Maharaja Surajmal Institute of Technology

Wireless Communications Practical Lab File

CSE – Evening (7th Semester)

Submitted To :	Submitted By :
Dr Bharti Sharma	Bhavya Khanna
	00696302717
	S.no. - 07

TABLE OF CONTENTS

EXPERIMENT - 1

AIM : Write an experiment to study the concept of Frequency Reuse.

THEORY :

Frequency reuse, or, frequency planning, is a technique of reusing frequencies and channels within a communication system to improve capacity and spectral efficiency. Frequency reuse is one of the fundamental concepts on which commercial wireless systems are based that involve the partitioning of an RF radiating area into cells. The increased capacity in a commercial wireless network, compared with a network with a single transmitter, comes from the fact that the same radio frequency can be reused in a different area for a completely different transmission.

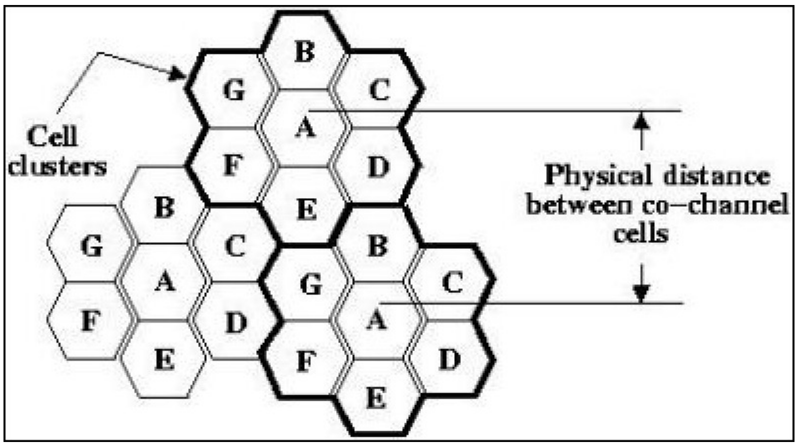


Figure 1: Frequency reuse technique of a cellular system.

Frequency reuse in mobile cellular systems means that frequencies allocated to the service are reused in a regular pattern of cells, each covered by one base station.

The repeating regular pattern of cells is called cluster. Since each cell is designed to use radio frequencies only within its boundaries, the same frequencies can be reused in other cells not far away without interference, in another cluster. Such cells are called ‘co-channel’ cells. The reuse of frequencies enables a cellular system to handle a huge number of calls with a limited number of channels. Figure 1 shows a frequency planning with cluster size of 7, showing the co-channels cells in different clusters by the same letter. The closest distance between the co- channel cells (in different clusters) is determined by the choice of the cluster size and the layout of the cell cluster. Consider a cellular system with S duplex channels available for use and let N be the number of cells in a cluster. If each cell is allotted K duplex channels with all being allotted unique and disjoint channel groups we have S = KN under normal circumstances. Now, if the cluster are repeated M times within the total area, the total number of duplex channels, or, the total number of users in the system would be T = MS = KMN. Clearly, if K and N remain constant, then

$$T \propto M$$

and, if T and K remain constant, then

$$N \propto 1/M$$

Hence the capacity gain achieved is directly proportional to the number of times a cluster is repeated, as well as, for a fixed cell size, small N decreases the size of the cluster with in turn results in the increase of the number of clusters and hence the capacity. However for small N, co-channel cells are located much closer and hence more interference. The value of N is determined by calculating the amount of interference that can be tolerated for a sufficient quality communication. Hence the smallest N having interference below the tolerated limit is used. However, the cluster size N cannot take on any value and is given only by the following equation

$$N = i^2 + ij + j^2, \qquad i \geq 0, j \geq 0, \text{(where i and j are integer numbers)}$$

CHANNEL ASSIGNMENT STRATEGIES

With the rapid increase in the number of mobile users, the mobile service providers had to follow strategies which ensure the effective utilization of the limited radio spectrum. With increased capacity and low interference being the prime objectives, a frequency reuse scheme was helpful in achieving these objectives. A variety of channel assignment strategies have been followed to aid these objectives. Channel assignment strategies are classified into two types: fixed and dynamic, as discussed below.

Fixed Channel Assignment (FCA)

In the fixed channel assignment strategy each cell is allocated a fixed number of voice channels. Any communication within the cell can only be made with the designated unused channels of that particular cell. Suppose if all the channels are occupied, then the call is blocked and the subscriber has to wait. This is simplest of the channel assignment strategies as it requires very simple circuitry but provides worst channel utilization. Later there was another approach in which the channels were borrowed from adjacent cells if all of its own designated channels were occupied. This was named as a borrowing strategy. In such cases the MSC supervises the borrowing process and ensures that none of the calls in progress are interrupted.

Dynamic Channel Assignment (DCA)

In dynamic channel assignment strategy channels are temporarily assigned for use in cells for the duration of the call. Each time a call attempt is made from a cell the corresponding BS requests a channel from MSC. The MSC then allocates a channel to the requesting BS. After the call is over the channel is returned and kept in a central pool. To avoid co-channel interference any channel that is in use in one cell can only be reassigned simultaneously to another cell in the system if the distance between the two cells is larger than minimum reuse distance. When compared to the FCA, DCA has reduced the likelihood of blocking and even increased the trunking capacity of the network as all of the channels are available to all cells, i.e., good quality of service. But this type of assignment strategy results in a heavy load on switching centers at heavy traffic conditions.

INTERFERENCE & SYSTEM CAPACITY

Co-channel interference (CCI)

Co-channel interference is the cross talk between two different radio transmitters using the same radio frequency as is the case with the co-channel cells. The reasons for CCI can be because of either adverse weather conditions or poor frequency planning or overly crowded radio spectrum.

Adjacent Channel Interference (ACI)

This is a different type of interference which is caused by adjacent channels i.e. channels in adjacent cells. It is the signal impairment which occurs to one frequency due to presence of another signal on a nearby frequency. This occurs when imperfect receiver filters allow nearby frequencies to leak into the passband. This problem is enhanced if the adjacent channel user is transmitting in a close range compared to the subscriber’s receiver while the receiver attempts to receive a base station on the channel. This is called the near-far effect. The more adjacent channels are packed into the channel block, the higher the spectral efficiency, provided that the performance degradation can be tolerated in the system link budget. This effect can also occur if a mobile close to a base station transmits on a channel close to one being used by a weak mobile. This problem might occur if the base station has a problem in discriminating the mobile user from the “bleed over” caused by the close adjacent channel mobile.

