

EXPERIMENT NO - 1

Aim: To study and implement crimping of RJ45 connector using crimping tool.

Theory:

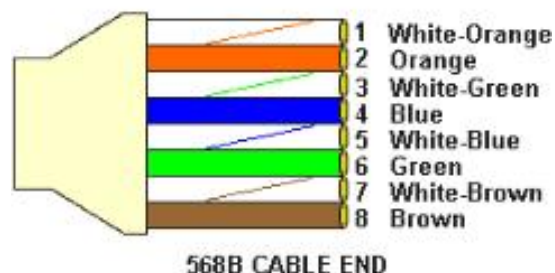
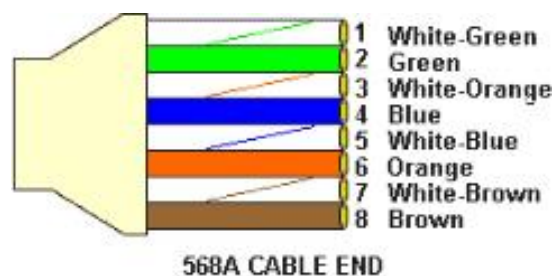
Crimping an RJ45 Connector Correctly Proper Wiring for Ethernet Cat5/Cat5e/Cat 6 Cables. Cables can transmit information along their length. To actually get that information where it needs to go, you need to make the right connections to an RJ45 connector.



Your cable run needs to terminate into a connector, and that connector needs a jack to plug into. Registered Jack 45 (RJ45) is a standard type of physical connector for network cables. RJ45 connectors are commonly seen with Ethernet cables and networks.

Modern Ethernet cables feature a small plastic plug on each end of the cable. That plug is inserted into RJ45 jacks of Ethernet devices. The term “plug” refers to the cable or “male” end of the connection while the term “jack” refers to the port or “female” end.

T568A or T568B Wiring Standard:



T568A and T568B are the two colour codes used for wiring eight-position modular plugs. Both are allowed under the ANSI/TIA/EIA wiring standards. The only difference between the two color codes is that the orange and green pairs are interchanged.

There is no transmission differences between T568A and T568B cabling schemes. North America's preference is for T568B. Both ends must use the same standard. It makes no difference to the transmission characteristics of data.

T568B wiring pattern is recognized as the preferred wiring pattern.

STEP 1:

Using a Crimping Tool, trim the end of the cable you're terminating, to ensure that the ends of the conducting wires are even



STEP 2:

Being careful not to damage the inner conducting wires, strip off approximately 1 inch of the cable jacket, using a modular crimping tool or a UTP cable stripper.



STEP 3:

Separate the 4 twisted wire pairs from each other, and then unwind each pair, so that you end up with 8 individual wires. Flatten the wires out as much as possible, since they'll need to be very straight for proper insertion into the connector.



STEP 4:

Holding the cable with the wire ends facing away from you. Moving from left to right, arrange the wires in a flat, side-by-side ribbon formation, placing them in the following order: white/orange, solid orange, white/green, solid blue, white/blue, solid green, white/brown, solid brown.



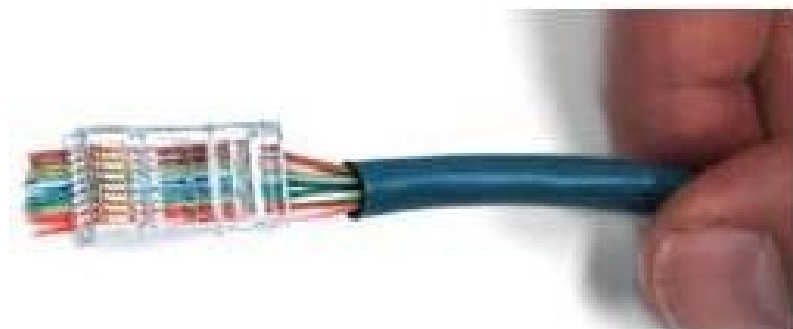
STEP 5:

Holding the RJ45 connector so that its pins are facing away from you and the plug-clip side is facing down, carefully insert the flattened, arranged wires into the connector, pushing through until the wire ends emerge from the pins. For strength of connection, also push as much of the cable jacket as possible into the connector.



STEP 6:

Check to make sure that the wire ends coming out of the connector's pin side are in the correct order; if not, remove them from the connector, rearrange into proper formation, and re-insert. Remember, once the connector is crimped onto the cable, it's permanent. If you realize that a mistake has been made in wire order after termination, you'll have to cut the connector off and start all over again!



STEP 7:

Insert the prepared connector/cable assembly into the RJ45 slot in your crimping tool. Firmly squeeze the crimper's handles together until you can't go any further. Release the handles and repeat this step to ensure a proper crimp.



STEP 8:

If your crimper doesn't automatically trim the wire ends upon termination, carefully cut wire ends to make them as flush with the connector's surface as possible. The closer the wire ends are trimmed, the better your final plug-in connection will be.



STEP 9:

After the first termination is complete, repeat process on the opposite end of your cable.



CONCLUSION : Thus, we have studied the use of crimping tool for RJ-45.

EXPERIMENT NO - 2

AIM: To study and run basic networking commands in Linux.

THEORY:

1. ifconfig

ifconfig(interface configuration) command is used to configure the kernel-resident network interfaces. It is used at the boot time to set up the interfaces as necessary. After that, it is usually used when needed during debugging or when you need system tuning. Also, this command is used to assign the IP address and netmask to an interface or to enable or disable a given interface.

```
student@lenovo804-ThinkCentre-M70e: ~  
student@lenovo804-ThinkCentre-M70e:~$ ifconfig  
docker0    Link encap:Ethernet  HWaddr 02:42:cf:c7:15:71  
            inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0  
            UP BROADCAST MULTICAST  MTU:1500  Metric:1  
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:0  
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)  
  
eth0       Link encap:Ethernet  HWaddr 44:37:e6:4d:df:1b  
            inet addr:10.1.8.4  Bcast:10.255.255.255  Mask:255.0.0.0  
            inet6 addr: fe80::4637:e6ff:fe4d:df1b/64 Scope:Link  
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
            RX packets:51944 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:18626 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1000  
            RX bytes:27621649 (27.6 MB)  TX bytes:2682227 (2.6 MB)  
            Interrupt:17  
  
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:2173 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:2173 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:0  
            RX bytes:193433 (193.4 KB)  TX bytes:193433 (193.4 KB)  
  
student@lenovo804-ThinkCentre-M70e:~$
```

2. NSLOOKUP

Nslookup (stands for “Name Server Lookup”) is a useful command for getting information from DNS server. It is a network administration tool for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or any other specific DNS record. It is also used to troubleshoot DNS related problems.


```
student@lenovo804-ThinkCentre-M70e: ~  
student@lenovo804-ThinkCentre-M70e:~$ nslookup www.atharvacoe.ac.in  
Server:          127.0.1.1  
Address:         127.0.1.1#53  
  
Non-authoritative answer:  
www.atharvacoe.ac.in    canonical name = atharvacoe.ac.in.  
Name:   atharvacoe.ac.in  
Address: 192.185.180.65  
  
student@lenovo804-ThinkCentre-M70e:~$
```

3. Ping

PING (Packet Internet Groper) command is used to check the network connectivity between host and server/host. This command takes as input the IP address or the URL and sends a data packet to the specified address with the message “PING” and get a response from the server/host this time is recorded which is called latency. Fast ping low latency means faster connection. Ping uses ICMP(Internet Control Message Protocol) to send an ICMP echo message to the specified host if that host is available then it sends ICMP reply message. Ping is generally measured in millisecond every modern operating system has this ping preinstalled.

```
student@lenovo804-ThinkCentre-M70e: ~  
student@lenovo804-ThinkCentre-M70e:~$ ping -c 4 10.1.8.3  
PING 10.1.8.3 (10.1.8.3) 56(84) bytes of data.  
64 bytes from 10.1.8.3: icmp_seq=1 ttl=64 time=0.324 ms  
64 bytes from 10.1.8.3: icmp_seq=2 ttl=64 time=0.333 ms  
64 bytes from 10.1.8.3: icmp_seq=3 ttl=64 time=0.316 ms  
64 bytes from 10.1.8.3: icmp_seq=4 ttl=64 time=0.302 ms  
  
--- 10.1.8.3 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3000ms  
rtt min/avg/max/mdev = 0.302/0.318/0.333/0.024 ms  
student@lenovo804-ThinkCentre-M70e:~$
```

4. TRACEROUTE

tracert command in Linux prints the route that a packet takes to reach the host. This command is useful when you want to know about the route and about all the hops that a packet takes. Below image depicts how tracert command is used to reach the Google(172.217.26.206) host from the local machine and it also prints detail about all the hops that it visits in between.

```

student@lenovo804-ThinkCentre-M70e:~$ traceroute
Usage:
  traceroute [ -46dFf first_ttl ] [ -g gate,... ] [ -i device ] [ -m max_ttl ] [ -N queries ] [ -p port ] [ -t tos ] [ -l flow_label ] [ -w waittime ] [ -q queries ] [ -s src_addr ] [ -z sendwait ] [
  --mark-num ] host [ packetlen ]
Options:
  -4                Use IPv4
  -6                Use IPv6
  -d --debug        Enable socket level debugging
  -F --dont-fragment Do not fragment packets
  -f first_ttl      --first=first_ttl      Start from the first_ttl hop (instead from 1)
  -g gate,...       --gateway=gate,...     Route packets through the specified gateway
                                          (maximum 8 for IPv4 and 127 for IPv6)
  -I --icmp         Use ICMP ECHO for tracerouting
  -T --tcp          Use TCP SYN for tracerouting (default port is 80)
  -i device         --interface=device     Specify a network interface to operate with
  -m max_ttl        --max-hops=max_ttl     Set the max number of hops (max TTL to be
                                          reached). Default is 30
  -N queries         --sin-queries=queries Set the number of probes to be tried
                                          simultaneously (default is 16)
  -n                Do not resolve IP addresses to their domain names
  -p port           --port=port           Set the destination port to use. It is either
                                          initial udp port value for "default" method
                                          (incremented by each probe, default is 33434), or
                                          initial seq for "icmp" (incremented as well,
                                          default from 1), or some constant destination
                                          port for other methods (with default of 80 for
                                          "tcp", 53 for "udp", etc.)
  -t tos            --tos=tos             Set the TOS (IPv4 type of service) or TC (IPv6
                                          traffic class) value for outgoing packets
  -l flow_label      --flowlabel=flow_label Use specified flow_label for IPv6 packets
  -w waittime        --wait=waittime      Set the number of seconds to wait for response to
                                          a probe (default is 5.0). Non-integer (float
                                          point) values allowed too
  -q queries         --queries=queries     Set the number of probes per each hop. Default is
                                          3
  -r                Bypass the normal routing and send directly to a
                                          host on an attached network
  -s src_addr        --source=src_addr     Use source src_addr for outgoing packets
  -z sendwait        --sendwait=sendwait   Minimal time interval between probes (default 0).
                                          If the value is more than 10, then it specifies a
                                          number in milliseconds, else it is a number of
                                          seconds (float point values allowed too)
  -e --extensions   Show ICMP extensions (if present), including MPLS
  -A --as-path-lookups Perform AS path lookups in routing registries and
                                          print results directly after the corresponding
                                          addresses
  -M name           --module=name         Use specified module (either builtin or external)

