13. Wireshark

i. Starting and Packet Capture Using Wire shark
ii. Viewing Captured Traffic
iii. Analysis and Statistics & Filters.

Wireshark is an open-source packet analyzer, It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a **sniffer, network protocol analyzer, and network analyzer**. It is also used by network security engineers to examine security problems.

Wireshark is a free to use application which is used to apprehend the data back and forth. It is often called as a free packet sniffer computer application. It puts the network card into an unselective mode, i.e., to accept all the packets which it receives.

Installing Wireshark:

- Open a web browser.
- Navigate to http://www.wireshark.org.
- Select Download Wireshark.
- Double-click on the file to open it.
- If you see a User Account Control dialog box, select Yes to allow the program to make changes to this computer.
- Select Next > to start the Setup Wizard.
- Review the license agreement. If you agree, select I Agree to continue.
- Select Next > to accept the default components.
- Select the shortcuts you would like to have created. Leave the file extensions selected. Select Next > to continue.
- Select Next > to accept the default install location.
- Select Install to begin installation.
- Select Next > to install WinPcap.
- Select Next > to start the Setup Wizard.
- Review the license agreement. If you agree, select I Agree to continue.
- Select Install to begin installation.
- Select Finish to complete the installation of WinPcap.
- Select Next > to continue with the installation of Wireshark.
- Select Finish to complete the installation of Wireshark.

i. Starting and Packet Capture Using Wire shark

Double-click the Wireshark icon, which is located on the desktop.

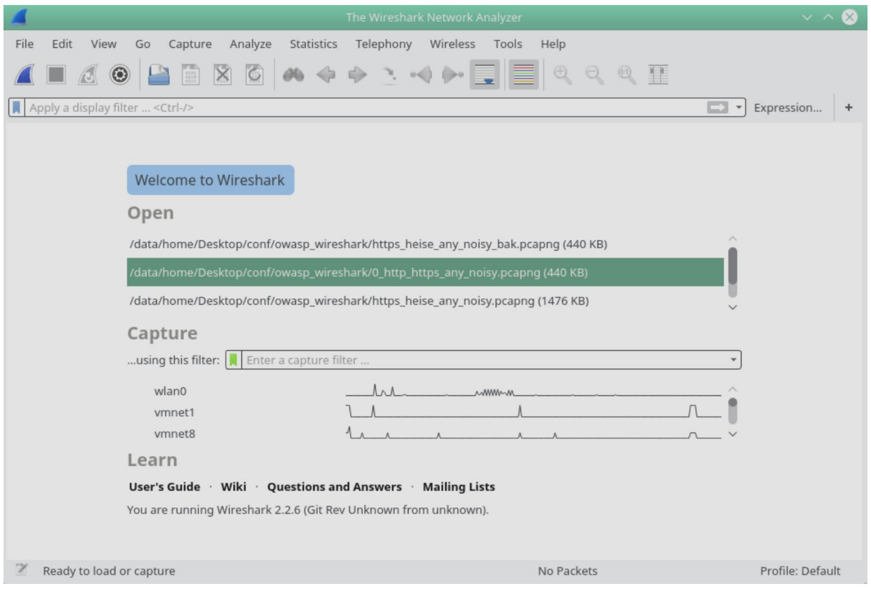
To start capturing

•Select a network interface

•Click on the blue shark fin button / press Ctrl + E

To stop capturing st capturing

•Click on the red stop button / press Ctrl + E



The Wireshark Network Analyzer

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression... +

Welcome to Wireshark

Open

/data/home/Desktop/conf/owasp_wireshark/https_heise_any_noisy_bak.pcapng (440 KB)
/data/home/Desktop/conf/owasp_wireshark/0_http_https_any_noisy.pcapng (440 KB)
/data/home/Desktop/conf/owasp_wireshark/https_heise_any_noisy.pcapng (1476 KB)

Capture

...using this filter: Enter a capture filter ...

wlan0
vmnet1
vmnet8

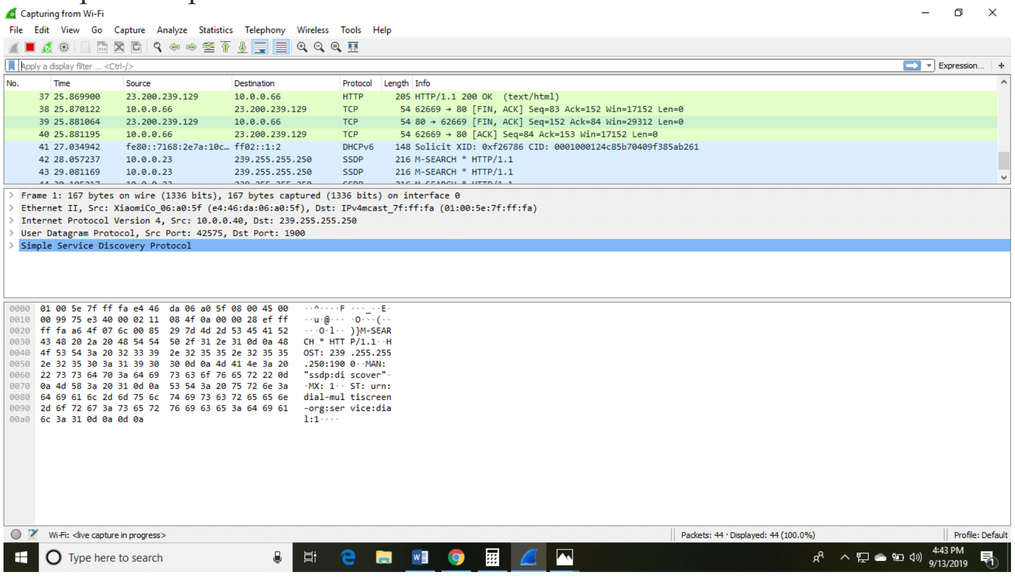
Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 2.2.6 (Git Rev Unknown from unknown).

Ready to load or capture No Packets Profile: Default

The Wireshark software window is shown above, and all the processes on the network are carried within this screen only. The options given on the list are the Interface list options. The number of interface options will be present. Selection of any option will determine all the traffic. **For example,** from the above fig. select the Wi-Fi option. After this, a new window opens up, which will show all the current traffic on the network. Below is the image which tells us about the live capture of packets and our Wireshark will look like:



Capturing from Wi-Fi

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/> Expression... +

```
No.  Time       Source              Destination      Protocol Length Info
  37 25.869900  23.200.239.129      10.0.0.66        HTTP      205 HTTP/1.1 200 OK  (text/html)
  38 25.870122  10.0.0.66           23.200.239.129   TCP        54 62669 → 80 [FIN, ACK] Seq=83 Ack=152 Win=17152 Len=0
  39 25.881064  23.200.239.129      10.0.0.66        TCP        54 80 → 62669 [FIN, ACK] Seq=152 Ack=84 Win=29312 Len=0
  40 25.881195  10.0.0.66           23.200.239.129   TCP        54 62669 → 80 [ACK] Seq=84 Ack=153 Win=17152 Len=0
  41 27.034942  fe80::7168:2e7a:10c… ff02::1:2       DHCPv6    148 Solicit XID: 0xf26786 CID: 0001000124c85b70409f385ab261
  42 28.057237  10.0.0.23           239.255.255.250  SSDP      216 M-SEARCH * HTTP/1.1
  43 29.081169  10.0.0.23           239.255.255.250  SSDP      216 M-SEARCH * HTTP/1.1
```

```
> Frame 1: 167 bytes on wire (1336 bits), 167 bytes captured (1336 bits) on interface 0
> Ethernet II, Src: XiaomiCo_06:a0:5f (e4:46:da:06:a0:5f), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
> Internet Protocol Version 4, Src: 10.0.0.40, Dst: 239.255.255.250
> User Datagram Protocol, Src Port: 42575, Dst Port: 1900
> Simple Service Discovery Protocol
```

```
0000  01 00 5e 7f ff fa e4 46  da 06 a0 5f 08 00 45 00   ··^····F ···_··E·
0010  00 99 75 e3 40 00 02 11  08 4f 0a 00 00 28 ef ff   ··u·@··· ·O···(··
0020  ff fa a6 4f 07 6c 00 85  29 7d 4d 2d 53 45 41 52   ···O·l·· )}M-SEAR
0030  43 48 20 2a 20 48 54 54  50 2f 31 2e 31 0d 0a 48   CH * HTT P/1.1··H
0040  4f 53 54 3a 20 32 33 39  2e 32 35 35 2e 32 35 35   OST: 239 .255.255
0050  2e 32 35 30 3a 31 39 30  30 0d 0a 4d 41 4e 3a 20   .250:190 0··MAN: 
0060  22 73 73 64 70 3a 64 69  73 63 6f 76 65 72 22 0d   "ssdp:di scover"·
0070  0a 4d 58 3a 20 31 0d 0a  53 54 3a 20 75 72 6e 3a   ·MX: 1·· ST: urn:
0080  64 69 61 6c 2d 6d 75 6c  74 69 73 63 72 65 65 6e   dial-mul tiscreen
0090  2d 6f 72 67 3a 73 65 72  76 69 63 65 3a 64 69 61   -org:ser vice:dia
00a0  6c 3a 31 0d 0a 0d 0a                               l:1····
```

Wi-Fi: <live capture in progress> Packets: 44 · Displayed: 44 (100.0%) Profile: Default

Type here to search 4:43 PM 9/13/2019

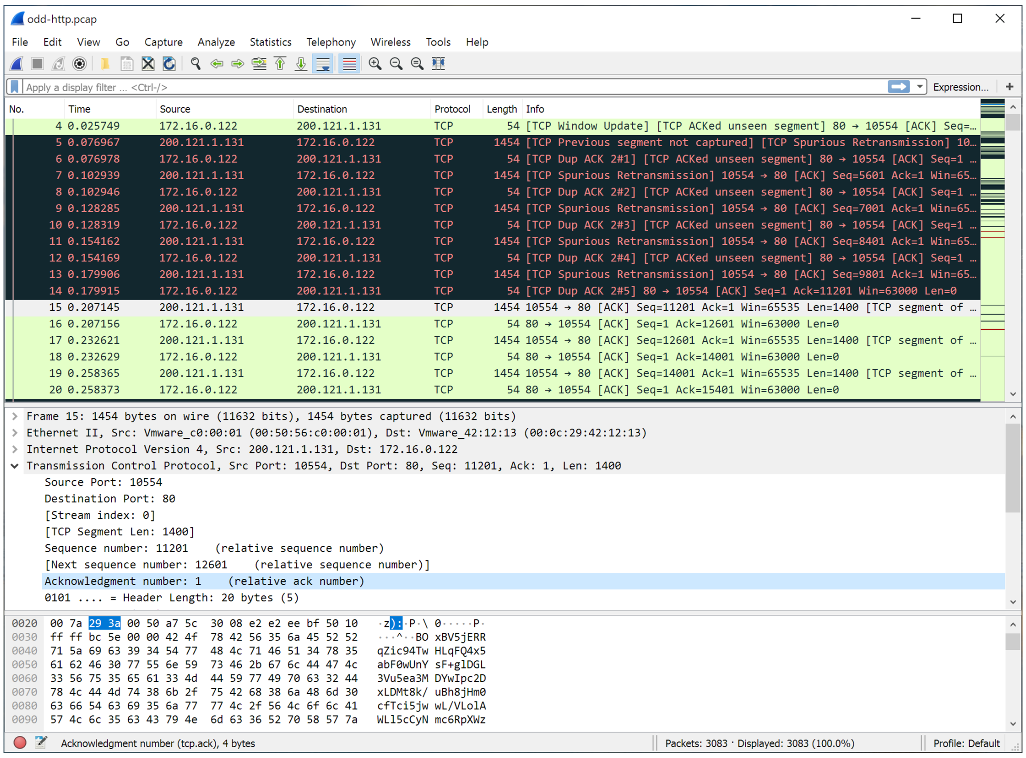
The screen/interface of the Wireshark is divided into five parts:

- First part contains a menu bar and the options displayed below it. This part is at the top of the window. File and the capture menus options are commonly used in Wireshark. The capture menu allows to start the capturing process. And the File menu is used to open and save a capture file.
- The second part is the packet listing window. It determines the packet flow or the captured packets in the traffic. It includes the packet number, time, source, destination, protocol, length, and info. We can sort the packet list by clicking on the column name.
- Next comes the packet header- detailed window. It contains detailed information about the components of the packets. The protocol info can also be expanded or minimized according to the information required.
- The bottom window called the packet contents window, which displays the content in ASCII and hexadecimal format.

ii. Viewing Captured Traffic

Once you have captured some packets or you have opened a previously saved capture file, you can view the packets that are displayed in the packet list pane by simply clicking on a packet in the packet list pane, which will bring up the selected packet in the tree view and byte view panes.

You can then expand any part of the tree to view detailed information about each protocol in each packet. Clicking on an item in the tree will highlight the corresponding bytes in the byte view. An example with a TCP packet selected is shown in below. It also has the Acknowledgment number in the TCP header selected, which shows up in the byte view as the selected bytes.