ENHANCING CAPACITY AND CELL COVERAGE

Cell-Splitting

Cell Splitting is based on the cell radius reduction and minimizes the need to modify the existing cell parameters. Cell splitting involves the process of subdividing a congested cell into smaller cells, each with its own base station and a corresponding reduction in antenna size and transmitting power. This increases the capacity of a cellular system since it increases the number of times that channels are reused. Since the new cells have smaller radii than the existing cells, inserting these smaller cells, known as microcells, between the already existing cells results in an increase of capacity due to the additional number of channels per unit area.

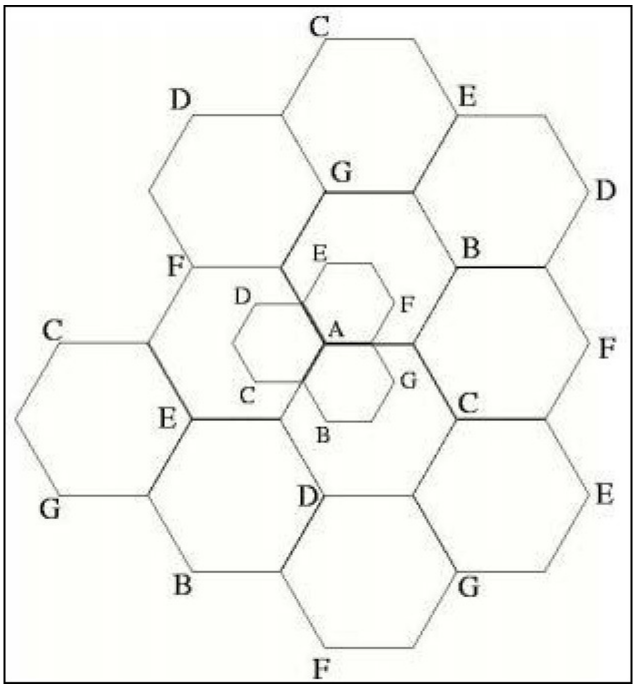


Figure 2: Splitting of congested seven-cell clusters.

Sectoring

Sectoring is basically a technique which can increase the SIR without necessitating an increase in the cluster size. Till now, it has been assumed that the base station is located in the center of a cell and radiates uniformly in all the directions behaving as an omni-directional antenna. However, it has been found that the co-channel interference in a cellular system may be decreased by replacing a single omni-directional antenna at the base station by several directional antennas, each radiating within a specified sector.

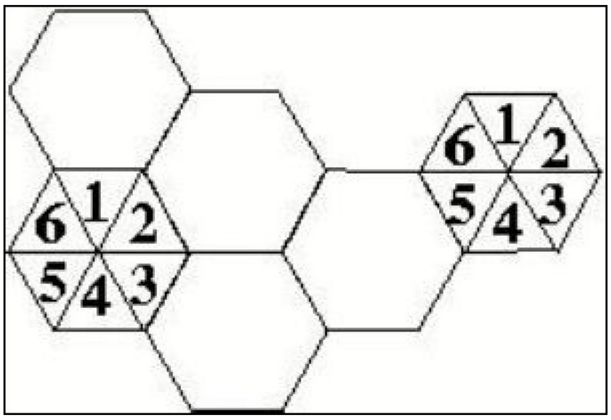


Figure 3: A seven-cell cluster with 60^o sectors.

EXPERIMENT - 2

AIM : Write an experiment to study the concept and implementation of CDMA.

THEORY :

Code Division Multiple Access (CDMA) is a sort of multiplexing that facilitates various signals to occupy a single transmission channel. It optimizes the use of available bandwidth. The technology is commonly used in ultra high frequency (UHF) cellular telephone systems, bands ranging between the 80MHz and 1.9GHz. The CDMA system is very different from time and frequency multiplexing. In this system, a user has access to the whole bandwidth for the entire duration. The basic principle is that different CDMA codes are used to distinguish among the different users.

Technology generally used are direct sequence spread spectrum modulation (DS-CDMA), frequency hopping or mixed CDMA detection (JDCDMA). Here, a signal is generated which extends over a wide bandwidth. A code called spreading code is used to perform this action. Using a group of codes, which are orthogonal to each other, it is possible to select a signal with a given code in the presence of many other signals with different orthogonal codes.

SOURCE CODE :

```
import java.io.*;

class cdma1 {
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int dataa,datab;

int keya[]=new int [6]; int keyb[]=new int [6]; int key1[]=new int [6];
int key2[]=new int [6]; int key3[]=new int[6];
int ina ,inb;

public void getdata()throws IOException {

 System.out.println("A-Enter data bit");
ina=Integer.parseInt(br.readLine());

if(ina>1 ||ina<0)
        System.out.println("Error enter binary");
System.out.println("B-Enter data bit");
inb=Integer.parseInt(br.readLine());
if(inb>1 ||inb<0)
        Systemout.println("Error enter binary");
```

```
System.out.println("A-Enter the 6-bit binary key");
for(int i=0; i<6; i++) {
        keya[i]=Integer.parseInt(br.readLine());
        if(keya[i]>1 ||keya[i]<0)
                System.out.println("Error enter binary");
}

System.out.println("B-Enter the 6 bit binary key");
for(int i=0;i<6;i++) {
        keyb[i]=Integer.parseInt(br.readLine());
        if(keyb[i]>1 ||keyb[i]<0)
                System.out.println("Error enter binary");

}
}

 public void compute()throws IOException
 { if (ina==0) dataa = -1;
 else    dataa=1;
 if(inb==0) datab=-1;
 else datab=1;

for(int i=0;i<6;i++) {
        if(keya[i]==0) key1[i]= -1;
        else key1[i]=1;
}

for(int i=0;i<6;i++) {
        if(keyb[i]==0) key2[i]= -1;
        else key2[i]=1;
}
for(int i=0;i<6;i++) {
        keya[i]=key1[i]*dataa; keyb[i]=key2[i]*datab; key3[i]=keya[i]+keyb[i];
}

for(int i=0;i<6;i++) {
        keya[i]=key3[i]*key1[i]; keyb[i]=key3[i]*key2[i];
}

int totala=0; int totalb=0;
for(int i=0;i<6;i++) {
        totala=totala+keya[i];
        totalb=totalb+keyb[i];
 }

System.out.println("Transformed key a");
for(int i=0;i<6;i++)
        System.out.print(key1[i]+" ");

System.out.println("Transformed key b");
for(int i=0;i<6;i++)
        System.out.print(key2[i]+" ");
System.out.println("The sum of a is "+totala);
System.out.println("The sum of b is "+totalb);

if(totala>=0)
        System.out.println("The data entered by A is 1");
 else
        System.out.println("The data entered by A is 0");

        if(totalb>=0)
                System.out.println("The data entered by B is 1");
        else
                System.out.println("The data entered by B is 0");
}
}

class cdma {
        public static void main(String args[])throws IOException
                { cdma1 c=new cdma1();
                c.getdata();
                c.compute();
        }
}
```

