.255.255.0"
```

both of which fetch and display on the terminal only network packets from, or to, all network addresses on network `192.168.8.0`.

At a low level, the activity along a network such as in Figure 1 is by ethernet packets. All such packets do not carry TCP/IP type data although they can be of interest. A `tshark` such as:

```
sudo tshark -f "multicast or broadcast"
```

fetches such auxiliary worker packets. By re-running this command with `multicast` and `broadcast` used separately as the `-f` fetch switch, the relative proportion of these packets on the network can be deduced using `tshark`. Alternately, then command:

```
sudo tshark -f "multicast or broadcast" -w workers.pcap
```

could be used to fetch both types of packets from the network and have them stored in a file for subsequence analysis. With the fetch being written into the file, the fetches are not shown on the terminal.

Table 1: A selection of fetch (-f switch) filters

Specifier	Description
host	4 decimal digit dot separated IP address
net	a range of 4 decimal digit dot separate IP address
src net	from a range of IP addresses
dst	to a range of IP addresses
mask	to apply to IP address
arp	Address Resolution Protocol
ether proto	ethernet type field
ether dst	ethernet MAC address of destination
broadcast	broadcast message across the network
multicast	ethernet multicast packet
tcp portrange	hyphen (-) separated range of TCP port numbers
dst port	TCP destination port number
tcp port	TCP port number
ip	all IPv4 traffic
pppoes	all PPPoE traffic
vlan	all VLAN traffic
port	TCP port number
not	NOT the following
and	logical AND of the two adjacent parameters
or	logical OR of the two adjacent parameters

The fetch switch is only for use with a network capture. I does not work when fetched network data is read from a file. For example, the command:

```
sudo tshark -f "multicast or broadcast" -r workers.pcap
```

will generate an error from `tshark`.

3.3 Yank filters

A yank filter is indicated by a `-Y` switch. A selection of those available are listed in Table 2. These switches give information on the terminal about network activity current on the network being monitored or from a capture file read into `tshark` using a `-r` switch. An example command is:

```
sudo tshark -Y "ip.addr == 192.168.8.244"
```

which displays packets on the network being monitored which are addressed to, or are coming from, IP address `192.168.8.244`. The command:

```
sudo tshark -Y "ip.addr == 192.168.8.244" -r mycaps.pcap
```

would perform the same function on the input stream being read from the fetch file `mycaps.pcap`.

Table 2: A selection of display (-Y switch) filters

Field	Description
frame.time	
frame.time_relative	Relative packet time stamp
frame.len	Length of the packet
frame.protocols	Protocol to which the packet belongs
frame.number	Packet number in the data stream
eth.addr	6 hex digit colon separated ethernet MAC address
eth.dst	6 hex digit colon separated destination MAC address
ip.addr	4 decimal digit dot separated IP address
ip.src	Sender's IP address
ip.dst	Receiver's IP address
ip.len	length of the IP packet
tcp.srcport	TCP source port
tcp.port	TCP port number
tcp.dstport	TCP destination port
udp.port	UDP port number
col.Info	Received packet's content
http.response.code	HTTP response code number
&&	logical AND
||	logical OR
$>$	greater than
$\geq$	greater or equal
$<$	less than
$\leq$	less than or equal
==	equal to
!	logical NOT

The yank switch can be combined with the fetch switch such as in the command:

```
sudo tshark -f "host 192.168.8.244" -Y "ip.dst"
```

which fetches all packets going to, or from, IP `192.168.8.244` and shows on the terminal the IP address of the destination of that packet.

A rule follows from this combination. When the yank (`-Y`) switch is used allow (without the `-f` switch), then it acts upon the input stream going into `tshark`. With the `-f` switch it acts upon what the `-f` switch produces.

The yank (`-Y`) switch cannot be used with a write (`-w`) on a `tshark` command line. This apply with or without the presence of a fetch (`-f`) switch. However, the yank switch can be used with a fetch (`-f`) switch if no read (`-r`) is present. For example, the command:

```
sudo tshark -f "host 192.168.8.244" -Y "ip.dst" -r mycaps.pcaps
```

will generate an error message from `tshark`. The command:

```
sudo tshark -Y "ip.dst" -w mycaps.pcaps
```

will also generate an error message.

The advantage provided by the yank switch is the increase logic which can be performed in it in comparison with the fetch (`-f`) switch. Table 2 list not only the parameters, but also below the line, the logical operators which can be used. All C language operators can be used, for example `ip.addr != 192.168.8.244` to indicate addresses not equal to `192.168.8.244`. Such expressions can be combined using AND (`&&`) and OR (`||`) to build more complex statements. An example of such a command is:

```
sudo tshark -Y "ip.addr != 192.168.8.244 && ip.len < 1500" \
-Y "ip.src == 192.168.14.5"
```

which would display all packets which have both an address other than `192.168.8.244` and a packet length less than 1500 bytes. Also packets coming from IP address `192.168.14.5`.

As these examples show, like with the `-f` parameters, the `-Y` paramteres are enclosed in double quote marks. Also, more than one `-Y` switch can be applied to a single `tshark` command.

3.4 Analysis

There are three switches used to analysis network packets: `-V`, `-O`, and `-T`.

The `-V` switch makes `tshark` verbose, showing details of each packet including their respective frame number, protocol field, etc. Can example of the command is:

```
sudo tshark -V
```

which would provide details of the current packets on the network.

The `-O` provides detail similar to the `-V` switch with the advantage of specifying the protocol of interest. An example command is:

```
sudo tshark -O icmp
```

which provides detail of the icmp protocol packets seen. A complete list of protocols available, and their abbreviations for use with `-O` is available from the command:

```
sudo tshark -G protocols | less
```

where `less` is used to handle the large number of selections available.

The -T switch controls the format of the printed output. The options available are listed in Table 3. If the `fields` options is used, then there needs to be one, or more, elaboration `-e` swtiches, each with their own parameter. The `-e` switch parameters are the same as used with the `-Y` switch. However, in the `-e` switch case, no logic operators (below the line) in Table 2 can be used. A complete list of available elaboration (`-e`) switch paramters is available from the command:

```
sudo tshark -G fields | less
```

where `less` is needed to handle the large number of selections available.

Table 3: Some useful format (-T) switch available

Specifier	Description
text	Text one line summary of each packet (the default)
ps	Postscript one line summary of each packet
fields	Content of packet fields specified using `-e` switches

An example command is:

```
sudo tshark -T fields -e frame.number -e ip.addr -e ip.len
```

which prints the IP address and length of each of those IP packets as they occur on the network to which the computer running this command is connected. Alternately, the same result can be obtained from an existing packet capture file by using the command:

```
sudo tshark -T fields -e frame.number -e ip.addr -e ip.len -r mycaptures.pcap
```

Notice the absence of the double quotes around the parameters.

The `-T` switch defining a field can be used with the `-O` switch specifying a protocol. An example of this is:

```
sudo tshark -O http -T fields -e frame.number -e ip.addr -e ip.len -e data
```

4 Packet content and statistics

The `-x` switch is useful to display the complete contents of a captured packet. Because of the amount of data shown, this switch is best left when processing packet captures read from a file. As an example, the command:

```
sudo tshard -r mycaptures.pcap -Y "ip.addr == 192.168.8.244" -x
```

would display the contents of the packets captured go to, or from, address `192.168.8.244` which are contained in the file `mycaptures.pcap`.

Statistical reports can be requested using the `-z` switch. A list of available reports is available from the command:

```
sudo tshark -z help
```

Multiple `-z` switches can be used on the one `tshark` command, with one report specified with each `-z` switch. An example command is:

```
sudo tshark -i enp2s0f0 -z http.tree -z hosts
```

The report/s are produced when the capture stream is terminated.

5 Examples which put it all together

Ther are some combination of parameters for passing to `tshark` via the command line which are useful and others which are not allowed.

5.1 Combining parameters

In the network of Figure 1 the address `192.168.8.244` is the router to the Internet. The MAC mini has the ethernet network interface `enp2s0f0`. The following sequence of `tshark` commands demonstrate how command parameters can be combined and the result that follows.