odd-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...

```
No.  Time      Source         Destination    Protocol Length Info
   4 0.025749  172.16.0.122   200.121.1.131  TCP        54 [TCP Window Update] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=…
   5 0.076967  200.121.1.131  172.16.0.122   TCP      1454 [TCP Previous segment not captured] [TCP Spurious Retransmission] 10…
   6 0.076978  172.16.0.122   200.121.1.131  TCP        54 [TCP Dup ACK 2#1] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 …
   7 0.102939  200.121.1.131  172.16.0.122   TCP      1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=5601 Ack=1 Win=65…
   8 0.102946  172.16.0.122   200.121.1.131  TCP        54 [TCP Dup ACK 2#2] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 …
   9 0.128285  200.121.1.131  172.16.0.122   TCP      1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=7001 Ack=1 Win=65…
  10 0.128319  172.16.0.122   200.121.1.131  TCP        54 [TCP Dup ACK 2#3] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 …
  11 0.154162  200.121.1.131  172.16.0.122   TCP      1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=8401 Ack=1 Win=65…
  12 0.154169  172.16.0.122   200.121.1.131  TCP        54 [TCP Dup ACK 2#4] [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 …
  13 0.179906  200.121.1.131  172.16.0.122   TCP      1454 [TCP Spurious Retransmission] 10554 → 80 [ACK] Seq=9801 Ack=1 Win=65…
  14 0.179915  172.16.0.122   200.121.1.131  TCP        54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
  15 0.207145  200.121.1.131  172.16.0.122   TCP      1454 10554 → 80 [ACK] Seq=11201 Ack=1 Win=65535 Len=1400 [TCP segment of …
  16 0.207156  172.16.0.122   200.121.1.131  TCP        54 80 → 10554 [ACK] Seq=1 Ack=12601 Win=63000 Len=0
  17 0.232621  200.121.1.131  172.16.0.122   TCP      1454 10554 → 80 [ACK] Seq=12601 Ack=1 Win=65535 Len=1400 [TCP segment of …
  18 0.232629  172.16.0.122   200.121.1.131  TCP        54 80 → 10554 [ACK] Seq=1 Ack=14001 Win=63000 Len=0
  19 0.258365  200.121.1.131  172.16.0.122   TCP      1454 10554 → 80 [ACK] Seq=14001 Ack=1 Win=65535 Len=1400 [TCP segment of …
  20 0.258373  172.16.0.122   200.121.1.131  TCP        54 80 → 10554 [ACK] Seq=1 Ack=15401 Win=63000 Len=0
```

```
> Frame 15: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits)
> Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: Vmware_42:12:13 (00:0c:29:42:12:13)
> Internet Protocol Version 4, Src: 200.121.1.131, Dst: 172.16.0.122
v Transmission Control Protocol, Src Port: 10554, Dst Port: 80, Seq: 11201, Ack: 1, Len: 1400
    Source Port: 10554
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 1400]
    Sequence number: 11201    (relative sequence number)
    [Next sequence number: 12601    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
```

```
0020  00 7a 29 3a 00 50 a7 5c  30 08 e2 e2 ee bf 50 10   ·z):·P·\ 0·····P·
0030  ff ff bc 5e 00 00 42 4f  78 42 56 35 6a 45 52 52   ···^··BO xBV5jERR
0040  71 5a 69 63 39 34 54 77  48 4c 71 46 51 34 78 35   qZic94Tw HLqFQ4x5
0050  61 62 46 30 77 55 6e 59  73 46 2b 67 6c 44 47 4c   abF0wUnY sF+glDGL
0060  33 56 75 35 65 61 33 4d  44 59 77 49 70 63 32 44   3Vu5ea3M DYwIpc2D
0070  78 4c 44 4d 74 38 6b 2f  75 42 68 38 6a 48 6d 30   xLDMt8k/ uBh8jHm0
0080  63 66 54 63 69 35 6a 77  77 4c 2f 56 4c 6f 6c 41   cfTci5jw wL/VLolA
0090  57 4c 6c 35 63 43 79 4e  6d 63 36 52 70 58 57 7a   WLl5cCyN mc6RpXWz
```

```
Acknowledgment number (tcp.ack), 4 bytes    Packets: 3083 · Displayed: 3083 (100.0%)    Profile: Default
```

iii. Analysis and Statistics & Filters.

Filters in Wireshark:

The filter block below the menu bar, from where a large amount of data can be filtered. For example, if we apply a filter for HTTP, only the interfaces with the HTTP will be listed.

If you want to filter according to the source, right-click on the source you want to filter and select 'Apply as Filter' and choose '...and filter.'

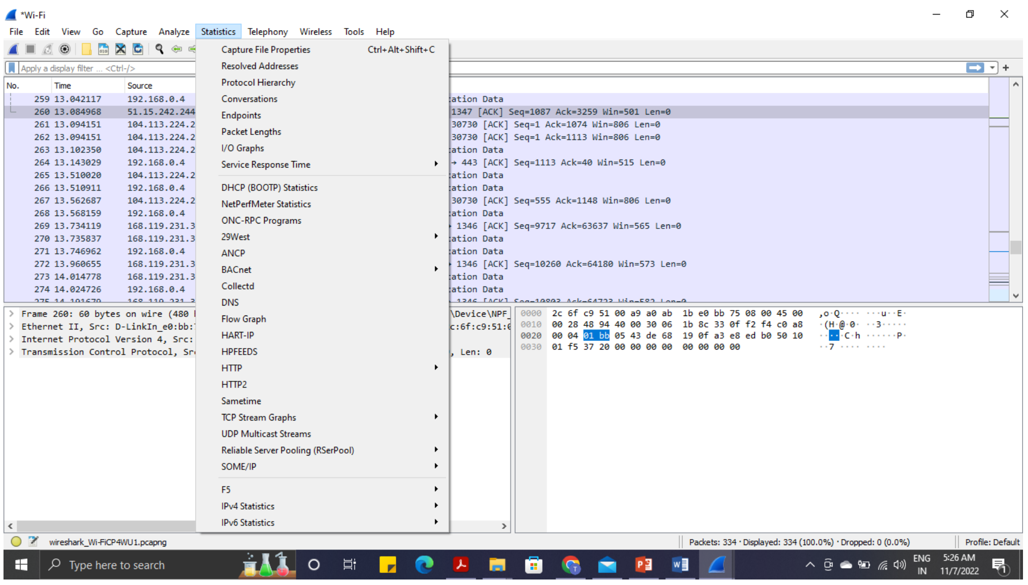
The list of filters used in Wireshark:

Filters	**Description**
ip.addr Example- ip.addr==10.0.10.142 ip.src ip.dst	It is used to specify the IP address as the source or the destination. This example will filter based on this IP address as a source and a destination. If we want for a particular source or destination then, It is used for the source filter. It is used for the destination.

protocol Example- dns or http 'Dns and http' is never used.	This command filters based on the protocol. It requires the packet to be either dns protocol or http protocol and will display the traffic based on this. We would not use the command 'dns and http' because it requires the packet to be both, dns as well as http, which is impossible.
tcp.port Example: tcp.port==443	It sets filter based on the specific port number. It will filter all the packets with this port number.
4. udp.port	It is same as tcp.port. Instead, udp is used.
tcp.analysis.flags example is shown in **fig(5)**.	Wireshark can flag TCP problems. This command will only display the issues that Wireshark identifies. Example, packet loss, tcp segment not captured, etc. are some of the problems. It quickly identifies the problem and is widely used.
6.!() For example, !(arp or dns or icmp) This is shown in **fig (6)**.	It is used to filter the list of protocols or applications, in which we are not interested. It will remove arp, dns, and icmp, and only the remaining will be left or it clean the things that may not be helpful.
Select any packet. Right-click on it and select 'Follow' and then select' TCP stream.' Shown in fig. (7).	It is used if you want to work on a single connection on a TCP conversation. Anything related to the single TCP connection will be displayed on the screen.
tcp contains the filter For example- tcp contains Facebook Or udp contains Facebook	It is used to display the packets which contain such words. In this, Facebook word in any packet in this trace file i.e., finding the devices, which are talking to Facebook. This command is useful if you are looking for a username, word, etc.
http.request For the responses or the response code, you can type http.response.code==200	It will display all the http requests in the trace file. You can see all the servers, the client is involved.
tcp.flags.syn==1 This is shown in fig (10). tcp.flags.reset	This will display all the packets with the sync built-in tcp header set to 1. This will show all the packets with tcp resets.