OUTPUT :

```
C:\jdk1.3\bin>javac cdma.java
C:\jdk1.3\bin>java cdma

A-Enter data bit
1

B-Enter data bit
0

A-Enter the 6bit binary key
0 1 0 0 1 1

B-Enter the 6 bit binary key
1 1 0 1 0 1

Transformed key a
-1 1 -1 -1 1 1

Transformed key b
1 1 -1 1 -1 1

The sum of a is 6

The sum of b is -6

The data entered by A is 1

The data entered by B is 0
```

EXPERIMENT - 3

AIM : Write an experiment to create/setup wireless networks.

THEORY :

Getting Online

By far the most important element of your home network is the router. After purchasing a router, you need to get it connected and online. If your service provider gave you a modem when they activated your internet service, this should be pretty simple. Just follow these steps:

- Turn off your modem.
- Unplug the modem's Ethernet cable from the PC.
- Connect that cable to the WAN or Internet port on your new router.
- Power on your modem (wait for a minute or two).
- Next power on your router (wait for a minute or two).
- Now use another Ethernet cable to connect the PC to your router's LAN port, and turn on your PC.

By default, most consumer routers are set up to use DHCP to automatically assign an IP address to your computer. So if everything worked right, you should now be online.

Accessing the Management Console

With the router and PC physically connected, you can now begin customizing the router's configuration. Most routers these days are managed via a web browser and are shipped using a default IP address, administrator account, and password. This default IP address will vary from vendor to vendor, so check your documentation to find out yours. Once you have that information, accessing the management console is easy.

- Launch your Web browser; it doesn't matter which one you use.
- Enter the router's IP address, which will look something like 192.168.0.1, into the browser's address bar and press Enter.
- You will see the router's login screen asking for the default administrator username and password. After your supply those credentials and press Enter, you should be looking at the management console.

Managing your IP addresses with DHCP

Next it's time to focus on your router's LAN configuration. As I previously mentioned, DHCP, which manages all the IP addresses on your network, is typically enabled by default on most consumer routers. Your router uses the IP address, a numeric identifier, to locate your PC and route the correct network traffic.

If the computer or mobile device does not already have an IP address, it will request one from the network's DHCP server, which is on the router. The DHCP server pulls an IP address from a pool of available IP addresses (called a Scope) and assigns it to the device. When the device disconnects from the network, or a certain amount of time has passed (referred to as a lease) the IP address returns to the DHCP pool.

To set the DHCP scope, follow these steps:

- Log in to the router's management console and look for a heading like LAN Setup (or something similar). Here you should see your LAN's IP address and subnet mask along with a section for DHCP server.
- Assign a range of IP addresses for the DHCP server to use. Assuming your router's IP address is 192.168.0.1 and you wanted to assign 50 IP addresses to the DHCP scope, you would set the Starting IP address to 192.168.0.2 and the Ending address to 192.168.0.51.

Activating Your Wi-Fi

With your network now properly configured you can move on to setting up your wireless network. This is actually very simple and should only take you couple of minutes. When you do this, make sure you use a computer that is connected to the network via an Ethernet cable. If you try to configure Wi-Fi over wireless, you'll lose the connection to the management console whenever changes to the configuration are applied.

Now follow these steps:

- Go into the router's management console and locate the section titled Wireless Setup. It might be labeled differently depending on your router, but it should be pretty obvious which one it is.
- The wireless network should be enabled by default, but if it isn't, turn it on. If you have a dual-band router, you should see the configuration settings for both the 2.4Gz and 5GHz networks. Both need to be configured independently.
- Next make sure the Channel is set to Auto and leave the Mode in its default setting. You can adjust these settings later if you feel the network is sluggish or if you're experiencing dropped connections.

- This brings us to the SSID. The SSID is the name of your wireless network. You can name the network just about anything you want, and you definitely should not leave it set to the default value. Something descriptive is always good. For instance, we might use something along the lines of PCM24 and PCM50. PCM24 would be the name assigned to the 2,4GHz network and PCM50 would be the name assigned to the 5GHz network. PCM of course stands for PCMag.
- The final step is to set the encryption your Wi-Fi network will use. There are a few choices here. The only one you're interested in using is WPA2.
- There might be a few WPA2 options listed, but you want the one listed as WPA2- PSK [AES]. This is the highest level of wireless security currently available. Some routers still offer WEP. Do not use it as it is vulnerable to brute-force attacks.
- Once you've set your encryption type to WPA2, assign a passphrase (aka passcode or key). This Passphrase needs to be between 8 and 63 characters long and should be made up of both letters (both uppercase and lowercase), numbers and characters ($@#%&). The longer the key, the more secure. Passphrases made up of random characters, such as hy#Pnj!123 are the best, but if you have to use a name or something else familiar, make sure to mix it up with some numbers and characters as well (ex: P@assword20!5).

All that's left now is to save your configuration changes and test the connection. Your wireless devices should now be online.