```

5. Netstat

Netstat command displays various network related information such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships etc.,

```

student@lenovo804-ThinkCentre-M70e:~$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:*                     *:*                     LISTEN
tcp        0      0 localhost:ipp            *:*                     LISTEN
tcp        0      0 10.1.8.4:40190           bom05s11-in-f2.1e:https TIME_WAIT
tcp        0      0 10.1.8.4:52797           151.101.2.114:https    TIME_WAIT
tcp        0      0 10.1.8.4:38575           bom05s15-in-f14.1:https ESTABLISHED
tcp        0      0 10.1.8.4:38576           bom05s15-in-f14.1:https ESTABLISHED
tcp        0      0 10.1.8.4:52065           bom05s15-in-f4.1e:https TIME_WAIT
tcp        0      0 10.1.8.4:52796           151.101.2.114:https    TIME_WAIT
tcp        0      0 10.1.8.4:40191           bom05s11-in-f2.1e:https TIME_WAIT
tcp        0      0 10.1.8.4:38634           bom05s15-in-f14.1:https ESTABLISHED
tcp        0      0 10.1.8.4:38637           bom05s15-in-f14.1:https TIME_WAIT
tcp        0      0 10.1.8.4:38573           bom05s15-in-f14.1:https ESTABLISHED
tcp        0      0 10.1.8.4:37409           server-52-222-135:https TIME_WAIT
tcp        0      0 10.1.8.4:41299           a184-30-54-102.de:https TIME_WAIT

```

6. ARP

arp command manipulates the System's ARP cache. It also allows a complete dump of the ARP cache. ARP stands for Address Resolution Protocol. The primary function of this protocol is to resolve the IP address of a system to its mac address, and hence it works between level 2(Data link layer) and level 3(Network layer).

```

student@lenovo804-ThinkCentre-M70e: ~
student@lenovo804-ThinkCentre-M70e:~$ arp -v
Address            HWtype  HWaddress           Flags Mask            Iface
10.8.1.3           (incomplete)
10.0.0.3           ether   08:35:71:f0:35:c0   C                    eth0
10.1.8.3           ether   44:37:e6:4d:e0:f7   C                    eth0
Entries: 3         Skipped: 0          Found: 3
student@lenovo804-ThinkCentre-M70e:~$

```

7. IP

ip command in Linux is present in the net-tools which is used for performing several network administration tasks. IP stands for Internet Protocol. This command is used to show or manipulate routing, devices, and tunnels. It is similar to *ifconfig* command but it is much more powerful with more functions and facilities attached to it. *ifconfig* is one of the deprecated commands in the net-tools of Linux that has not been maintained for many years. **ip** command is used to perform several tasks like assigning an address to a network interface or configuring network interface parameters.

It can perform several other tasks like configuring and modifying the default and static routing, setting up tunnel over IP, listing IP addresses and property information, modifying the status of the interface, assigning, deleting and setting up IP addresses and routes.

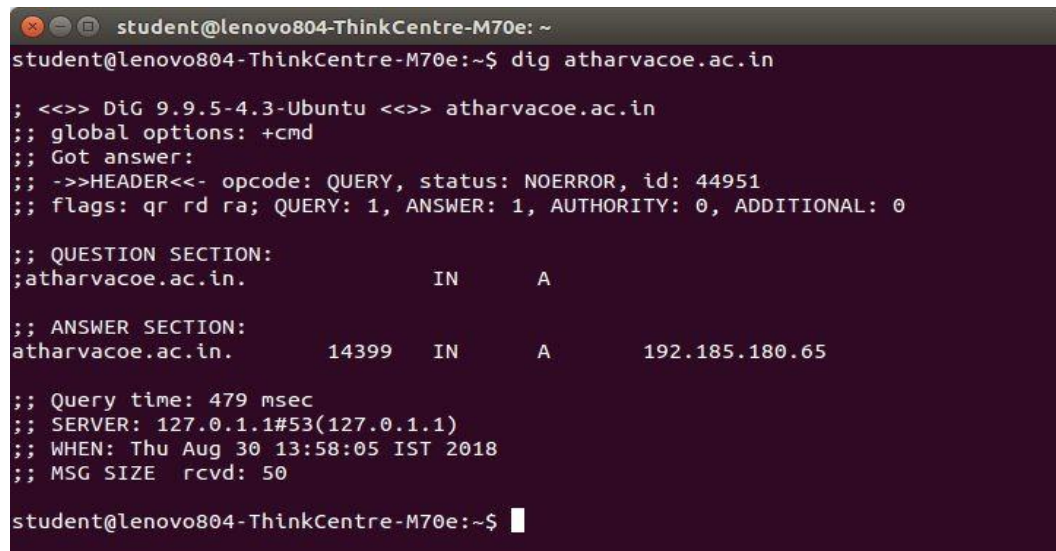
```

student@lenovo804-ThinkCentre-M70e: ~
student@lenovo804-ThinkCentre-M70e:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 44:37:e6:4d:df:1b brd ff:ff:ff:ff:ff:ff
    inet 10.1.8.4/8 brd 10.255.255.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::4637:e6ff:fe4d:df1b/64 scope link
        valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:cf:c7:15:71 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 scope global docker0
        valid_lft forever preferred_lft forever
student@lenovo804-ThinkCentre-M70e:~$

```


8. Dig

dig command stands for *Domain Information Groper*. It is used for retrieving information about DNS name servers. It is basically used by network administrators. It is used for verifying and troubleshooting DNS problems and to perform DNS lookups. Dig command replaces older tools such as nslookup and the host.

A terminal window with a dark purple background and white text. The window title is 'student@lenovo804-ThinkCentre-M70e: ~'. The command 'dig atharvacoe.ac.in' has been executed. The output shows DNS query details: DiG 9.9.5-4.3-Ubuntu, global options: +cmd, Got answer, opcode: QUERY, status: NOERROR, id: 44951, flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0. The QUESTION SECTION shows 'atharvacoe.ac.in. IN A'. The ANSWER SECTION shows 'atharvacoe.ac.in. 14399 IN A 192.185.180.65'. Additional info includes Query time: 479 msec, SERVER: 127.0.1.1#53(127.0.1.1), WHEN: Thu Aug 30 13:58:05 IST 2018, MSG SIZE rcvd: 50. The prompt 'student@lenovo804-ThinkCentre-M70e:~\$' is visible at the bottom.

```
student@lenovo804-ThinkCentre-M70e: ~
student@lenovo804-ThinkCentre-M70e:~$ dig atharvacoe.ac.in

; <<>> DiG 9.9.5-4.3-Ubuntu <<>> atharvacoe.ac.in
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44951
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;atharvacoe.ac.in.                IN      A

;; ANSWER SECTION:
atharvacoe.ac.in.                14399   IN      A      192.185.180.65

;; Query time: 479 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Thu Aug 30 13:58:05 IST 2018
;; MSG SIZE  rcvd: 50

student@lenovo804-ThinkCentre-M70e:~$
```

CONCLUSION: Hence, in this experiment, we have successfully studied some basic networking commands and also implemented them in Linux.



EXPERIMENT NO - 3

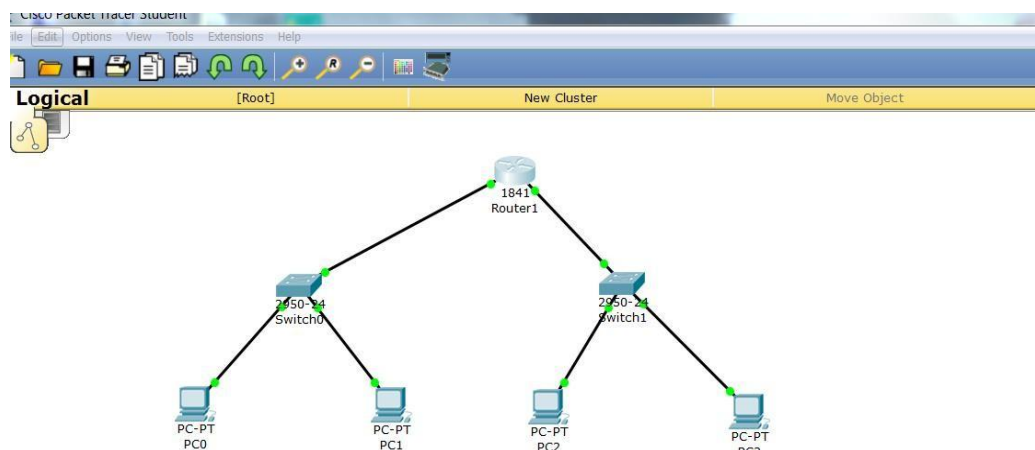
AIM: Build a simple network topology and configure it for static routing protocol using packet tracer. Setup a network and configure IP addressing, subnetting, masking.

THEORY: Cisco Packet Tracer is a cross-platform visual simulation tool designed by Cisco Systems that allows users to create network topologies and imitate modern computer networks. The software allows users to simulate the configuration of Cisco routers and switches using a simulated command line interface. Packet Tracer makes use of a drag and drop user interface, allowing users to add and remove simulated network devices as they see fit. The software is mainly focused towards Certified Cisco Network Associate Academy students as an educational tool for helping them learn fundamental CCNA concepts.

Example Steps: Note: Here you have to perform the practical and write it stepwise along with screenshots.

1. Pick a total of 4 pcs in the packet tracer application.
2. We need 2 routers.
3. We need a single router.

Connect the devices as shown below:



3. Give the appropriate IP addresses to the pcs accordingly. 5. Test the network with the help of packets.

CONCLUSION: Hence we have successfully created simple network using CISCO PACKET TRACER.



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering

Experiment No - 04

Aim:

To study and implement CRC.