Statistics in Wireshark:
Wireshark provides a wide range of network statistics.These statistics range from general information about the loaded capture file (like the number of captured packets), to statistics about specific protocols (e.g. statistics about the number of HTTP requests and responses captured).
General statistics

- Summary about the capture file like: packet counts, captured time period.
- Protocol Hierarchy of the captured packets.
- Conversations e.g. traffic between specific Ethernet/IP/… addresses.
- Endpoints e.g. traffic to and from an Ethernet/IP/… address.
- IO Graphs visualizing the number of packets (or similar) in time.



```
No.  Time        Source
259 13.042117   192.168.0.4        ation Data
260 13.084968   51.15.242.244      1347 [ACK] Seq=1087 Ack=3259 Win=501 Len=0
261 13.094151   104.113.224.2      30730 [ACK] Seq=1 Ack=1074 Win=806 Len=0
262 13.094151   104.113.224.2      30730 [ACK] Seq=1 Ack=1113 Win=806 Len=0
263 13.102350   104.113.224.2      ation Data
264 13.143029   192.168.0.4        → 443 [ACK] Seq=1113 Ack=40 Win=515 Len=0
265 13.510020   104.113.224.2      ation Data
266 13.510911   192.168.0.4        ation Data
267 13.562687   104.113.224.2      30730 [ACK] Seq=555 Ack=1148 Win=806 Len=0
268 13.568159   192.168.0.4        ation Data
269 13.734119   168.119.231.3      1346 [ACK] Seq=9717 Ack=63637 Win=565 Len=0
270 13.735837   168.119.231.3      ation Data
271 13.746962   192.168.0.4        ation Data
272 13.960655   168.119.231.3      1346 [ACK] Seq=10260 Ack=64180 Win=573 Len=0
273 14.014778   168.119.231.3      ation Data
274 14.024726   192.168.0.4        ation Data

> Frame 260: 60 bytes on wire (480            \Device\NPF_
> Ethernet II, Src: D-LinkIn_e0:bb:           c:6f:c9:51:
> Internet Protocol Version 4, Src:
> Transmission Control Protocol, Sr           , Len: 0

0000  2c 6f c9 51 00 a9 a0 ab  1b e0 bb 75 08 00 45 00   ,o·Q···· ···u··E·
0010  00 28 48 94 40 00 30 06  1b 8c 33 0f f2 f4 c0 a8   ·(H·@·0· ··3·····
0020  00 04 01 bb 05 43 de 68  19 0f a3 e8 ed b0 50 10   ······C·h ······P·
0030  01 f5 37 20 00 00 00 00  00 00 00 00               ··7 ···· ····

wireshark_Wi-FiCP4WU1.pcapng    Packets: 334 · Displayed: 334 (100.0%) · Dropped: 0 (0.0%)    Profile: Default
```

The statistics for capture file properties menu

1. From the Statistics menu, choose Capture File Properties:
2. As you can see in the following screenshot, we have the following:
    a) File: Provides file data, such as filename and path, length, and so on

    b) Time: Start time, end time, and duration of capture

    c) Capture: Hardware information for the PC that Wireshark is installed on

    d) Interfaces: Interface information—the interface registry identifier on the left, if capture filter is turned on, interface type and packet size limit

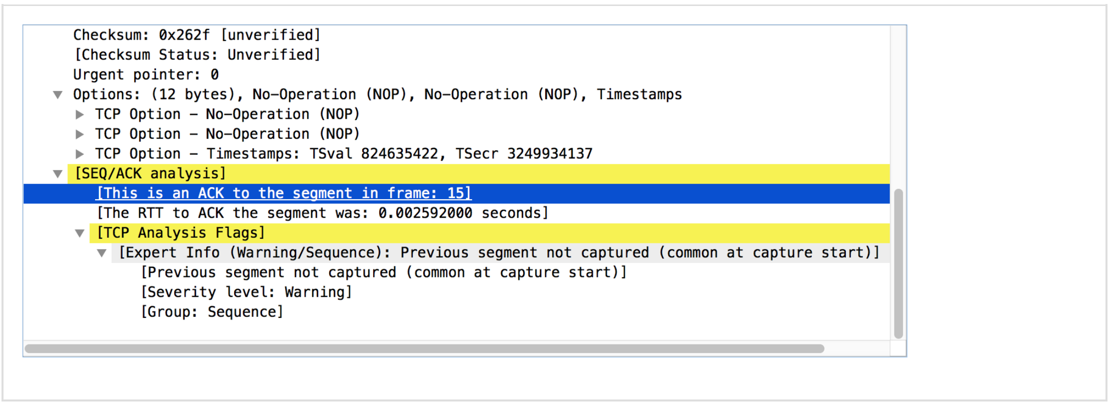
    e) Statistics: General capture statistics, including captured and displayed packets:

Analysis in Wireshark:

TCP Analysis

By default, Wireshark's TCP dissector tracks the state of each TCP session and provides additional information when problems or potential problems are detected. Analysis is done once for each TCP packet when a capture file is first opened. Packets are processed in the order in which they appear in the packet list. You can enable or disable this feature via the "Analyze TCP sequence numbers" TCP dissector preference.

Figure. "TCP Analysis" packet detail items



```
    Checksum: 0x262f [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▼ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    ▶ TCP Option - No-Operation (NOP)
    ▶ TCP Option - No-Operation (NOP)
    ▶ TCP Option - Timestamps: TSval 824635422, TSecr 3249934137
  ▼ [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 15]
      [The RTT to ACK the segment was: 0.002592000 seconds]
    ▼ [TCP Analysis Flags]
      ▼ [Expert Info (Warning/Sequence): Previous segment not captured (common at capture start)]
          [Previous segment not captured (common at capture start)]
          [Severity level: Warning]
          [Group: Sequence]
```

TCP Analysis flags are added to the TCP protocol tree under “SEQ/ACK analysis”. Each flag is described below. Terms such as “next expected sequence number” and “next expected acknowledgment number” refer to the following”:

Next expected sequence number

The last-seen sequence number plus segment length. Set when there are no analysis flags and for zero window probes. This is initially zero and calculated based on the previous packet in the same TCP flow. Note that this may not be the same as the tcp.nxtseq protocol field.

Next expected acknowledgment number

The last-seen sequence number for segments. Set when there are no analysis flags and for zero window probes.

Last-seen acknowledgment number

Always set. Note that this is not the same as the next expected acknowledgment number.

Last-seen acknowledgment number

Always updated for each packet. Note that this is not the same as the next expected acknowledgment number.

TCP ACKed unseen segment

Set when the expected next acknowledgment number is set for the reverse direction and it’s less than the current acknowledgment number.