EXPERIMENT - 4

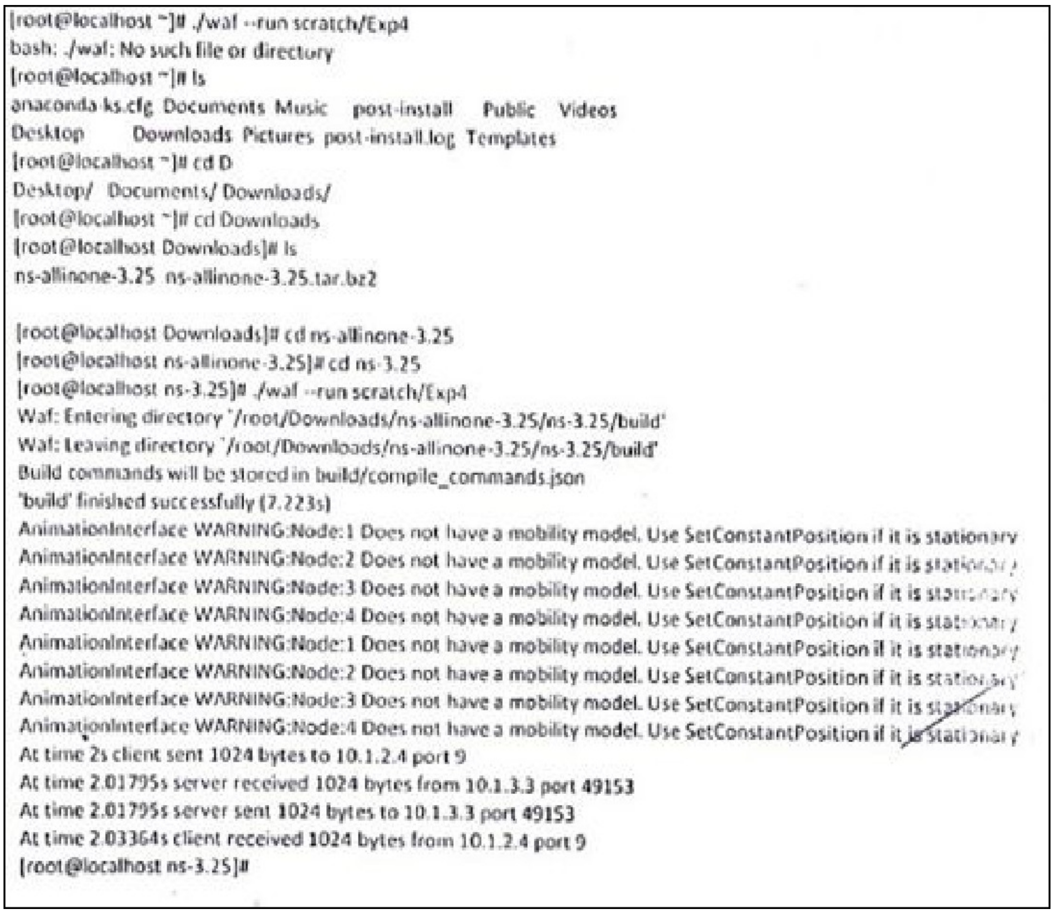
AIM : Write an experiment for Simulation in NS3 to connect.

SOURCE CODE :

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("ThirdScriptExample");
int main (int argc, char *argv[]) {
bool verbose = true; uint32_t nCsma = 3; uint32_t nWifi = 3; bool tracing = false;
CommandLine cmd;
cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
cmd.AddValue ("tracing", "Enable pcap tracing", tracing);
cmd.Parse (argc,argv);
if (nWifi > 18) {
        std::cout << "nWifi should be 18 or less, otherwise grid layout exceeds the bounding
        box" << std::endl;
        return 1;
}
if (verbose) {
        LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
        LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
}
NodeContainer p2pNodes;
p2pNodes.Create (2);
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);
NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsma);
CsmaHelper csma;
```

```
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);
NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());
WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac", "Ssid", SsidValue (ssid), "ActiveProbing", BooleanValue
(false)); NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);
mac.SetType ("ns3::ApWifiMac", "Ssid", SsidValue (ssid));
NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator", "MinX", DoubleValue (0.0),
"MinY", DoubleValue (0.0), "DeltaX", DoubleValue (5.0), "DeltaY", DoubleValue (10.0),
"GridWidth", UintegerValue (3), "LayoutType", StringValue ("RowFirst"));
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel", "Bounds",
RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);
InternetStackHelper stack;
stack.Install (csmaNodes); stack.Install (wifiApNode); stack.Install (wifiStaNodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
Simulator::Stop (Seconds (10.0));
if (tracing == true) {
        pointToPoint.EnablePcapAll ("third");
        phy.EnablePcap ("third", apDevices.Get (0));
        csma.EnablePcap ("third", csmaDevices.Get (0), true);
}
Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```

OUTPUT :



```
[root@localhost ~]# ./waf --run scratch/Exp4
bash: ./waf: No such file or directory
[root@localhost ~]# ls
anaconda-ks.cfg  Documents  Music     post-install      Public     Videos
Desktop          Downloads  Pictures  post-install.log  Templates
[root@localhost ~]# cd D
Desktop/  Documents/ Downloads/
[root@localhost ~]# cd Downloads
[root@localhost Downloads]# ls
ns-allinone-3.25  ns-allinone-3.25.tar.bz2

[root@localhost Downloads]# cd ns-allinone-3.25
[root@localhost ns-allinone-3.25]# cd ns-3.25
[root@localhost ns-3.25]# ./waf --run scratch/Exp4
Waf: Entering directory `/root/Downloads/ns-allinone-3.25/ns-3.25/build'
Waf: Leaving directory `/root/Downloads/ns-allinone-3.25/ns-3.25/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (7.223s)
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.01795s server received 1024 bytes from 10.1.3.3 port 49153
At time 2.01795s server sent 1024 bytes to 10.1.3.3 port 49153
At time 2.03364s client received 1024 bytes from 10.1.2.4 port 9
[root@localhost ns-3.25]#
```

EXPERIMENT - 5

AIM : Write an experiment to simulate adhoc networks using NS3.

SOURCE CODE :