Theory:

Cyclic Redundancy Check

- This is a type of polynomial code in which a bit string is represented in the form of polynomials with coefficients 0 and 1 only.
- Polynomial arithmetic uses a modulo-2 arithmetic i.e. addition and subtraction are identical to EX-OR .
- For CRC code, sender and receiver should agree upon an operator polynomial $Q(x)$. Both high and low order bits of generator must be 1.
- To compute CRC for some frame with m -bits corresponding to the polynomial $M(x)$, the frame must be longer than generator.
- The idea is to append a CRC to the end of frame in such way that the frame is divisible by $Q(x)$.
- If there is remainder, there has been a transmission error.
- A CRC will be valid if and only if,
- It should have only one bit less than divisor.
- Appending the CRC to end of data unit should result in bit sequence which is exactly divisible by divisor CRC generation..

STEP 01: Append string of n 0's to the data unit where n is 1 less than the number of bits in the predefined divisor.

STEP 02: Divide the newly generated data unit in step 01 by divisor. This is a binary division.

STEP 03: The remainder obtained after division in step 02 is n -bit CRC.

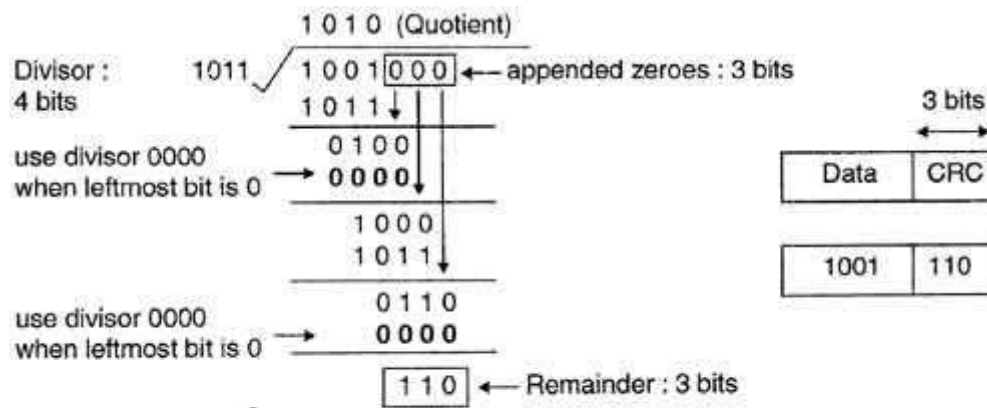
STEP 04: This CRC will replace the n 0's appended to data unit in step 01, to get the code word to be transmitted.



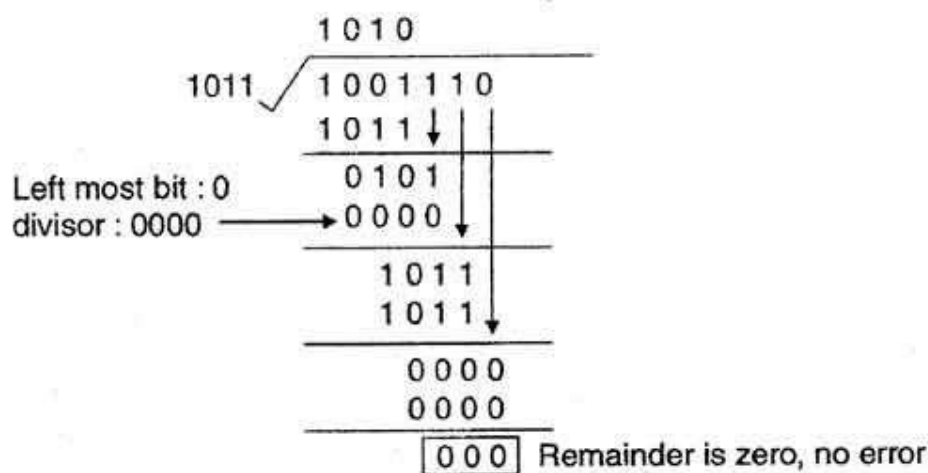
CRC Checker:

- Code word received at receiver consists of data and CRC.
- The receiver treat it as one unit and divide it by the same(n+1) bit divisor which was used at transmitter.
- The remainder of the division is checked.
- If the remainder is zero then the received code word is error free and hence should be accepted.
- But a non-zero remainder indicates presence of errors; hence corresponding code word should be rejected.

For Example:



CRC generated (Binary division)



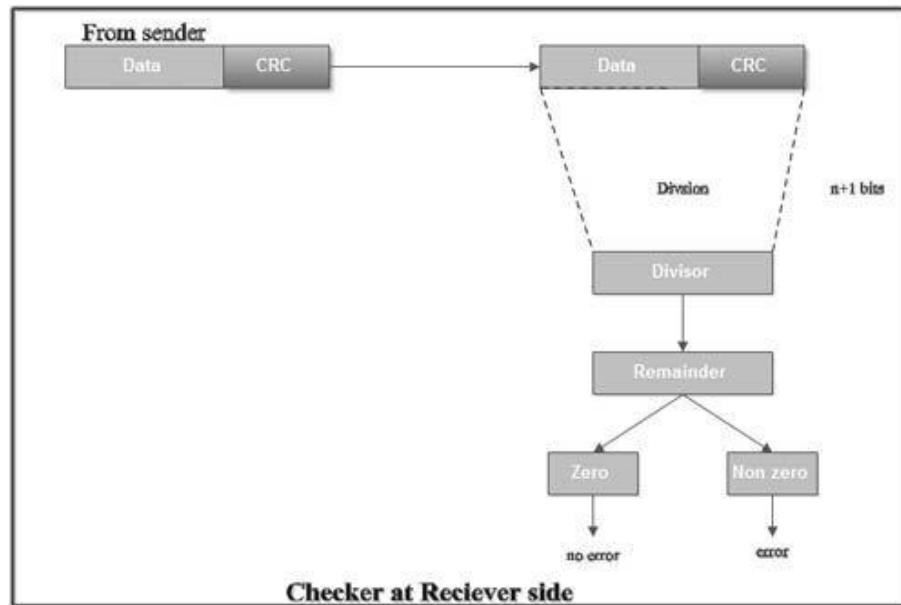
CRC decoded (binary division)



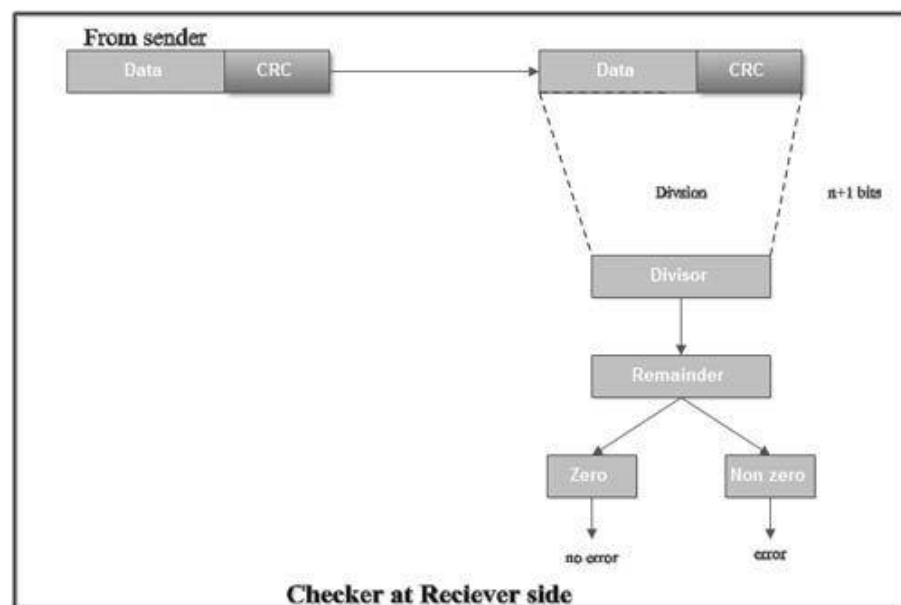
Diagram:

Cyclic Redundancy Check

- Sender's side



- Receiver's Side



Conclusion: In this way we have successfully implemented CRC.



EXPERIMENT NO - 05

AIM:

Java program for Socket Programming

THEORY:

Java Socket Programming

- Java Socket programming is used for communication between the applications running on different JRE.
- Java Socket programming can be connection-oriented or connection-less.
- Socket and Server Socket classes are used for connection-oriented socket programming and Datagram Socket and Datagram Packet classes are used for connection-less socket programming.

The client in socket programming must know two information:

- a. IP Address of Server, and
- b. Port number.

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and Server Socket.

The Socket class is used to communicate client and server. Through this class, we can read and write message. The Server Socket class is used at server-side. The accept() method of Server Socket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

#Socket class

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

#ServerSocket class

The Server Socket class can be used to create a server socket. This object is used to establish communication with the clients.

Creating Server:

To create the server application, we need to create the instance of Server Socket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering

number, it returns an instance of Socket.

```
ServerSocket ss=new ServerSocket(6666);
```

```
Socket s=ss.accept();//establishes connection and waits for the client
```

Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

```
Socket s=new Socket("localhost",6666);
```

Code:

MyServer.java file

```
import java.io.*; import java.net.*; public class MyServer
public static void main(String[] args){ try
{
ServerSocket ss=new ServerSocket(6666); Socket s=ss.accept();//establishes
connection
DataInputStream      dis=new      DataInputStream(s.getInputStream());
Stringstr=(String)dis.readUTF();
System.out.println("message= "+str); ss.close();
}
catch(Exception e){System.out.println(e);}
}
}
```

MyClient.java file

```
import java.io.*; import java.net.*; public class MyClient
{
public static void main(String[] args)
{ try
{
Socket s=new Socket("localhost",6666);
```



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering

```
DataOutputStream dout=new DataOutputStream(s.getOutputStream());  
dout.writeUTF("Hello Server");  
dout.flush();  
dout.close();  
s.close();  
} catch (Exception e) {System.out.println(e);}  
}  
}
```

Output:

To execute this program open two command prompts and execute each program at each command prompt as displayed in the below figures.

First run Myserver.java file in terminal/cmd,

A terminal window titled 'harsh@harsh-Inspiron-3542: ~/Desktop' showing the following commands and output:
harsh@harsh-Inspiron-3542:~/Desktop\$ javac MyServer.java
harsh@harsh-Inspiron-3542:~/Desktop\$ java MyServer

Running MyServer.java

Then in new terminal/cmd run MyClient.java file,

A terminal window titled 'harsh@harsh-Inspiron-3542: ~/Desktop' showing the following commands and output:
harsh@harsh-Inspiron-3542:~/Desktop\$ javac MyClient.java
harsh@harsh-Inspiron-3542:~/Desktop\$ java MyClient
harsh@harsh-Inspiron-3542:~/Desktop\$

Running MyClient.java

As soon as you run MyClient program a message is sent to server and displayed in MyServer Terminal/CMD as shown below,

A terminal window titled 'harsh@harsh-Inspiron-3542: ~/Desktop' showing the following commands and output:
harsh@harsh-Inspiron-3542:~/Desktop\$ javac MyServer.java
harsh@harsh-Inspiron-3542:~/Desktop\$ java MyServer
message= Hello Server
harsh@harsh-Inspiron-3542:~/Desktop\$

Message displayed in MyServer after running MyClient

CONCLUSION: So, in this experiment we have successfully understood the concept of Socket Programming and implemented it using Java Programming.