***TCP Dup ACK* <frame>#<acknowledgment number>**

Set when all of the following are true:

- The segment size is zero.
- The window size is non-zero and hasn’t changed.
- The next expected sequence number and last-seen acknowledgment number are non-zero (i.e., the connection has been established).
- SYN, FIN, and RST are not set.

TCP Fast Retransmission

Set when all of the following are true:

- This is not a keepalive packet.
- In the forward direction, the segment size is greater than zero or the SYN or FIN is set.
- The next expected sequence number is greater than the current sequence number.
- We have more than two duplicate ACKs in the reverse direction.
- The current sequence number equals the next expected acknowledgment number.
- We saw the last acknowledgment less than 20ms ago.

Supersedes “Out-Of-Order” and “Retransmission”.

TCP Keep-Alive

Set when the segment size is zero or one, the current sequence number is one byte less than the next expected sequence number, and none of SYN, FIN, or RST are set.

Supersedes “Fast Retransmission”, “Out-Of-Order”, “Spurious Retransmission”, and “Retransmission”.

TCP Keep-Alive ACK

Set when all of the following are true:

- The segment size is zero.
- The window size is non-zero and hasn’t changed.
- The current sequence number is the same as the next expected sequence number.
- The current acknowledgment number is the same as the last-seen acknowledgment number.
- The most recently seen packet in the reverse direction was a keepalive.
- The packet is not a SYN, FIN, or RST.

Supersedes “Dup ACK” and “ZeroWindowProbeAck”.

TCP Out-Of-Order

Set when all of the following are true:

- This is not a keepalive packet.
- In the forward direction, the segment length is greater than zero or the SYN or FIN is set.
- The next expected sequence number is greater than the current sequence number.
- The next expected sequence number and the next sequence number differ.
- The last segment arrived within the Out-Of-Order RTT threshold. The threshold is either the value shown in the “iRTT” (tcp.analysis.initial_rtt) field under “SEQ/ACK analysis” if it is present, or the default value of 3ms if it is not.

Supersedes “Retransmission”.

TCP Port numbers reused

Set when the SYN flag is set (not SYN+ACK), we have an existing conversation using the same addresses and ports, and the sequence number is different than the existing conversation’s initial sequence number.

TCP Previous segment not captured

Set when the current sequence number is greater than the next expected sequence number.

TCP Spurious Retransmission

Checks for a retransmission based on analysis data in the reverse direction. Set when all of the following are true:

- The SYN or FIN flag is set.
- This is not a keepalive packet.
- The segment length is greater than zero.
- Data for this flow has been acknowledged. That is, the last-seen acknowledgment number has been set.
- The next sequence number is less than or equal to the last-seen acknowledgment number.

Supersedes “Fast Retransmission”, “Out-Of-Order”, and “Retransmission”.

TCP Retransmission

Set when all of the following are true:

- This is not a keepalive packet.
- In the forward direction, the segment length is greater than zero or the SYN or FIN flag is set.
- The next expected sequence number is greater than the current sequence number.

TCP Window Full

Set when the segment size is non-zero, we know the window size in the reverse direction, and our segment size exceeds the window size in the reverse direction.

TCP Window Update

Set when the all of the following are true:

- The segment size is zero.
- The window size is non-zero and not equal to the last-seen window size.
- The sequence number is equal to the next expected sequence number.
- The acknowledgment number is equal to the last-seen acknowledgment number.
- None of SYN, FIN, or RST are set.

TCP Ambiguous Interpretations

Some captures are quite difficult to analyze automatically, particularly when the time frame may cover both Fast Retransmission and Out-Of-Order packets. A TCP preference allows to switch the precedence of these two interpretations at the protocol level.

TCP Conversation Completeness

TCP conversations are said to be complete when they have both opening and closing handshakes, independently of any data transfer. However, we might be interested in identifying complete conversations with some data sent, and we are using the following bit values to build a filter value on the tcp.completeness field :

- 1 : SYN
- 2 : SYN-ACK
- 4 : ACK
- 8 : DATA
- 16 : FIN
- 32 : RST

For example, a conversation containing only a three-way handshake will be found with the filter 'tcp.completeness==7' (1+2+4) while a complete conversation with data transfer will be found with a longer filter as closing a connection can be associated with FIN or RST packets, or even both : 'tcp.completeness==31 or tcp.completeness==47 or tcp.completeness==63'

14. How to run Nmap scan

Nmap is short for Network Mapper. It is an open-source Linux command-line tool that is used to scan IP addresses and ports in a network and to detect installed applications.
Nmap allows network admins to find which devices are running on their network, discover open ports and services, and detect vulnerabilities.

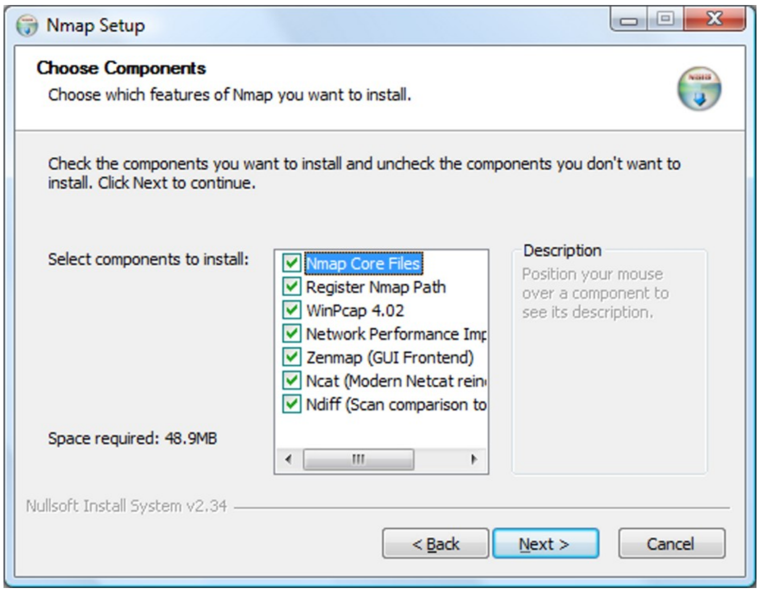
Why use Nmap?
There are a number of reasons why security pros prefer Nmap over other scanning tools.
First, Nmap helps you to quickly map out a network without sophisticated commands or configurations. It also supports simple commands (for example, to check if a host is up) and complex scripting through the Nmap scripting engine.
Other features of Nmap include:

- Ability to quickly recognize all the devices including servers, routers, switches, mobile devices, etc on single or multiple networks.
- Helps identify services running on a system including web servers, DNS servers, and other common applications. Nmap can also detect application versions with reasonable accuracy to help detect existing vulnerabilities.
- Nmap can find information about the operating system running on devices. It can provide detailed information like OS versions, making it easier to plan additional approaches during penetration testing.
- During security auditing and vulnerability scanning, you can use Nmap to attack systems using existing scripts from the Nmap Scripting Engine.
- Nmap has a graphical user interface called Zenmap. It helps you develop visual mappings of a network for better usability and reporting.

Installing Nmap on Windows:

Step 1
Download the Windows version of Nmap from www.nmap.org.
File name is **nmap-7.93-setup.exe**

Step 2
Launch the Nmap setup program. Select the default installation (recommended) which will install the entire Nmap suite of utilities.


