University of Victoria

CENG 460

Computer Communication Networks

Lab 1 - Introduction to WireShark and Layered Protocol

Instructor:
Dr. Lin Cai

Teaching Assistant:
Amir Andaliby

Tyler Stephen V00812021
A01 - B04

February 6, 2016

1 Introduction

This lab will demonstrate the multi-layered design of a communications network by inspecting actual ethernet traffic. Common Unix tools will be used to investigate host machine configuration and make simple calls to Internet addresses. WireShark, a network analysis tool, is used to inspect raw traffic and expose information contained in each traffic layer.

2 Procedure

2.1 Getting familiar with WireShark

Workstations in ELW A321 run a version of Scientific Linux and are connected to the University of Victoria LAN. WireShark, a network analysis tool, is installed on the workstations and is able to intercept all traffic over a workstation's ethernet connection. This portion of the lab generates simple traffic in order to become familiar with the WireShark's interface.

WireShark is started and set to listen to traffic on the workstation's ethernet connection. A webpage is requested via the console using `wget`. This minimizes additional callback traffic generated by the browser version of most webpages and makes it easier to associate captured traffic to console commands.

2.2 Networking tools

Unix has several useful tools for investigating and configuring network interfaces:

- `ping` will generate traffic to send to a destination and display statistics for the transmission times
- `netstat` displays information about the local network interfaces and can display local network statistics
- `ifconfig` configures and controls a machine's network interface.

Each of these tools will be used to make conclusions about network traffic. See Section 3.2 for examples of the output.

2.3 Layered protocol

WireShark is capable of displaying the data in each layer of network protocol in a frame. An HTTP request / response conversation is recorded in *lab1-wget-trace.pacp* and is examined in this section. Information from the headers in each layer is used to determine the data length. Headers also indicate the type of parent protocol.

3 Discussion

3.1 Running WireShark

List at least three different protocols that appear in an unfiltered capture.

After invoking `wget http://www.google.ca` at the command line, WireShark fills with traffic. The following protocols were observed:

- DNS
- HTTP
- TCP
- TLSv1.2

How long did it take from the HTTP GET message being sent to the HTTP OK reply being received?

This is determined by comparing the values of the Time field in the Captured Packets window of WireShark. The value of Time corresponds to the elapsed time the packet was recorded since WireShark began its capture. For this capture, GET was sent at +7.271199 ms and OK was received at +56.242508 ms. Thus, the elapsed time is 48.971318 ms.

3.2 Network tools

How many Ethernet interfaces are in your computer, how to determine it?

The output of the `ifconfig` command indicates that this computer has 1 network interface, `eth0`. The loopback interface `lo` is a virtual network interface and is only used for self traffic.

```
~ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:13:72:99:9D:C3
          inet addr:192.168.109.17  Bcast:192.168.109.255  Mask:255.255.255.0
          inet6 addr: fe80::213:72ff:fe99:9dc3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:392233 errors:0 dropped:0 overruns:0 frame:0
          TX packets:76107 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:386553389 (368.6 MiB)  TX bytes:6078867 (5.7 MiB)
          Interrupt:16

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8330 (8.1 KiB)  TX bytes:8330 (8.1 KiB)
```

How to turn down/up an Ethernet interface?

Use the command `ifconfig <interface> up|down`. For example, to turn up the ethernet interface invoke `ifconfig eth0 up`.

Ping 10 packets to two websites. Compare the statistic results (packet loss, average round-trip time)

```
~ ping -c 10 www.google.com
PING www.google.com (173.194.33.180) 56(84) bytes of data.
64 bytes from sea09s18-in-f20.1e100.net (173.194.33.180): icmp_seq=1 ttl=51 time
    =6.28 ms
<<removed>>

--- www.google.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 6.140/6.193/6.310/0.090 ms
```

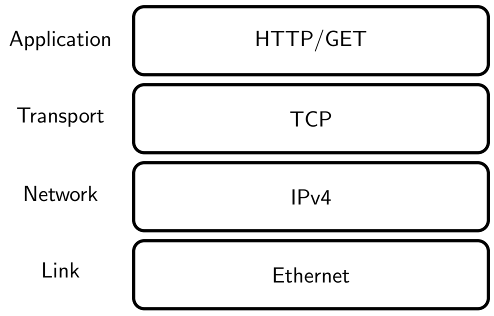
```
~ ping -c 10 www.uvic.ca
PING www.uvic.ca (142.104.197.120) 56(84) bytes of data.
64 bytes from wwwlbserver.uvic.ca (142.104.197.120): icmp_seq=1 ttl=248 time=0.510
    ms
<<removed>>

--- www.uvic.ca ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9001ms
rtt min/avg/max/mdev = 0.408/0.457/0.513/0.033 ms
```

Neither ping resulted in any packet loss. The ping to `www.uvic.ca` had a faster round trip time than the ping to `www.google.com`. This is expected because, presumably, the call to `www.uvic.ca` does not have to leave the LAN whereas the call to `www.google.com` must travel through the internet to reach the nearest Google server. This could be verified by executing `tracert` to determine the packet travel route.

3.3 Layered protocol

Draw the structure of a HTTP GET packet.



In the provided trace (lab1-wget-trace.pacp), calculate the average overhead of all the packets from the server to the client (in percentage).

First, it is necessary to determine the IPs of the server and client. Clients will initiate a GET request and servers will give the response. The HTTP GET request is sent from `142.104.81.181` to `74.125.129.94`. Therefore, the filter

```
ip.src==74.125.129.94 and ip.dst==142.104.81.181
```

will isolate traffic from the server to the client.

Next, the frame length and data length for each frame are determined from WireShark. The Capture window has a column, Length, which gives the frame length. The TCP header `Len` gives the data length. The difference between the two lengths is the length of the header. These values are recorded, *manually*, for each frame.

The average overhead is given by:

$$\% \text{ Overhead} = \frac{\sum(\text{frame length} - \text{data length})}{\sum \text{frame length}} \times 100$$

and yeilds a 6.78% overhead for the recorded traffic.

Which Ethernet header field tells the next higher layer protocol is IP? What value is it used?

The header `Type: IP (0x0800)` indicates that the network layer is IP.

Which IP header field tells the next higher layer protocol is TCP? What value is it used?

The header `Protocol: TCP (6)` indicates that the transport layer is TCP.

4 Conclusion

Retrieving a webpage with `wget` demonstrated that several protocols are necessary to retrieve HTTP content. The high volume of frame traffic was exposed by WireShark. The OSI model effectively obscures the bulk of this communication via protocols which communicate layer-to-layer.

`netstat` and `ifconfig` can be used to determine the network configuration of a Unix machine. `ping` displays the transmission success and round trip time statistics from a machine to a remote address.

WireShark was able to display the header information from multiple protocols in *lab1-wget-trace.pcap*. The headers contained information about the type of the parent layer. Headers made up a total of 6.78% of the data from the server to the client.

5 Feedback

1. Use `tracert` to examine the routes traveled during the `ping` comparison in Section 3.2. Does this explain the difference in round trip times? Compare `.ca`, `.com`, `.co.uk` and `.cn` routes.

2. Write a script to compute traffic overhead.