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering
EXPERIMENT NO - 06

Aim:

Study and implement Leaky bucket algorithm.

Theory:

- Congestion is a situation in Communication Networks in which too many packets are present in a part of the subnet, performance degrades.
- Congestion in a network may occur when the load on the network (i.e. the number of packets sent to the network) is greater than the capacity of the network (i.e. the number of packets a network can handle.).
- Network congestion occurs in case of traffic overloading.
- Congestion Control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened.

The congesting control algorithms are basically divided into two groups:

1. Open Loop Congestion Control

- Policies are used to prevent the congestion before it happens.
- Congestion control is handled either by the source or by the destination.

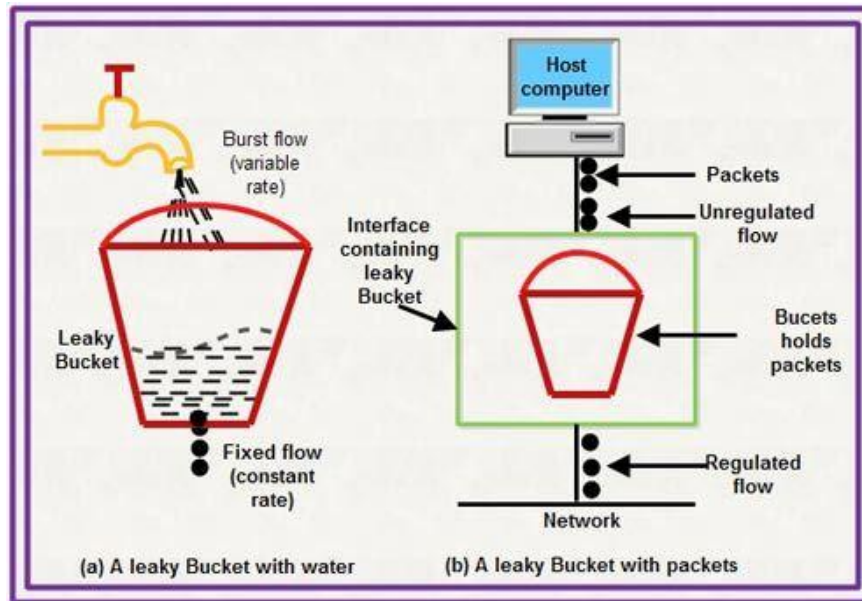
2. Closed Loop Congestion Control

- Closed loop congestion control mechanisms try to remove the congestion after it happens.

Congestion control algorithm

Leaky Bucket Algorithm

- It is a traffic shaping mechanism that controls the amount and the rate of the traffic sent to the network.
- A leaky bucket algorithm shapes bursty traffic into fixed rate traffic by averaging the data rate.
- Imagine a bucket with a small hole at the bottom.
- The rate at which the water is poured into the bucket is not fixed and can vary but it leaks from the bucket at a constant rate. Thus (as long as water is present in bucket), the rate at which the water leaks does not depend on the rate at which the water is input to the bucket.
- Also, when the bucket is full, any additional water that enters into the bucket spills over the sides and is lost.



- Imagine a bucket with a small hole in the bottom. No matter at what rate water enters the bucket, the outflow is at constant rate. When the bucket is full with water additional water entering spills over the sides and is lost.
- Similarly, each network interface contains a leaky bucket and the following steps are involved in leaky bucket algorithm:
 - When host wants to send packet, packet is thrown into the bucket.
 - The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
 - Bursty traffic is converted to a uniform traffic by the leaky bucket.
 - In practice the bucket is a finite queue that outputs at a finite rate.

Drawbacks of Leaky Bucket Algorithm

- The algorithm allows only an average (constant) rate of data flow.
- Its major problem is that it cannot deal with bursty data.
- It does not consider the idle time of the host. For example, if the host was idle for 10 seconds and now it is willing to send data at a very high speed for another 10 seconds, the total data transmission will be divided into 20 seconds and average data rate will be maintained.
- The host is having no advantage of sitting idle for 10 seconds.
- To overcome this problem, a token bucket algorithm is used.
- A token bucket algorithm allows bursty data transfers.

Conclusion: In this way we have implemented Leaky bucket algorithm successfully.



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering
EXPERIMENT NO - 07

AIM:

Use Wireshark to understand the operation of TCP/IP layers:

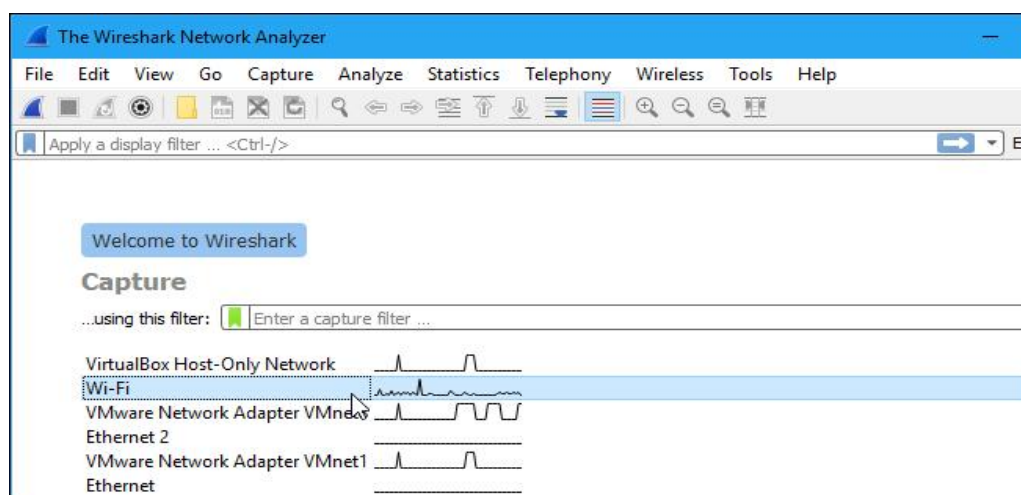
- Ethernet Layer: Frame header, Frame size etc.
- Data Link Layer: MAC address, ARP (IP and MAC address binding)
- Network Layer: IP Packet (header, fragmentation), ICMP (Query and Echo)
- Transport Layer: TCP Ports, TCP handshake segment etc.
- Application Layer: DHCP, FTP, HTTP header formats

THEORY:

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and displays them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets.

Capturing Packets

After downloading and installing Wireshark, you can launch it and double-click the name of a network interface under Capture to start capturing packets on that interface. For example, if you want to capture traffic on your wireless network, click your wireless interface. You can configure advanced features by clicking Capture > Options, but this isn't necessary for now.



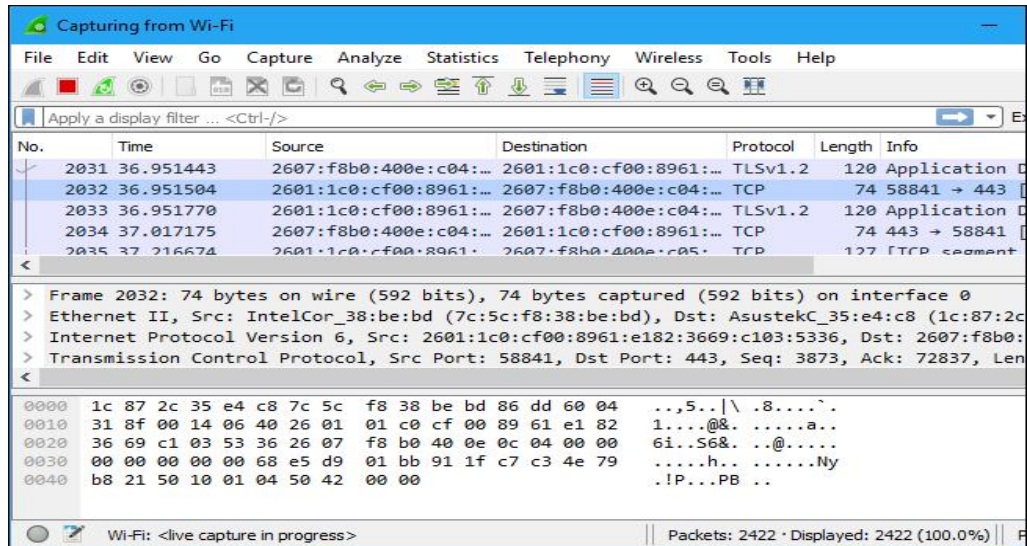
As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.



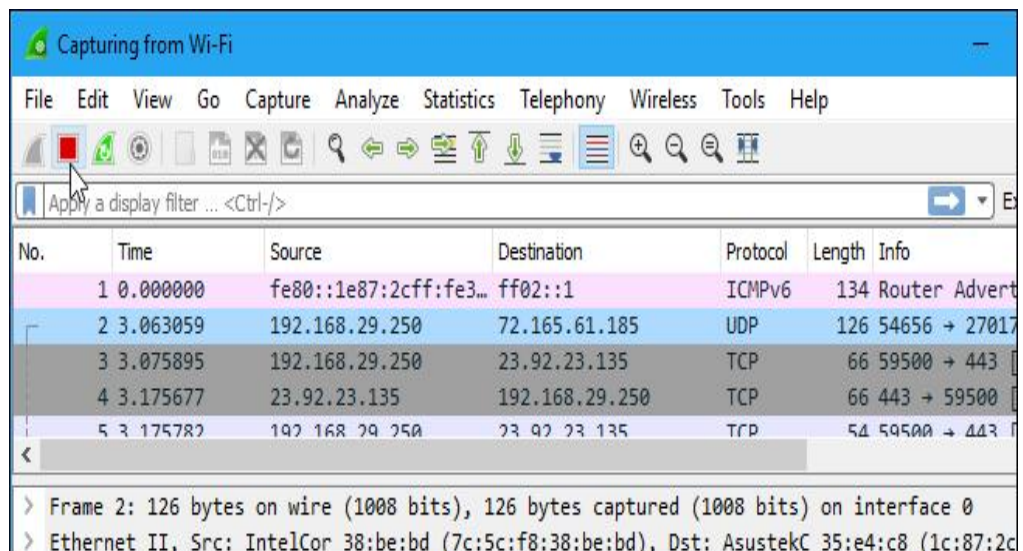
Shivajirao S Jondhale College of Engineering, Dombivli (E)

Department of Computer Engineering

If you have promiscuous mode enabled—it's enabled by default—you'll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click Capture > Options and verify the “Enable promiscuous mode on all interfaces” checkbox is activated at the bottom of this window.