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/stats-module.h"
#include "ns3/wifi-module.h"
#include <iostream>
NS_LOG_COMPONENT_DEFINE ("Main");
using namespace ns3;
class Experiment {
    public:
          Experiment (); Experiment (std::string name);
          Gnuplot2dDataset Run (const WifiHelper &wifi, const YansWifiPhyHelper &wifiPhy,
          const NqosWifiMacHelper &wifiMac, const YansWifiChannelHelper &wifiChannel);
    private:
          void ReceivePacket (Ptr<Socket> socket);
          void SetPosition (Ptr<Node> node, Vector position);
          Vector GetPosition (Ptr<Node> node);
          void AdvancePosition (Ptr<Node> node);
          Ptr<Socket> SetupPacketReceive (Ptr<Node> node);
          uint32_t m_bytesTotal; Gnuplot2dDataset m_output;
};
Experiment::Experiment () {}
Experiment::Experiment (std::string name): m_output (name){
          m_output.SetStyle (Gnuplot2dDataset::LINES);
}
void Experiment::SetPosition (Ptr<Node> node, Vector position){
          Ptr<MobilityModel> mobility = node->GetObject<MobilityModel> ();
          mobility->SetPosition (position);
}
Vector Experiment::GetPosition (Ptr<Node> node) {
          Ptr<MobilityModel> mobility = node->GetObject<MobilityModel> ();
          return mobility->GetPosition ();
}
void Experiment::AdvancePosition (Ptr<Node> node) {
          Vector pos = GetPosition (node);
          double mbs = ((m_bytesTotal * 8.0) / 1000000);
          m_bytesTotal = 0; m_output.Add (pos.x, mbs);
```

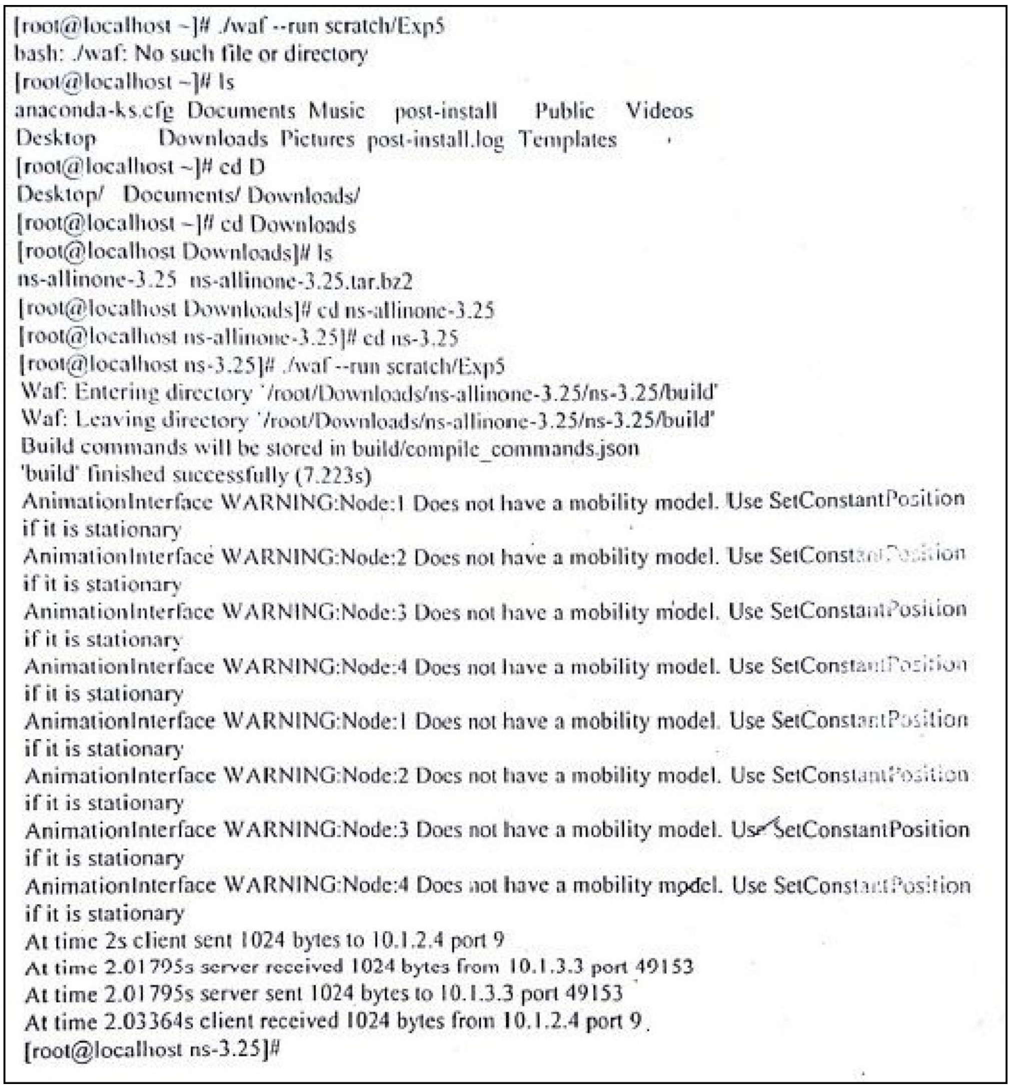
```
         pos.x += 1.0;
        if (pos.x >= 210.0) { return; }
        SetPosition (node, pos);
        Simulator::Schedule (Seconds (1.0), &Experiment::AdvancePosition, this, node);
}
void Experiment::ReceivePacket (Ptr<Socket> socket) {
        Ptr<Packet> packet;
        while ((packet = socket->Recv ())) { m_bytesTotal += packet->GetSize (); }
}
Ptr<Socket>Experiment::SetupPacketReceive (Ptr<Node> node){
        TypeId tid = TypeId::LookupByName ("ns3::PacketSocketFactory");
        Ptr<Socket> sink = Socket::CreateSocket (node, tid);
        sink->Bind ();
        sink->SetRecvCallback (MakeCallback (&Experiment::ReceivePacket, this));
        return sink;
}
Gnuplot2dDataset Experiment::Run (const WifiHelper &wifi, const YansWifiPhyHelper
&wifiPhy, const NqosWifiMacHelper &wifiMac, const YansWifiChannelHelper
&wifiChannel) {
m_bytesTotal = 0; NodeContainer c; c.Create (2);
PacketSocketHelper packetSocket;
packetSocket.Install (c);
YansWifiPhyHelper phy = wifiPhy;
phy.SetChannel (wifiChannel.Create ());
NqosWifiMacHelper mac = wifiMac;
NetDeviceContainer devices = wifi.Install (phy, mac, c);
MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
positionAlloc->Add (Vector (0.0, 0.0, 0.0));
positionAlloc->Add (Vector (5.0, 0.0, 0.0));
mobility.SetPositionAllocator (positionAlloc);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (c);
PacketSocketAddress socket;
socket.SetSingleDevice (devices.Get (0)->GetIfIndex ());
socket.SetPhysicalAddress (devices.Get (1)->GetAddress ());
socket.SetProtocol (1);
OnOffHelper onoff ("ns3::PacketSocketFactory", Address (socket));
onoff.SetConstantRate (DataRate (60000000));
onoff.SetAttribute ("PacketSize", UintegerValue (2000));
ApplicationContainer apps = onoff.Install (c.Get (0));
apps.Start (Seconds (0.5));
apps.Stop (Seconds (250.0));
```