Nmap Setup

Choose Components
Choose which features of Nmap you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:
- ✓ Nmap Core Files
- ✓ Register Nmap Path
- ✓ WinPcap 4.02
- ✓ Network Performance Imp
- ✓ Zenmap (GUI Frontend)
- ✓ Ncat (Modern Netcat rein
- ✓ Ndiff (Scan comparison to

Description
Position your mouse over a component to see its description.

Space required: 48.9MB

Nullsoft Install System v2.34

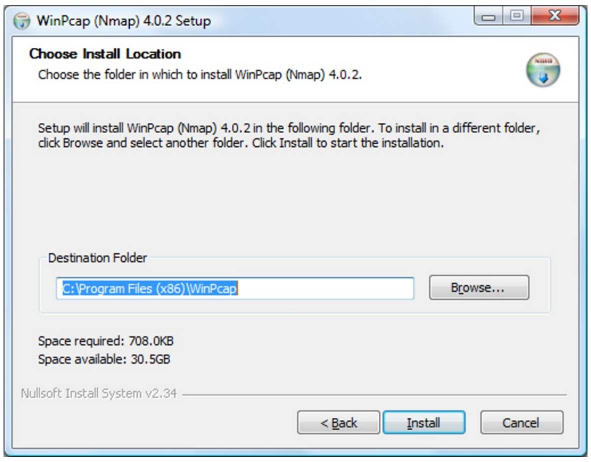
< Back | Next > | Cancel

Nmap for Windows installer

Step 3

During installation, a helper program called WinPcap will also be installed. WinPcap is required for Nmap to function properly on the Windows platform so do not skip this step.

WnPcap for Windows installer


WinPcap (Nmap) 4.0.2 Setup

Choose Install Location

Choose the folder in which to install WinPcap (Nmap) 4.0.2.

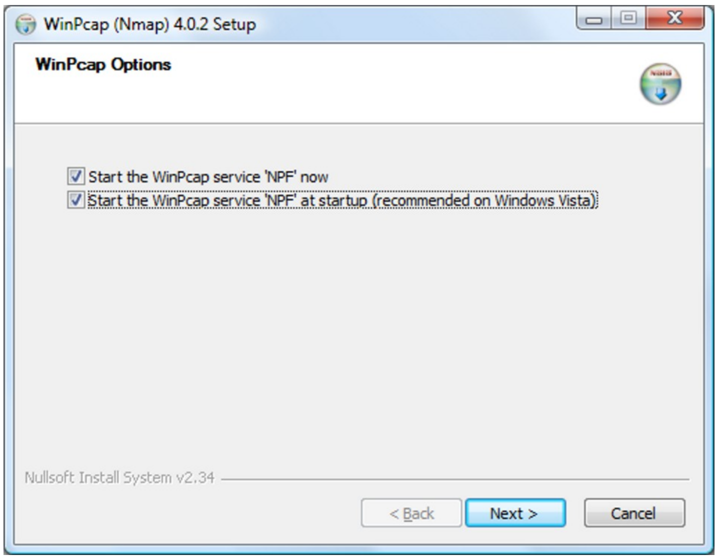
Setup will install WinPcap (Nmap) 4.0.2 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Program Files (x86)\WinPcap

Space required: 708.0KB
Space available: 30.5GB

Nullsoft Install System v2.34

After the WinPcap installation has completed you are given the option to configure its service settings. The default options will enable the WinPcap service to start when Windows boots. This is recommended as Nmap will not function correctly when the WinPcap service is not running


WinPcap (Nmap) 4.0.2 Setup

WinPcap Options

- [✓] Start the WinPcap service 'NPF' now
- [✓] Start the WinPcap service 'NPF' at startup (recommended on Windows Vista)

Nullsoft Install System v2.34

Step 4

Once Nmap has been successfully installed you can verify it is working correctly by executing nmap scanme.insecure.org on the command line (located in Start > Programs > Accessories > Command Prompt).

```
C:\>nmap scanme.insecure.org
Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-07 09:36 Central
Daylight Time
Interesting ports on scanme.nmap.org (64.13.134.52):
Not shown: 994 filtered ports
PORT       STATE  SERVICE
 25/tcp    closed smtp
 70/tcp    closed gopher
 80/tcp    open   http
 110/tcp   closed pop3
 113/tcp   closed auth
 31337/tcp closed Elite
Nmap done: 1 IP address (1 host up) scanned in 9.25 seconds
C:\>
```

Nmap test scan on Microsoft Windows

If the results of your scan are similar to the results above, then you have successfully installed Nmap. If you receive an error, set the path in environment variables and run.

Scan a single target:
Executing Nmap with no command line options will perform a basic scan on the specified target. A target can be specified as an IP address or host name (which Nmap will try to resolve).

Usage syntax: nmap [target]

```
C:\> nmap 192.168.10.1

Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-07 19:38 CDT
Interesting ports on 192.168.10.1:
Not shown: 997 filtered ports
PORT   STATE  SERVICE
20/tcp closed ftp-data
21/tcp closed ftp
80/tcp open   http

Nmap done: 1 IP address (1 host up) scanned in 7.21 seconds
```

Single target scan

The resulting scan shows the status of ports detected on the specified target. The table below describes the output fields displayed by the scan.

PORT	STATE	SERVICE
Port number/protocol	*Status of the port*	*Type of service for the port*

A default Nmap scan will check for the 1000 most commonly used TCP/IP ports. Ports that respond to a probe are classified into one of six port states: open, closed, filtered, unfiltered, open|filtered, closed|filtered. See Appendix B for more information about port states.

Scan multiple targets:

Nmap can be used to scan multiple hosts at the same time. The easiest way to do this is to string together the target IP addresses or host names on the command line (separated by a space).

Usage syntax: nmap [target1 target2 etc]

```
$ nmap 192.168.10.1 192.168.10.100 192.168.10.101

Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-07 20:30 CDT
Interesting ports on 192.168.10.1:
Not shown: 997 filtered ports
PORT   STATE  SERVICE
20/tcp closed ftp-data
21/tcp closed ftp
80/tcp open   http

Interesting ports on 192.168.10.100:
Not shown: 995 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
2049/tcp open  nfs

Nmap done: 3 IP addresses (2 hosts up) scanned in 6.23 seconds
```

Multiple target scan

Scan a range of IP addresses:

A range of IP addresses can be used for target specification as demonstrated in the example below.

Usage syntax: nmap [Range of IP addresses]

$ nmap 192.168.10.1-100

Scan an entire subnet:

Nmap can be used to scan an entire subnet using CIDR (Classless Inter-Domain Routing) notation.

Usage syntax: nmap [Network/CIDR]

$ nmap 192.168.10.1/24

Exclude targets from a scan:

The **--exclude** option is used with Nmap to exclude hosts from a scan.

Usage syntax: nmap [targets] --exclude [target(s)]

$ nmap 192.168.10.0/24 --exclude 192.168.10.100

15. Operating System Detection using Nmap

One of Nmap's most remarkable (and incredibly useful) features is its ability to detect operating systems and services on remote systems. This feature analyzes responses from scanned targets and attempts to identify the host's operating system and installed services.