Click the red “Stop” button near the top left corner of the window when you want to stop capturing traffic.



Color Coding

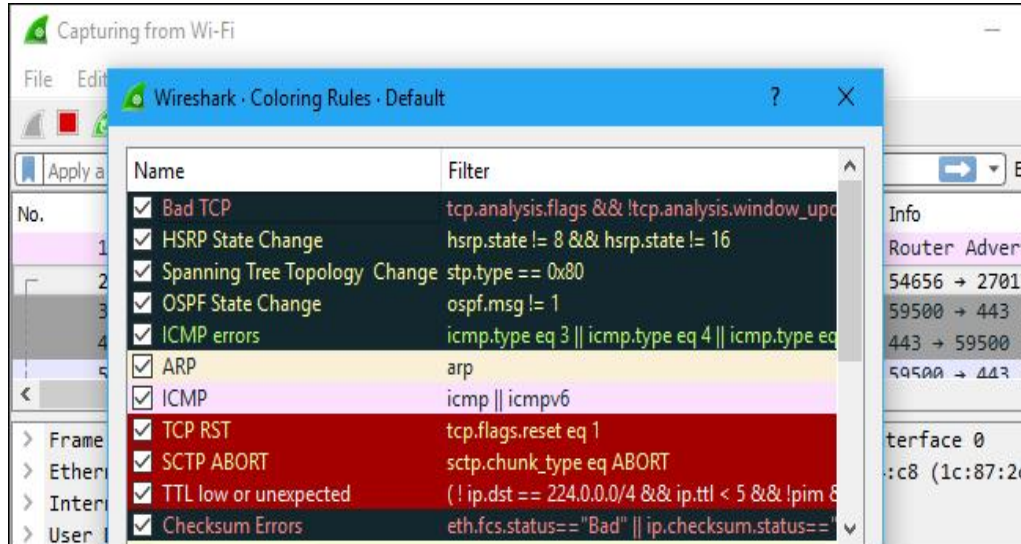
You'll probably see packets highlighted in a variety of different colors. Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black



Shivajirao S Jondhale College of Engineering, Dombivli (E) Department of Computer Engineering

identifies packets with errors—for example, they could have been delivered out of order.

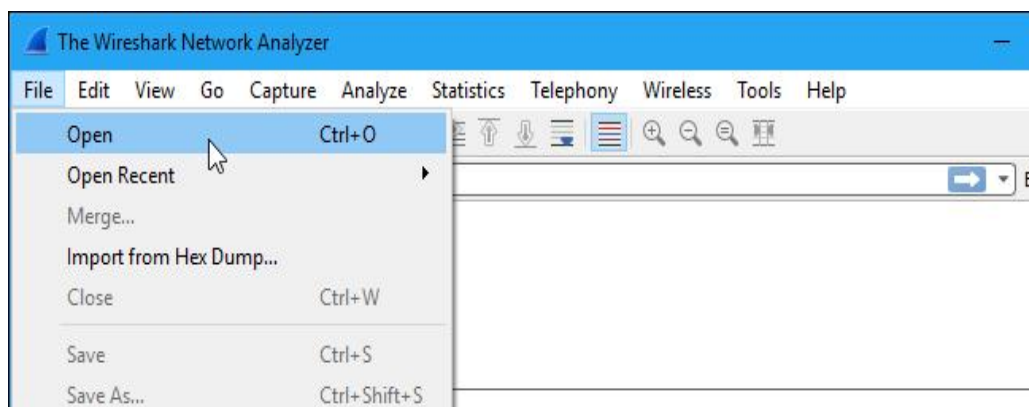
To view exactly what the color codes mean, click View > Coloring Rules. You can also customize and modify the coloring rules from here, if you like.



Sample Captures

If there's nothing interesting on your own network to inspect, Wireshark's wiki has you covered. The wiki contains a page of sample capture files that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one.

You can also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.



Filtering Packets

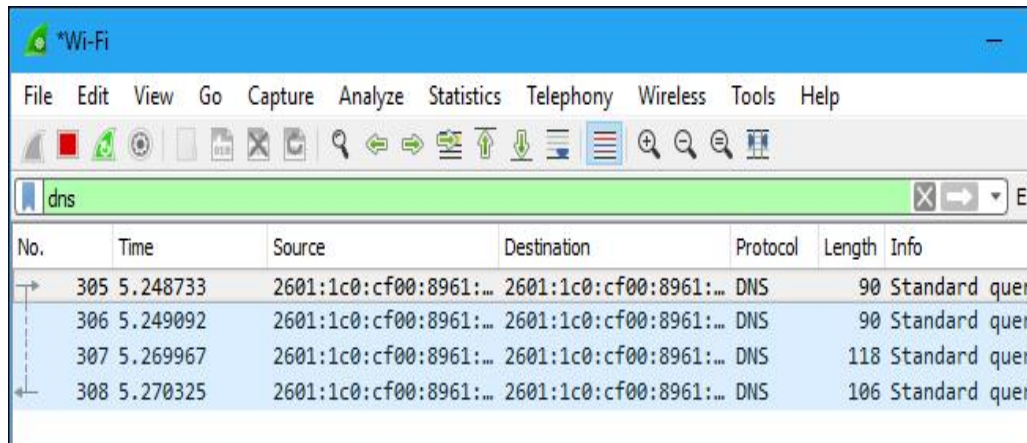
If you're trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using



Shivajirao S Jondhale College of Engineering, Dombivli (E) Department of Computer Engineering

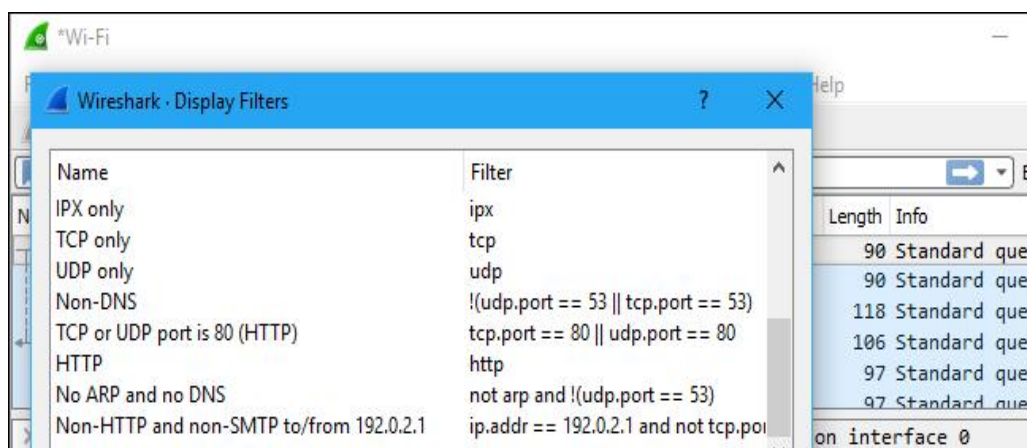
the network so you can narrow down the traffic. Still, you'll likely have a large amount of packets to sift through. That's where Wireshark's filters come in.

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type "dns" and you'll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.



You can also click Analyze > Display Filters to choose a filter from among the default filters included in Wireshark. From here, you can add your own custom filters and save them to easily access them in the future.

For more information on Wireshark's display filtering language, read the [Building display filter expressions](#) page in the official Wireshark documentation.

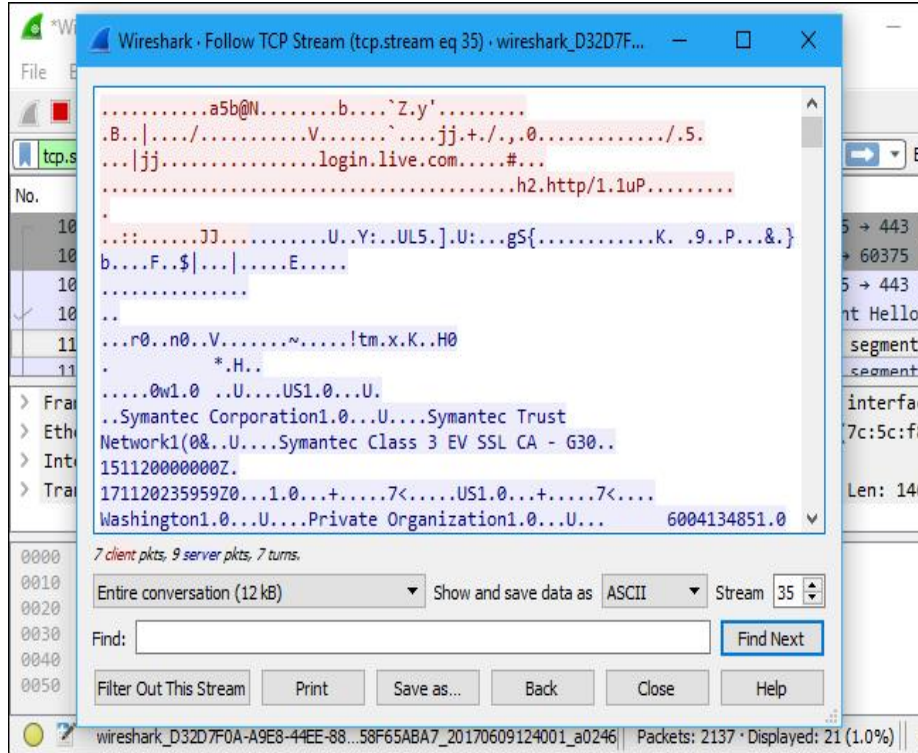


Another interesting thing you can do is right-click a packet and select Follow > TCP Stream.