```
Simulator::Schedule (Seconds (1.5), &Experiment::AdvancePosition, this, c.Get (1));
Ptr<Socket> recvSink = SetupPacketReceive (c.Get (1));
Simulator::Run (); Simulator::Destroy (); return m_output;
}

int main (int argc, char *argv[]) {
Config::SetDefault ("ns3::FragmentationThreshold", StringValue ("2200"));
Config::SetDefault ("ns3::RtsCtsThreshold", StringValue ("2200"));
CommandLine cmd; cmd.Parse (argc, argv);
Gnuplot gnuplot = Gnuplot ("reference-rates.png");
Experiment experiment;
WifiHelper wifi = WifiHelper::Default ();
wifi.SetStandard (WIFI_PHY_STANDARD_80211a);
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
Gnuplot2dDataset dataset;
wifiMac.SetType ("ns3::AdhocWifiMac");
NS_LOG_DEBUG ("54");
experiment = Experiment ("54mb");
wifi.SetRemoteStationManager ("ns3::", "DataMode", StringValue ("OfdmRate54Mbps"));
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("48");
experiment = Experiment ("48mb");
wifi.SetRemoteStationManager ("ns3::", "DataMode", StringValue ("OfdmRate48Mbps"));
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("36");
experiment = Experiment ("36mb");
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
("OfdmRate36Mbps"));
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("24");
experiment = Experiment ("24mb");
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
("OfdmRate24Mbps"));
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("18");
experiment = Experiment ("18mb");
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
("OfdmRate18Mbps"));
```

```
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("12");
experiment = Experiment ("12mb");
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
("OfdmRate12Mbps"));
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("9");
experiment = Experiment ("9mb");
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
("OfdmRate9Mbps"));
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("6");
experiment = Experiment ("6mb");
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
"DataMode", StringValue ("OfdmRate6Mbps"));
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
gnuplot.GenerateOutput (std::cout);
gnuplot = Gnuplot ("rate-control.png");
wifi.SetStandard (WIFI_PHY_STANDARD_holland);
NS_LOG_DEBUG ("arf");
experiment = Experiment ("arf");
wifi.SetRemoteStationManager ("ns3::ArfWifiManager");
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("aarf");
experiment = Experiment ("aarf");
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("aarf-cd");
experiment = Experiment ("aarf-cd");
wifi.SetRemoteStationManager ("ns3::AarfcdWifiManager");
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("cara");
experiment = Experiment ("cara");
wifi.SetRemoteStationManager ("ns3::CaraWifiManager");
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
```

```
NS_LOG_DEBUG ("rraa");
experiment = Experiment ("rraa");
wifi.SetRemoteStationManager ("ns3::RraaWifiManager");
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
NS_LOG_DEBUG ("ideal");
experiment = Experiment ("ideal");
wifi.SetRemoteStationManager ("ns3::IdealWifiManager");
dataset = experiment.Run (wifi, wifiPhy, wifiMac, wifiChannel);
gnuplot.AddDataset (dataset);
gnuplot.GenerateOutput (std::cout);
return 0;
}
```

OUTPUT :