The process of identifying a target's operating system and software versions is known as TCP/IP fingerprinting. Although it is not an exact science, Nmap developers have taken great care in making TCP/IP fingerprinting an accurate and reliable feature.

Feature	Option
Operating System Detection	**-O**
Attempt to Guess an Unknown OS	**--osscan-guess**
Service Version Detection	**-sV**
Perform a RPC Scan	**--version-trace**
Troubleshooting Version Scans	**-sR**

Operating System Detection:

The -O parameter enables Nmap's operating system detection feature.

Usage syntax: nmap -O [target]

```
# nmap -O 10.10.1.48

Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-11 13:09 Central
Daylight Time
...

MAC Address: 00:0C:29:D5:38:F4 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.9 - 2.6.28
Network Distance: 1 hop

...
```

Output of Nmap's operating system detection feature

As demonstrated above, Nmap is (in most cases) able to identify the operating system on a remote target. Operating system detection is performed by analyzing responses from the target for a set of predictable characteristics which can be used to identify the type of OS on the remote system.

In order for OS detection to work properly there must be at least one open and one closed port on the target system. When scanning multiple targets, the --osscan-limit option can be combined with -O to instruct Nmap not to OS scan hosts that do not meet this criteria.

Attempt to Guess an Unknown Operating System:

If Nmap is unable to accurately identify the OS, you can force it to guess by using the **--osscan-guess** option.

Usage syntax: nmap -O --osscan-guess [target]

```
# nmap -O --osscan-guess 10.10.1.11

Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-17 13:25 CDT
Interesting ports on 10.10.1.11:
Not shown: 999 closed ports
PORT     STATE SERVICE
3001/tcp open  nessus
MAC Address: 00:20:4A:69:FD:94 (Pronet Gmbh)
Aggressive OS guesses: Enerdis Enerium 200 energy monitoring device or
Mitsubishi XD1000 projector (96%), Lantronix UDS200 external serial
device server (96%), Lantronix Xport-03 embedded serial device server
(firmware 6.1.0.3) (95%), Larus 54580 NTP server (95%), Lantronix
Evolution OS (93%), Lantronix UDS1100 external serial device server
(92%), Lantronix XPort embedded Ethernet device server (90%),
Stonewater Control Systems environmental monitoring appliance (88%),
FreeBSD 6.3-PRERELEASE (88%), Crestron MC2E, MP2E, PRO2, or QM-RMC
control and automation system (2-Series) (87%)
...
```

Nmap operating system guess output

The example above displays a list of possible matches for the target's operating system. Each guess is listed with a percentage of confidence Nmap has in the supplied match.

16. Do the following using NS2 Simulator
i. NS2 Simulator-Introduction
ii. Simulate to Find the Number of Packets Dropped by TCP/UDP
iii. Simulate to Find the Number of Packets Dropped due to Congestion

i. NS2 Simulator-Introduction

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2.

BASIC ARCHITECTURE:

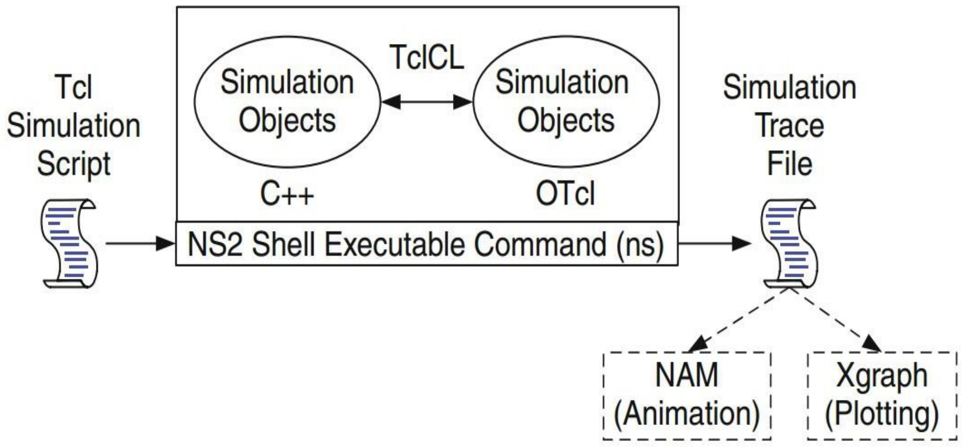


Fig. 2.1. Basic architecture of NS.

Figure 2.1 shows the basic architecture of NS2. NS2 provides users with an executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns.

In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a fronten).

The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g.,_o10) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may defines its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively. Before proceeding further, the readers are encouraged to learn C++ and OTcl languages. We refer the readers to [14] for the detail of C++, while a brief tutorial of Tcl and OTcl tutorial are given in Appendices A.1 and A.2, respectively.

INSTALLATION OF NS2:

1. Install Ubuntu as virtual machine by using Oracle VM Virtual Box Manager.
2. Install NS2 by using the command in terminal.
   sudo apt-get install ns2
3. Nam is also needed to install. Nam (Network Animator) is an animation tool to graphically represent the network and packet traces. Use this command.
   sudo apt-get install nam
4. Install tcl by using the command in terminal
   sudo apt install tcl

CONCEPT OVERVIEW:

Tcl scripting

Tcl is a general purpose scripting language. [Interpreter]

• Tcl runs on most of the platforms such as Unix, Windows, and Mac.
• The strength of Tcl is its simplicity.
• It is not necessary to declare a data type for variable prior to the usage.

Basics of TCL

Syntax: command arg1 arg2 arg3

Hello World!

puts stdout{Hello, World!} Hello, World!

Variables Command
Substitution set a 5 set len
[string length foobar]

set b $a set len [expr [string length foobar] + 9]

Wired TCL Script Components

Create the event scheduler

Open new files & turn on the
tracing Create the nodes

Setup the links

Configure the traffic type (e.g., TCP, UDP,
etc) Set the time of traffic generation (e.g.,
CBR, FTP) Terminate the simulation

NS Simulator Preliminaries.

1. Initialization and termination aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.
5. Tracing and random variables.

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

set ns [new Simulator]

Which is thus the first line in the tcl script. This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

In order to have output files with data on the simulation (trace files) or files used for visualization (nam files), we need to create the files using —open command:

```
#Open the Trace file
          set tracefile1 [open out.tr w]
          $ns trace-all $tracefile1
#Open the NAM trace file
          Set namefile [open out.nam w]
          $ns namtrace-all $namfile
```

Definition of a network of links and nodes

The way to define a node is

set n0 [$ns node]

Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

$ns duplex-link $n0 $n2 10Mb 10ms DropTail

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction.
To define a directional link instead of a bi-directional one, we should replace —duplex-link by —simplex-link.
In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

```
#set Queue Size of link (n0-n2) to 20
$ns queue-limit $n0 $n2 20
```

FTP over TCP

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received.

There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line:

set tcp [new Agent/TCP]

The command **$ns attach-agent $n0 $tcp** defines the source node of the tcp connection.
The command **set sink [new Agent /TCPSink]** Defines the behavior of the destination node of TCP and assigns to it a pointer called sink.