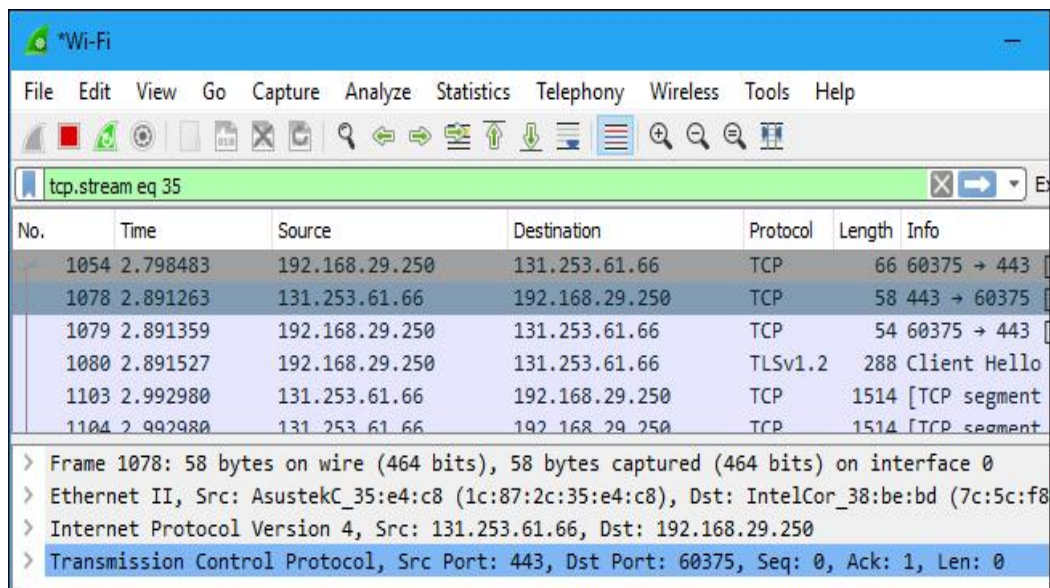


Shivajirao S Jondhale College of Engineering, Dombivli (E) Department of Computer Engineering

You'll see the full TCP conversation between the client and the server. You can also click other protocols in the Follow menu to see the full conversations for other protocols, if applicable.



Close the window and you'll find a filter has been applied automatically. Wireshark is showing you the packets that make up the conversation.



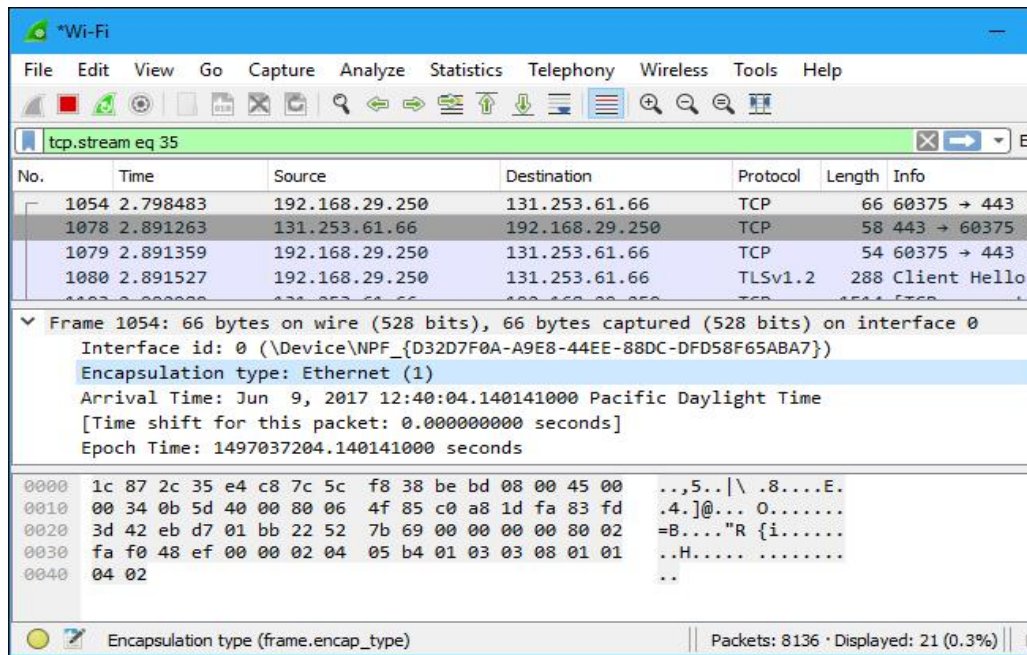
Inspecting Packets

Click a packet to select it and you can dig down to view its details.

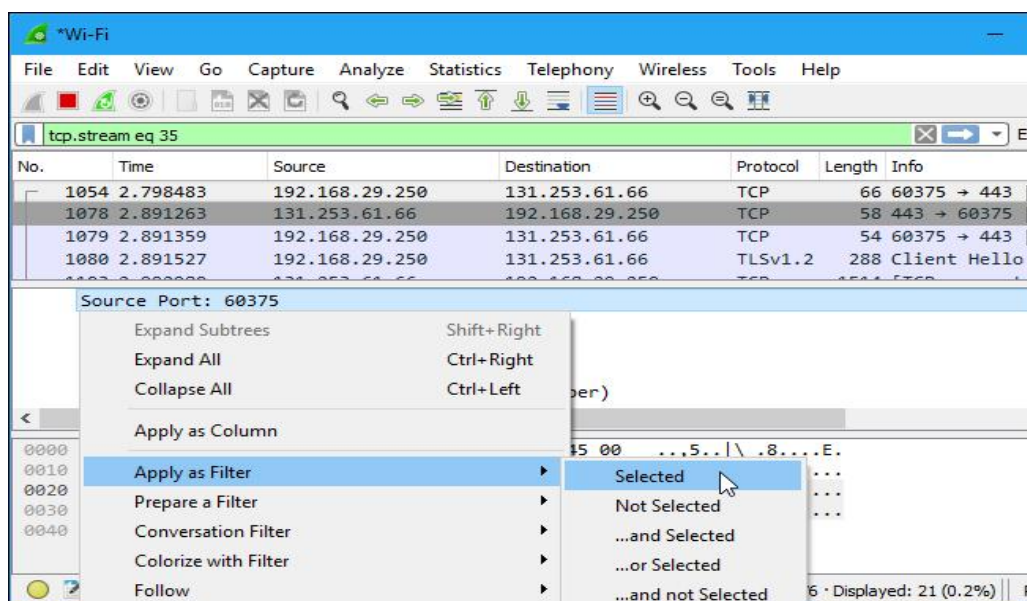


Shivajirao S Jondhale College of Engineering, Dombivli (E)

Department of Computer Engineering



You can also create filters from here — just right-click one of the details and use the Apply as Filter submenu to create a filter based on it.



Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

CONCLUSION: Thus, we have studied the working of Wire Shark.



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering
EXPERIMENT NO - 08

AIM:

Perform network discovery using discovery tools (eg. Nmap, mrtg)

Theory :

Nmap (Network Mapper) is a security scanner originally written by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich) used to discover hosts and services on a computer network, thus creating a "map" of the network. To accomplish its goal, Nmap sends specially crafted packets to the target host and then analyzes the responses. Unlike many simple port scanners that just send packets at some predefined constant rate, Nmap accounts for the network conditions (latency fluctuations, network congestion, the target interference with the scan) during the run. Also, owing to the large and active user community providing feedback and contributing to its features, Nmap has been able to extend its discovery capabilities beyond simply figuring out whether a host is up or down and which ports are open and closed; it can determine the operating system of the target, names and versions of the listening services, estimated uptime, type of device, and presence of a firewall.

Nmap features include:

- Host Discovery – Identifying hosts on a network. For example, listing the hosts which respond to pings or have a particular port open.
- Port Scanning – Enumerating the open ports on one or more target hosts.
- Version Detection – Interrogating listening network services listening on remote devices to determine the application name and version number.
- OS Detection – Remotely determining the operating system and some hardware characteristics of network devices.

Basic commands working in Nmap:

- For target specifications: `nmap<target's URL or IP with spaces between them>`
- For OS detection: `nmap -O <target-host's URL or IP>`
- For version detection: `nmap -sV<target-host's URL or IP>`



Shivajirao S Jondhale College of Engineering, Dombivli (E) Department of Computer Engineering

SYN scan is the default and most popular scan option for good reasons. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by restrictive firewalls. It is also relatively unobtrusive and stealthy since it never completes TCP connections

Algorithm\Implementation Steps\Installation Steps:

1. Download Nmap from www.nmap.org and install the Nmap Software with WinPcapDriver utility.

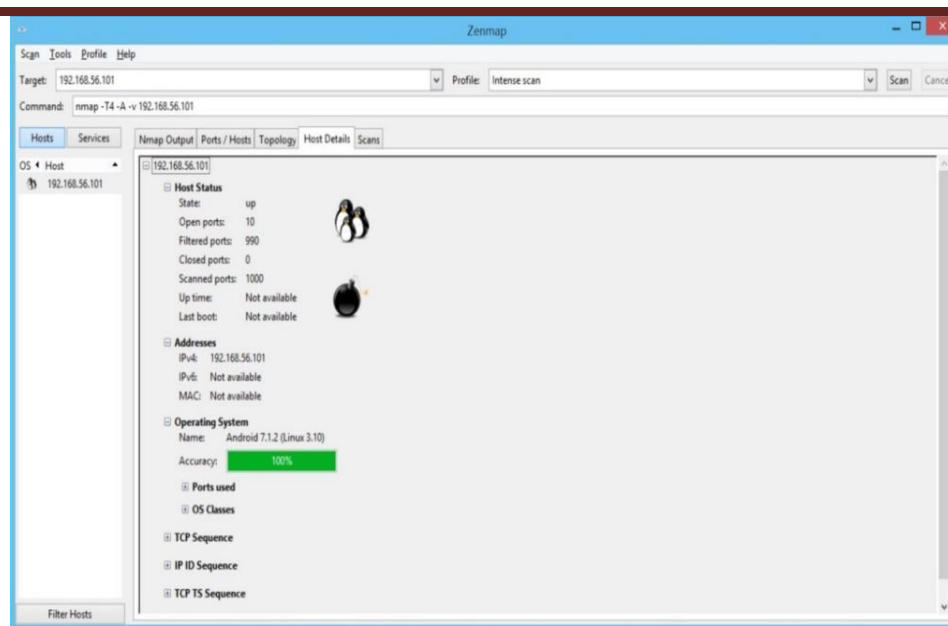
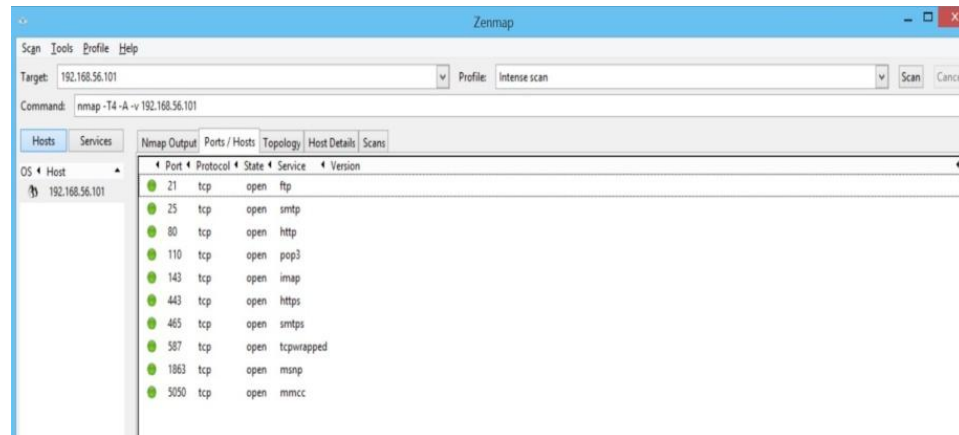
1. Execute the Nmap-Zenmap GUI tool from Program Menu or Desktop Icon
2. Type the Target Machine IP Address(ie.GuestOS or any website Address)
- 3.Perform the profiles shown in the utility.

```
Starting Nmap 7.80 ( https://nmap.org ) at 2019-09-24 09:56 India Standard Time
NSE: Loaded 151 scripts for scanning:
NSE: Script Pre-scanning.
Initiating NSE at 09:56
Completed NSE at 09:56, 0.00s elapsed
Initiating NSE at 09:56
Completed NSE at 09:56, 0.00s elapsed
Initiating NSE at 09:56
Completed NSE at 09:56, 0.00s elapsed
Initiating Ping Scan at 09:56
Completed Ping Scan at 09:56, 0.21s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 09:56
Completed Parallel DNS resolution of 1 host. at 09:56, 0.02s elapsed
Initiating SYN Stealth Scan at 09:56
Scanning 192.168.56.101 [1000 ports]
Discovered open port 25/tcp on 192.168.56.101
Discovered open port 80/tcp on 192.168.56.101
Discovered open port 135/tcp on 192.168.56.101
Discovered open port 443/tcp on 192.168.56.101
Discovered open port 465/tcp on 192.168.56.101
Discovered open port 587/tcp on 192.168.56.101
Discovered open port 593/tcp on 192.168.56.101
Discovered open port 595/tcp on 192.168.56.101
Completed SYN Stealth Scan at 09:56, 6.80s elapsed (1000 total ports)
Initiating Service scan at 09:56
Scanning 18 services on 192.168.56.101
Service scan timing: About 60.00% done; ETC: 09:58 (0:01:00 remaining)
Service scan timing: About 60.00% done; ETC: 10:00 (0:01:44 remaining)
Completed Service scan at 09:59, 156.51s elapsed (18 services on 1 host)
Initiating OS detection (try #1) against 192.168.56.101
Initiating Traceroute at 09:59
Initiating Parallel DNS resolution of 1 host. at 09:59
Completed Parallel DNS resolution of 1 host. at 09:59, 0.00s elapsed
NSE: Script scanning 192.168.56.101.
Initiating NSE at 09:59
Completed NSE at 10:00, 169.07s elapsed
Initiating NSE at 10:00
Completed NSE at 10:04, 156.88s elapsed
Initiating NSE at 10:04
Completed NSE at 10:04, 0.00s elapsed
Nmap scan report for 192.168.56.101
Host is up (0.001s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE
25/tcp    open  smtp
80/tcp    open  http
135/tcp   open  msrpc
443/tcp   open  https
465/tcp   open  smtps
587/tcp   open  smtp
593/tcp   open  msrpc
595/tcp   open  msrpc
WARNING: OS scan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: phone/general purpose
Running: Google Android 7.x, Linux 3.x|2.6.x
OS CPE: cpe:/o:google:android:7.1.2 cpe:/o:linux:linux_kernel:3.10 cpe:/o:linux:linux_kernel:2.6
OS details: Android 7.1.2 (Linux 3.10), Linux 2.6.18 - 2.6.22
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=259 (Good luck!)
IP ID Sequence Generation: All zeros
```



Shivajirao S Jondhale College of Engineering, Dombivli (E)

Department of Computer Engineering



CONCLUSION: Thus, we have studied different options to scan ports in Nmap.



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering
EXPERIMENT NO - 09

Aim:

To install and configure NS2 in Linux environment.

Theory:

NS(Network Simulator)

NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. [NS](#) is an event driven network simulator developed at UC Berkeley that simulates variety of IP networks. It implements network protocols such as TCP and UDP, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations.

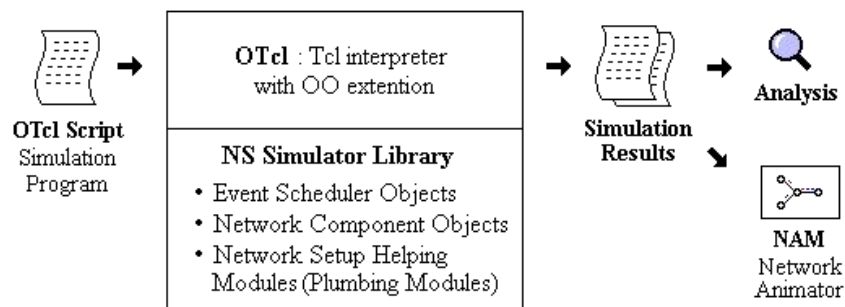


Figure 1. Simplified User's View of NS

As shown in Figure 1, in a simplified user's view, NS is Object-oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries, and network setup (plumbing) module libraries (actually, plumbing modules are implemented as member functions of the base simulator object). In other words, to use NS, you program in OTcl script language. To setup and run a simulation network, a user should write an OTcl script that initiates an event scheduler, sets up the network topology using the network objects and the plumbing functions in the library, and tells traffic sources when to start and stop transmitting packets through the event scheduler. The term "plumbing" is used for a network setup, because setting up a network is plumbing possible data paths among network objects by setting the "neighbor" pointer of an object to the address of an appropriate object. When a



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering

user wants to make a new network object, he or she can easily make an object either by writing a new object or by making a compound object from the object library, and plumb the data path through the object. This may sound like complicated job, but the plumbing OTcl modules actually make the job very easy. As shown in Figure 1, when a simulation is finished, NS produces one or more text-based output files that contain detailed simulation data, if specified to do so in the input Tcl (or more specifically, OTcl) script. The data can be used for simulation analysis (two simulation result analysis examples are presented in later sections) or as an input to a graphical simulation display tool called [Network Animator \(NAM\)](#) that is developed as a part of VINT project. NAM has a nice graphical user interface similar to that of a CD player (play, fast forward, rewind, pause and so on), and also has a display speed controller. Furthermore, it can graphically present information such as throughput and number of packet drops at each link, although the graphical information cannot be used for accurate simulation analysis.

Another major component of NS beside network objects is the event scheduler. An event in NS is a packet ID that is unique for a packet with scheduled time and the pointer to an object that handles the event. In NS, an event scheduler keeps track of simulation time and fires all the events in the event queue scheduled for the current time by invoking appropriate network components, which usually are the ones who issued the events, and let them do the appropriate action associated with packet pointed by the event. Network components communicate with one another passing packets, however this does not consume actual simulation time. All the network components that need to spend some simulation time handling a packet (i.e. need a delay) use the event scheduler by issuing an event for the packet and waiting for the event to be fired to itself before doing further action handling the packet.

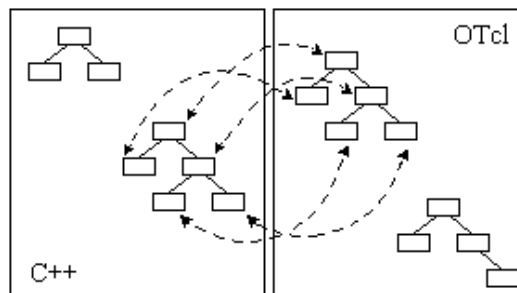


Figure 2. C++ and OTcl: The Duality



Shivajirao S Jondhale College of Engineering, Dombivli (E) Department of Computer Engineering

NS is written not only in OTcl but in C++ also. For efficiency reason, NS separates the data path implementation from control path implementations. In order to reduce packet and event processing time (not simulation time), the event scheduler and the basic network component objects in the data path are written and compiled using C++. These compiled objects are made available to the OTcl interpreter through an OTcl linkage that creates a matching OTcl object for each of the C++ objects and makes the control functions and the configurable variables specified by the C++ object act as member functions and member variables of the corresponding OTcl object. In this way, the controls of the C++ objects are given to OTcl. It is also possible to add member functions and variables to a C++ linked OTcl object. The objects in C++ that do not need to be controlled in a simulation or internally used by another object do not need to be linked to OTcl. Likewise, an object (not in the data path) can be entirely implemented in OTcl. Figure 2 shows an object hierarchy example in C++ and OTcl. One thing to note in the figure is that for C++ objects that have an OTcl linkage forming a hierarchy, there is a matching OTcl object hierarchy very similar to that of C++.

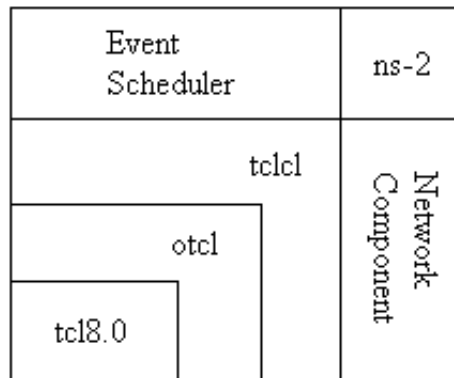


Figure 3. Architectural View of NS

Figure 3 shows the general architecture of NS. In this figure a general user (not an NS developer) can be thought of standing at the left bottom corner, designing and running simulations in Tcl using the simulator objects in the OTcl library. The event schedulers and most of the network components are implemented in C++ and available to OTcl through an OTcl linkage that is implemented using tclcl. The whole thing together makes NS, which is a OO extended Tcl interpreter with network simulator libraries.



Installation and configuration of NS2 in LINUX environment

We assume that you have already installed fully functional Linux desktop computer with Gnome. In our setup we are going to use ubuntu 9.10 (desktop edition). This is 32 bit environment

1. Download the stable release of ns2

- root@ubuntu:~# mkdir ns2
- root@ubuntu:~# cd ns2/
- root@ubuntu:~/ns2# wget <http://citylan.dl.sourceforge.net/project/nsnam/allinone/ns-allinone-2.34/ns-allinone-2.34.tar.gz>

2. Extract this Tar File

- root@ubuntu:~/ns2# tar -xzf ns-allinone-2.34.tar.gz

Now you will have this directory ns-allinone-2.34.