```
[root@localhost ~]# ./waf --run scratch/Exp5
bash: ./waf: No such file or directory
[root@localhost ~]# ls
anaconda-ks.cfg  Documents  Music     post-install     Public     Videos
Desktop          Downloads  Pictures  post-install.log  Templates
[root@localhost ~]# cd D
Desktop/  Documents/ Downloads/
[root@localhost ~]# cd Downloads
[root@localhost Downloads]# ls
ns-allinone-3.25  ns-allinone-3.25.tar.bz2
[root@localhost Downloads]# cd ns-allinone-3.25
[root@localhost ns-allinone-3.25]# cd ns-3.25
[root@localhost ns-3.25]# ./waf --run scratch/Exp5
Waf: Entering directory '/root/Downloads/ns-allinone-3.25/ns-3.25/build'
Waf: Leaving directory '/root/Downloads/ns-allinone-3.25/ns-3.25/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (7.223s)
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition
if it is stationary
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.01795s server received 1024 bytes from 10.1.3.3 port 49153
At time 2.01795s server sent 1024 bytes to 10.1.3.3 port 49153
At time 2.03364s client received 1024 bytes from 10.1.2.4 port 9
[root@localhost ns-3.25]#
```

EXPERIMENT - 6

AIM : Simulation in NS3 to create wifi simple infrastructure nodes.

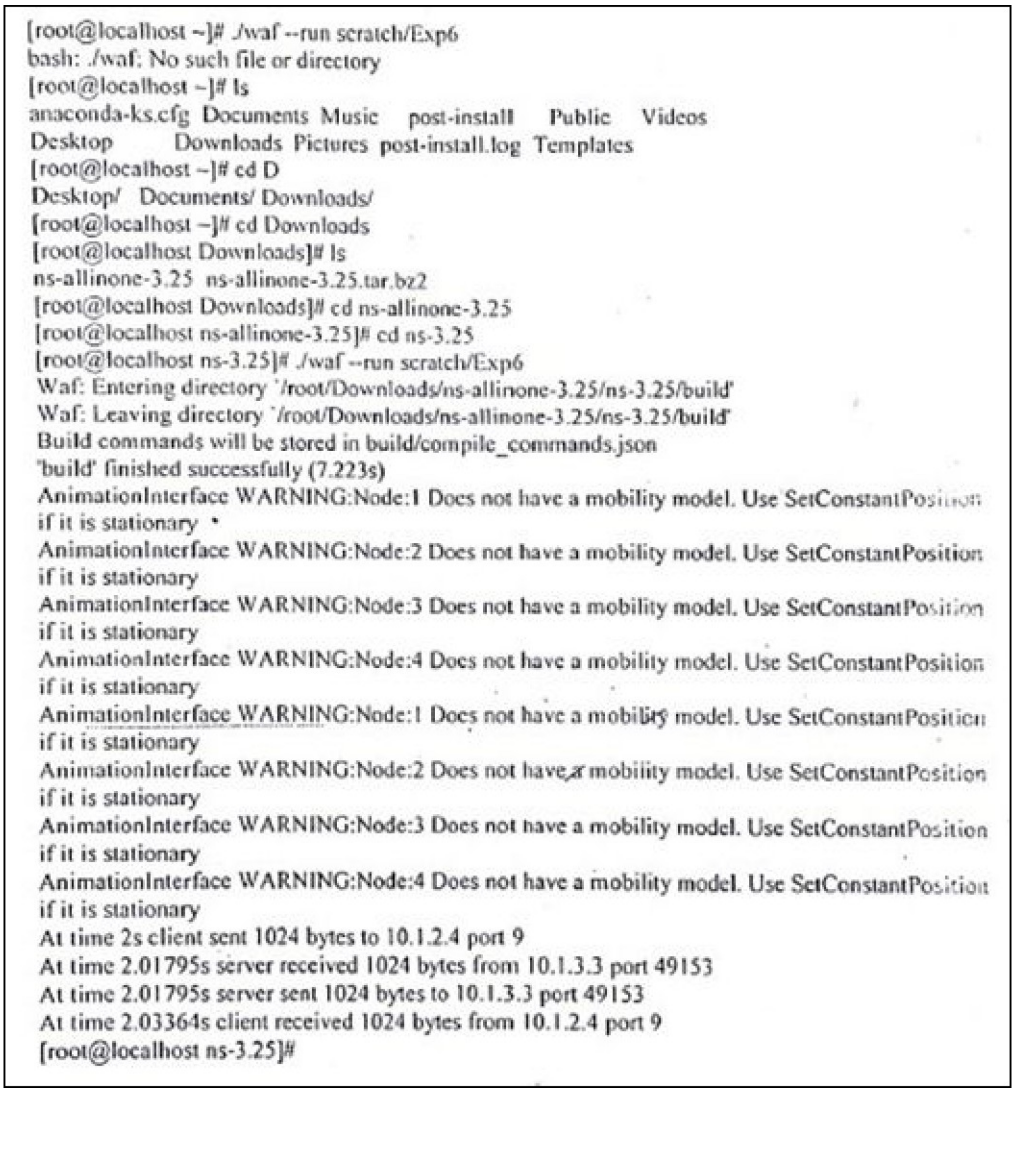
SOURCE CODE :

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include <bits/stdc++.h>
NS_LOG_COMPONENT_DEFINE ("WifiSimpleInfra");
using namespace ns3;

void ReceivePacket (Ptr<Socket> socket) {
        NS_LOG_UNCOND ("Received one packet!");
}

static void GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize, uint32_t pktCount, Time
pktInterval ) {
if (pktCount > 0) {
socket->Send (Create<Packet> (pktSize));
Simulator::Schedule (pktInterval, &GenerateTraffic, socket, pktSize,pktCount-1, pktInterval);
}
else { socket->Close (); }
}

int main (int argc, char *argv[])
{ std::string phyMode
("DsssRate1Mbps"); double rss = -80; //
-dBm
uint32_t packetSize = 1000; // bytes
uint32_t numPackets = 1;
double interval = 1.0; // seconds
bool verbose = false;
CommandLine cmd;
cmd.AddValue ("phyMode", "Wifi Phy mode", phyMode);
cmd.AddValue ("rss", "received signal strength", rss);
cmd.AddValue ("packetSize", "size of application packet sent", packetSize);
cmd.AddValue ("numPackets", "number of packets generated", numPackets);
cmd.AddValue ("interval", "interval (seconds) between packets", interval);
cmd.AddValue ("verbose", "turn on all WifiNetDevice log components", verbose);
cmd.Parse (argc, argv);
Time interPacketInterval = Seconds (interval);
```

```
Config::SetDefault ("ns3::WifiRemoteStationManager::FragmentationThreshold",
StringValue ("2200"));
Config::SetDefault ("ns3::WifiRemoteStationManager::RtsCtsThreshold", StringValue
("2200"));
("ns3::WifiRemoteStationManager::NonUnicastMode", StringValue (phyMode));
NodeContainer c;
c.Create (2);
WifiHelper wifi;
if (verbose) { wifi.EnableLogComponents (); }
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
wifiPhy.Set ("RxGain", DoubleValue (0) );
wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::FixedRssLossModel","Rss",DoubleValue (rss));
wifiPhy.SetChannel (wifiChannel.Create ());
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager","DataMode",StringValue
(phyMode), "ControlMode",StringValue (phyMode));
Ssid ssid = Ssid ("wifi-default");
wifiMac.SetType ("ns3::StaWifiMac", "Ssid", SsidValue (ssid), "ActiveProbing",
BooleanValue (false));
NetDeviceContainer staDevice = wifi.Install (wifiPhy, wifiMac, c.Get (0));
NetDeviceContainer devices = staDevice;
wifiMac.SetType ("ns3::ApWifiMac", "Ssid", SsidValue (ssid));
NetDeviceContainer apDevice = wifi.Install (wifiPhy, wifiMac, c.Get (1));
devices.Add (apDevice);
 MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
positionAlloc->Add (Vector (0.0, 0.0, 0.0));
positionAlloc->Add (Vector (5.0, 0.0, 0.0));
mobility.SetPositionAllocator (positionAlloc);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (c);
InternetStackHelper internet;
internet.Install (c);
Ipv4AddressHelper ipv4;
NS_LOG_INFO ("Assign IP Addresses.");
ipv4.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer i = ipv4.Assign (devices);
TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
Ptr<Socket> recvSink = Socket::CreateSocket (c.Get (0), tid);
```

```
InetSocketAddress local = InetSocketAddress (Ipv4Address::GetAny (), 80);
recvSink->Bind (local);
recvSink->SetRecvCallback (MakeCallback (&ReceivePacket));
Ptr<Socket> source = Socket::CreateSocket (c.Get (1), tid);
InetSocketAddress remote = InetSocketAddress (Ipv4Address ("255.255.255.255"), 80);
source->SetAllowBroadcast (true);
source->Connect (remote);
wifiPhy.EnablePcap ("wifi-simple-infra", devices);
NS_LOG_UNCOND ("Testing " << numPackets << "packets sent with receiver rss" << rss);
Simulator::ScheduleWithContext (source->GetNode ()->GetId (), Seconds (1.0),
&GenerateTraffic, source, packetSize, numPackets, interPacketInterval);
Simulator::Stop (Seconds (30.0)); Simulator::Run (); Simulator::Destroy ();
return 0;
}
```

OUTPUT :



```
[root@localhost ~]# ./waf --run scratch/Exp6
bash: ./waf: No such file or directory
[root@localhost ~]# ls
anaconda-ks.cfg  Documents  Music     post-install      Public     Videos
Desktop          Downloads  Pictures  post-install.log  Templates
[root@localhost ~]# cd D
Desktop/  Documents/ Downloads/
[root@localhost ~]# cd Downloads
[root@localhost Downloads]# ls
ns-allinone-3.25  ns-allinone-3.25.tar.bz2
[root@localhost Downloads]# cd ns-allinone-3.25
[root@localhost ns-allinone-3.25]# cd ns-3.25
[root@localhost ns-3.25]# ./waf --run scratch/Exp6
Waf: Entering directory `/root/Downloads/ns-allinone-3.25/ns-3.25/build'
Waf: Leaving directory `/root/Downloads/ns-allinone-3.25/ns-3.25/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (7.223s)
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition
if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition
if it is stationary
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.01795s server received 1024 bytes from 10.1.3.3 port 49153
At time 2.01795s server sent 1024 bytes to 10.1.3.3 port 49153
At time 2.03364s client received 1024 bytes from 10.1.2.4 port 9
[root@localhost ns-3.25]#
```

EXPERIMENT - 7

AIM : Write an experiment to analyze network traffic using WIRESHARK tool.