```
sudo tshark -i enp2s0f0 -f "host 192.168.8.244"
```

The packets going to and from address `192.168.8.244` are matched, captured (but stored nowhere), and shown on terminal display.

```
sudo tshark -i enp2s0f0 -f "host 192.168.8.244" -Y "eth.addr"
```

This command matches the packets going to and from URL `192.168.8.244` which are shown on the terminal, but no ethernet address is shown. The display filter (`-Y "eth.addr"`) is ignored.

```
sudo tshark -i enp2s0f0 -f "host 192.168.8.244" -Y "eth.addr" -w my.pcap
```

gives an error message indicating display filters aren't supported when capturing and saving the captured packets

Another sequence is the command:

```
sudo tshark -i enp2s0f0 -f "host 192.168.8.244" -T fields -e "eth.addr"
```

which captures (but does not store) all packets to or from address `192.168.8.244` and shows on the terminal the ethernet address of the packet containing that address. This is difficult to understand. The command:

```
sudo tshark -i enp2s0f0 -f "host 192.168.8.244" -T fields -e "eth.addr" \
-e "ip.src" -e "ip.dst"
```

by contrast, shows on the terminal the ethernet address of the packet together with the IP address of the source and destination of the packet on the same line for each packet captured. By contrast, the command:

```
sudo tshark -i enp2s0f0 -f "host 192.168.8.244" -T fields -e "eth.addr" \
-e "ip.src" -e "ip.dst -w mycaptures.pcap"
```

prints the same information as the packets are captured but now stores them in the file `mycaptures.pcap`. Say that last command captured, or showed, 12 lines of capture data. Then the command:

```
sudo tshark -r mycatures.pcap
```

shows 12 packets as having been captured in the file, not just the information shown on the terminal when the capturing was occurring. A command such as:

```
sudo tshark -r mycatures.pcap
```

reads the file ands displays a summary of the 12 packets recorded in the capture file specified. The command:

```
sudo tshark -r mycaptures.pcap -Y "frame.number == 6" -x
```

prints the contents of frame number 6 on the terminal. Which the command:

```
sudo tshark -r mycaptures.pcaps -T fields -e "frame.number == 6" -x
```

is an error because the switch `-x` cannot be used with `-T fields`. And the command:

```
sudo tshark -r mycaptures.pcap -T fields -e "frame.number == 6" -e "ip.src"
```

is in error because `-T fields` cannot have a `-e` swtich containing a logical operator (==).

5.2 Practical commands

Where as the command

```
%sudo tshark -i any  -T field -e "ip.src" -e "ip.dst"
sudo tshark -i enp2s0f0
```

issued from the terminal having address `192.168.8.7` displays all packets passed on the network, the command:

```
%sudo tshark -i any  -T field -e "ip.src" -e "ip.dst"
sudo tshark -i enp2s0f0 -f "host 192.168.8.7" -f "not broadcast and not multicast"
```

removes the lower level packets. The interchanges with this `192.169.8.7` address can be more easily seen. Say, from this output, address `203.0.178.192` is seen interacting with `192.168.8.7`. The command:

```
host 203.0.178.192
```

indicates this address is associated with the URL `pop3.iinet.net.au` (the `host` program is a standard TCP program, but not part of `tshark`.

One of the shortest `tshark` commands:

```
sudo tshark -i enp2s0f0 -f "ip"
```

can produce useful output by not showing low level packets passing on the network. Contrast that output with the output from the command:

```
sudo tshark -i enp2s0f0
```

The command:

```
sudo tshark -i enp2s0f0 -f "host 192.168.14.5"
```

monitors the network packets from and to address `192.168.14.5`. If on the device with that address, the command:

```
ping 192.168.8.7
```

is given, the request and response packets are seen through `tshark`.