3. Configuration and Installation of ns2

- root@ubuntu:~/ns2# cd ns-allinone-2.34
- root@ubuntu:~/ns2/ns-allinone-2.34# apt-get install autoconf
- root@ubuntu:~/ns2/ns-allinone-2.34# apt-get install libc6-dev g++ gcc
- root@ubuntu:~/ns2/ns-allinone-2.34# apt-get install build-essential
- root@ubuntu:~/ns2/ns-allinone-2.34# apt-get install libx11-dev
- root@ubuntu:~/ns2/ns-allinone-2.34# apt-get install x-dev
- root@ubuntu:~/ns2/ns-allinone-2.34# apt-get install xorg-dev

Need to add patch for OTCL

- For Ubuntu 9.10 (karmic), you may encounter this error in the linking of otcl: So,
- root@ubuntu:~/ns2/ns-allinone-2.34# cd otcl-1.13
- root@ubuntu:~/ns2/ns-allinone-2.34# cp configure configure.orig
- Modify the following line
- --- configure.orig 2009-11-02 12:14:52.556167945 -0800
- +++ configure 2009-11-02 12:17:28.966706099 -0800
- @@ -6301,7 +6301,7 @@
- ;;



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering

- Linux*)
- SHLIB_CFLAGS="-fpic"
- SHLIB_LD="ld -shared"
- + SHLIB_LD="gcc -shared"
- SHLIB_SUFFIX=".so"
- DL_LIBS="-ldl"
- SHLD_FLAGS=""
- root@ubuntu:~/ns2/ns-allinone-2.34# ./install

By executing this command all of the required ns2 software will be installed except xgraph. For xgraph installation execute the following command. By the way, I used gnuplot, gnuplot has more advanced features. So,

- root@ubuntu:~/ns2# apt-get install xgraph
- root@ubuntu:~/ns2/ns-allinone-2.34# apt-get install gnuplot

4. Export System Variables

Execute the following commands

- root@ubuntu:~/ns2/ns-allinone-2.34# cp /root/.bashrc /root/.bashrc.bk
- root@ubuntu:~/ns2/ns-allinone-2.34# vi /root/.bashrc
- export PATH=\$PATH:/root/ns2/ns-allinone-2.34/bin:/root/ns2/ns-allinone-2.34/tcl8.4.18/unix:/root/ns2/ns-allinone-2.34/tk8.4.18/unix
- export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:/root/ns2/ns-allinone-2.34/otcl-1.13:/root/ns2/ns-allinone-2.34/lib
- export TCL_LIBRARY="/root/ns2/ns-allinone-2.34/tcl8.4.18/library"

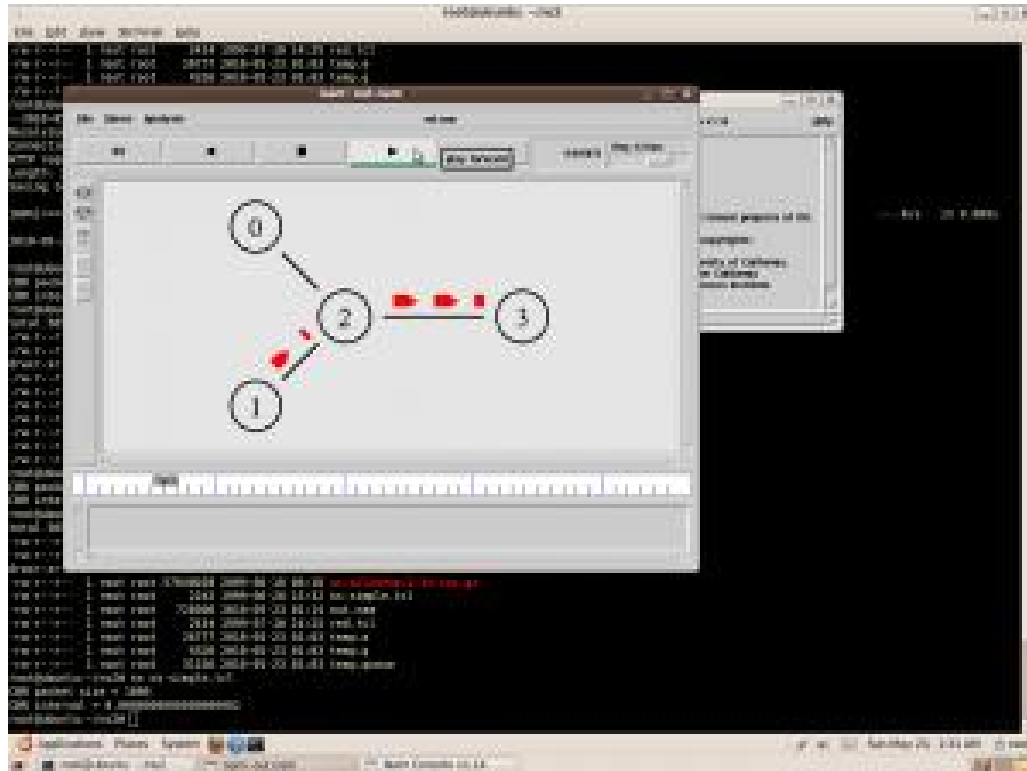
5. NS2 Testing

Now, your ns2 system is ready for use, download a test script and test it.

- root@ubuntu:~/ns2# wget <http://nile.wpi.edu/NS/Example/ns-simple.tcl>
- root@ubuntu:~/ns2# ns ns-simple.tcl



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering



NS-2-Demonstration

Conclusion: - Hence we studied NS (Version 2) Network Simulator installation and configuration in Linux environment.



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering
EXPERIMENT NO - 10

Aim:

To study & implement shortest path routing.

Theory:

One of the important function of network layer is to route the packet from source to destination machine routing algorithm is responsible for deciding the output line over Which packet is to be sent

Static routing algorithm is a type of algorithm where routing decision is not based on the measurement or estimation of current traffic & topology .The routes is decided in advances ,offline & it is downloaded to the routers .It is also called as non-adaptive algorithm.

Shortest path routing is an example of static algorithm .A graph of subnet is built in which each representation os router & link or connection line is present. To choose path between paor of routers,shortest path is found by measuring the path length.The metric can be number of hops or geographical distance in kilometers

Dijkstra's Algorithm:

This algorithm is used for computing the shortest path from root node to every other node to every other node in the network.

The total number of nodes are divided into 2 groups namely P&T.In P we have nodes for which shortest path in already found in T the remaining nodes are placed.

At the time of sorting ,P is initialized to the current node & T is initialized to null .The algorithm then repeats these steps.

- 1)Shortest path describe node say P.Write P in the P set.
- 2)For this node P,add it's neighbour n to T set .The addition of these will have to satisfy following conditions:
 - a)If the neighbouring node is not present in T,then add it adding with the cost to reach it through P.
 - b)It n is already present in T & the path to n through P has lower cost ,then removed earlier instances of n &add new instances connected with the cost to reach through P.
- 3)Pick neighbour n which has the smallest value in T & If it is not present in T then add it to P.Use it annotation to determine rater P to released A.



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering

4) Stop When T is Empty.

The goal of this algorithm is to find out every possible way in which an outside node can be reached by a node already present in P & choose the shortest of these as path to desired node

Example

Q) show computations at node A using Dijkstra algorithm

Add Starting node A is P(permanent) group. Add neighbour nodes B & D in T(temporary) group along with their costs to reach them through A.

Pick up neighbour to T-group, continue till last node to obtain find table

Permanent (P)

Temporary(T)

1) A

B(A,1), D(A,2)

2) A, B(A,1)

D(A,2), C(B,2)

3) A, B(A,1), D(A,2)

E(D,4), C(B,2)

3) A, B(A,1), D(A,2), C(B,2)

E(C,3), E(C,4)

cannot be include

4) A < B(A,1), D(A,2), C(B,2), E(1,3)

F(E,6), F(E,7) cannot

be included

5) A, B(A,1), D(A,2), C(B,2), E(C,3), F(E,6)

Empty (null)

The shortest path to other nodes from A is

Conclusion:

Hence, we have studied & implemented shortest path routing using Dijkstra's Algorithm.

Program

```
import java.util.*;
public class Dijkstra
{
    public int distance[]=new int[10];
    public int cost[][]=new int [10][10];
    public void calc(int n,int s)
    {
        int flag[]=new int[n+1];
        int i,minpos=1,k,cminimum;
        for(i=1;i<=n;i++)
```



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering

```
{
    flag[i]=0;
    this.distance[i]=this.cost[s][i];
}
c=2;
while(c<=n)
{
    minimum=99;
    for(k=1;k<=n;k++)
    {
        if(this.distance[k]<minimum && flag[k]!=1)
        {
            minimum=this.distance[i];
            minpos=k;
        }
    }
    flag[minpos]=1;
    c++;
    for(k=1;k<=n;k++)
    {
        if(this.distance[minpos]+this.cost[minpos][k]<
        this.distance[k] && flag[k]!=1)
        this.distance[k]=this.distance[minpos]+this.cost[minpos][k];
    }
}

public static void main(String args[])
{
    int nodes,source,i,j;
    Scanner in=new Scanner(System.in);
    System.out.println("Enter the Number of Nodes \n");
    nodes=in.nextInt();
    Dijkstra d=new Dijkstra();
    System.out.println("Enter the cost matrix weight: \n");
```



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering

```
for(i=1;i<=nodes;i++)
for(j=1;j<=nodes;j++)
{
d.cost[i][j]=in.nextInt();
if(d.cost[i][j]==0);
d.cost[i][j]=999;
}
System.out.println("Enter the source vertex:\n");
source=in.nextInt();
d.calc(nodes,source);
System.out.println("The Shortest Path from Source\t"+source+"\t to all other
vertices are :\n");
for(i=1;i<=nodes;i++)
if(i!=source);
System.out,println("Source:"+source+"\t destination :"+i+"\t Mincost
is :"+d.distance[i)+"\t");
}
}
```

*****Result*****

Enter the Number of nodes

8

Enter the Cost Matrix Weights:

0 2 0 0 0 0 6 0

2 0 7 0 2 0 0 0

0 7 0 3 0 3 0 0

0 0 3 0 0 0 0 2

0 2 0 0 0 2 1 0

0 0 3 0 2 0 0 2

6 0 0 0 1 0 0 4

0 0 0 2 0 2 4 0

Enter the Source Vertex :

1

The Shortest Path from Source 1 to all other vertices are:



Shivajirao S Jondhale College of Engineering, Dombivli (E)
Department of Computer Engineering

source :1	destination :2	MinCost is :2
source :1	destination :3	MinCost is :9
source :1	destination :4	MinCost is :12
source :1	destination :5	MinCost is :4
source :1	destination :6	MinCost is :6
source :1	destination :7	MinCost is :5
source :1	destination :8	MinCost is :8