#Setup a UDP

connection set udp

[new Agent/UDP]
$ns attach-agent $n1
$udp set null [new
Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_2

#setup a CBR over UDP connection
The below shows the definition of a CBR application using a UDP agent
The command **$ns attach-agent $n4 $sink** defines the destination node. The command **$ns connect**
$tcp $sink finally makes the TCP connection between the source and destination nodes.

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetsize_ 100
$cbr set rate_ 0.01Mb
$cbr set random_ false

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes.This can be changed to another value, say 552bytes, using the command **$tcp set packetSize_ 552**.
When we have several flows, we may wish to distinguish them so that we can identify them with different colors in the visualization part. This is done by the command **$tcp set fid_ 1** that assigns to the TCP connection a flow identification of ―1.We shall later give the flow identification of ―2‖ to the UDP connection.

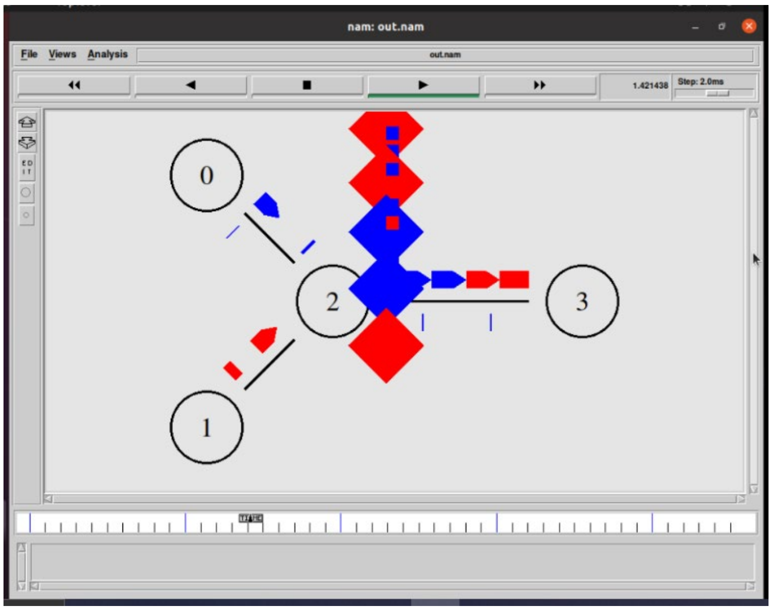
ii. Simulate to Find the Number of Packets Dropped by TCP/UDP

Program:

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}

#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5


#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

```
#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false


#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

#Run the simulation
$ns run
```

Output:



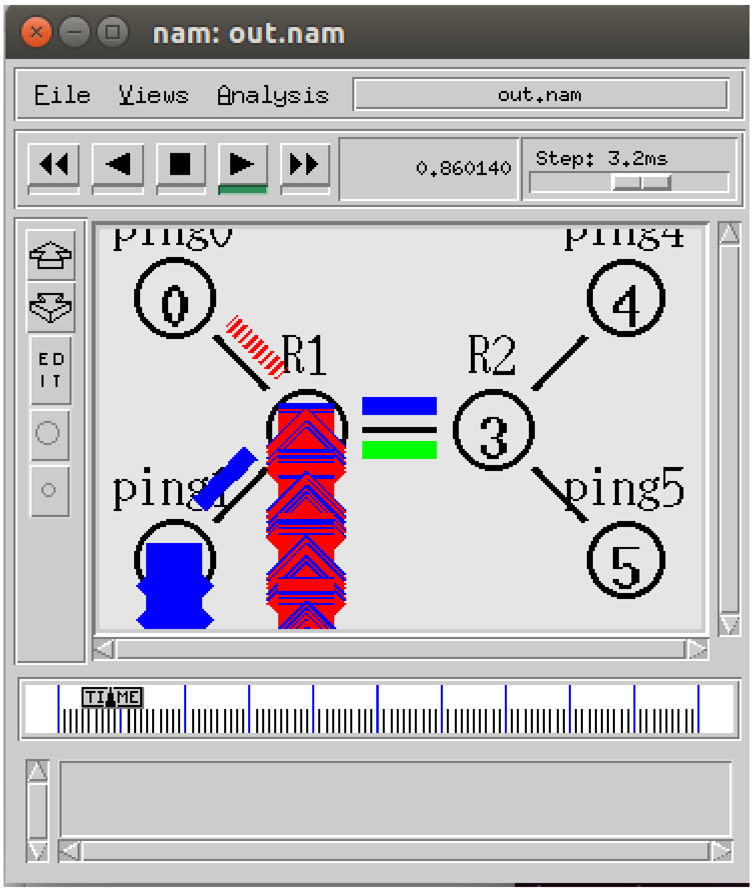
iii. Simulate to Find the Number of Packets Dropped due to Congestion

Program:

```
#===================================
#     Simulation parameters setup
#===================================
set val(stop)   10.0                         ;# time of simulation end

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile

#===================================
#        Nodes Definition
#===================================
#Create 6 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n0 label "ping0"
$n1 label "ping1"
$n2 label "R1"
$n3 label "R2"
$n4 label "ping4"
$n5 label "ping5"
$ns color 1 red
$ns color 2 blue
$ns color 3 green
$ns color 4 orange

#===================================
#        Links Definition
#===================================
#Createlinks between nodes
$ns duplex-link $n0 $n2 1Mb 10ms DropTail

$ns duplex-link $n1 $n2 0.4Mb 10ms DropTail

$ns duplex-link $n2 $n3 4Kb 10ms DropTail

$ns duplex-link $n3 $n4 1Mb 10ms DropTail

$ns duplex-link $n3 $n5 1Mb 10ms DropTail


#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
```

```
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down
#add manually
set ping0 [new Agent/Ping]
$ns attach-agent $n0 $ping0

set ping1 [new Agent/Ping]
$ns attach-agent $n1 $ping1

set ping4 [new Agent/Ping]
$ns attach-agent $n4 $ping4

set ping5 [new Agent/Ping]
$ns attach-agent $n5 $ping5

$ns connect $ping0 $ping4
$ns connect $ping1 $ping5
proc sendPingPacket {} {
        global ns ping0 ping1
        set intervalTime 0.001
        set now [$ns now]
        $ns at [expr $now + $intervalTime] "$ping0 send"
        $ns at [expr $now + $intervalTime] "$ping1 send"
        $ns at [expr $now + $intervalTime] "sendPingPacket"
}
#rtt=round trip time(packet travel from src to dest and back to src)
Agent/Ping instproc recv {from rtt} {
        global seq
        $self instvar node_
        puts "The node [$node_ id] received an ACK from the node $from with RTT
$rtt ms"
}
$ping0 set class_ 1
$ping1 set class_ 2
$ping4 set class_ 3
$ping5 set class_ 4
#end
#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
#add manually
$ns at 0.01 "sendPingPacket"
$ns at 10.0 "finish"
$ns run
#end
```

Output:


nam: out.nam
File Views Analysis
out.nam
0.860140
Step: 3.2ms
EDIT
TIME
ping0
ping4
ping1
ping5
R1
R2
0
1
2
3
4
5