THEORY :

Wireshark is a third-party graphical user interface (GUI) network protocol analyzer that is used to interactively dump and analyze network traffic. Similar to the snoop command, you can use Wireshark to browse packet data on a live network or from a previously saved capture file. By default, Wireshark uses the libpcap format for file captures, which is also used by the tcpdump utility and other similar tools. A key advantage of using Wireshark is that it is capable of reading and importing several other file formats besides the libpcap format.

Both TShark and Wireshark provide several unique features, including the following:

- Capable of assembling all of the packets in a TCP conversation and displaying the data in that conversation in ASCU, EBCDIC or hex format
- Contain more filterable fields than in other network protocol analyzers Use a syntax that is richer than other network protocol analyzers for eating filters.

Wireshark is a useful tool to determine the cause of slow network connections. The following steps show you how to configure **Wireshark**:

Install Wireshark:

On Windows, download Wireshark and install with the default selections

On Linux, enter the commands:

yum search wireshark

yum install wireshark.x86_64k

1. yum install wireshark-gnome

   Note: Replace "wireshark.x86_64" with the build that corresponds to your Linux workstation platform.

2. If the Protocol field lists "UNKNOWN", select Analyze->Enabled Protocols->Enable All
3. **Configure the interface to be analyzed:**

   When starting, the "Welcome to Wireshark" screen will show a heading named "Capture". Choose the interface with a graph activity.
4. **Define filters**

   Suppose you want to monitor traffic to port 20182. At the top, click the **Expression** item

next to the **Filter** item
Choose Field Name TCP - Transmission Control Protocol and expand to select tcp.dstport - Destination Port, Relation ==, Value 20182 where 20182 is the Helix server port number you are connecting to. Click OK.

5. Capture Data

Click Capture -> Start from the Menu

Click tcp.destport == 20182 and press the enter key.

Run: p4 info

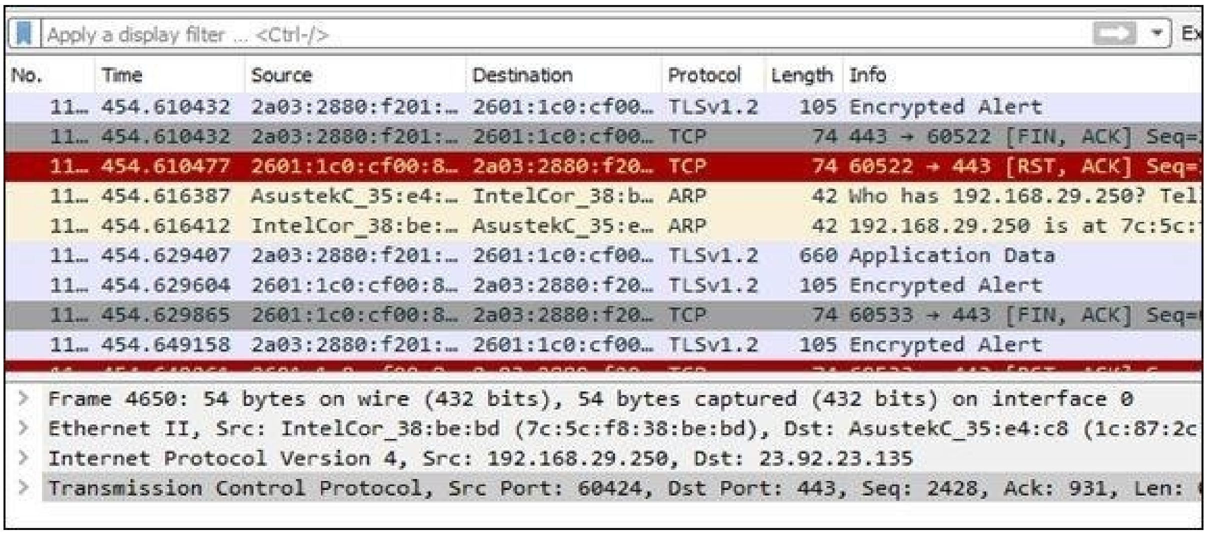
and check that you see an immediate response.

6. Launch the application you want to analyze and reproduce the issue.
7. Select the **Capture | Stop** menu item when you have completed reproducing the issue.
8. Double-click any of the packets to see the packet contents
9. To save the results, select the **File | Save as...** menu item to save the output as a .pcap file for analysis. Search the output as described below.

EXPERIMENT - 8

AIM : Experiment to capture, filter and inspect elements using WIRESHARK tool.

How to Use Wireshark to Capture, Filter and Inspect Packets



```
Apply a display filter ... <Ctrl-/>
No.    Time        Source              Destination         Protocol  Length Info
  11… 454.610432 2a03:2880:f201:…   2601:1c0:cf00…      TLSv1.2     105 Encrypted Alert
  11… 454.610432 2a03:2880:f201:…   2601:1c0:cf00…      TCP          74 443 → 60522 [FIN, ACK] Seq=
  11… 454.610477 2601:1c0:cf00:8…   2a03:2880:f20…      TCP          74 60522 → 443 [RST, ACK] Seq=
  11… 454.616387 AsustekC_35:e4:…   IntelCor_38:b…      ARP          42 Who has 192.168.29.250? Tel
  11… 454.616412 IntelCor_38:be:…   AsustekC_35:e…      ARP          42 192.168.29.250 is at 7c:5c:
  11… 454.629407 2a03:2880:f201:…   2601:1c0:cf00…      TLSv1.2     660 Application Data
  11… 454.629604 2601:1c0:cf00:8…   2a03:2880:f20…      TLSv1.2     105 Encrypted Alert
  11… 454.629865 2601:1c0:cf00:8…   2a03:2880:f20…      TCP          74 60533 → 443 [FIN, ACK] Seq=
  11… 454.649158 2a03:2880:f201:…   2601:1c0:cf00…      TLSv1.2     105 Encrypted Alert

> Frame 4650: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: IntelCor_38:be:bd (7c:5c:f8:38:be:bd), Dst: AsustekC_35:e4:c8 (1c:87:2c
> Internet Protocol Version 4, Src: 192.168.29.250, Dst: 23.92.23.135
> Transmission Control Protocol, Src Port: 60424, Dst Port: 443, Seq: 2428, Ack: 931, Len:
```

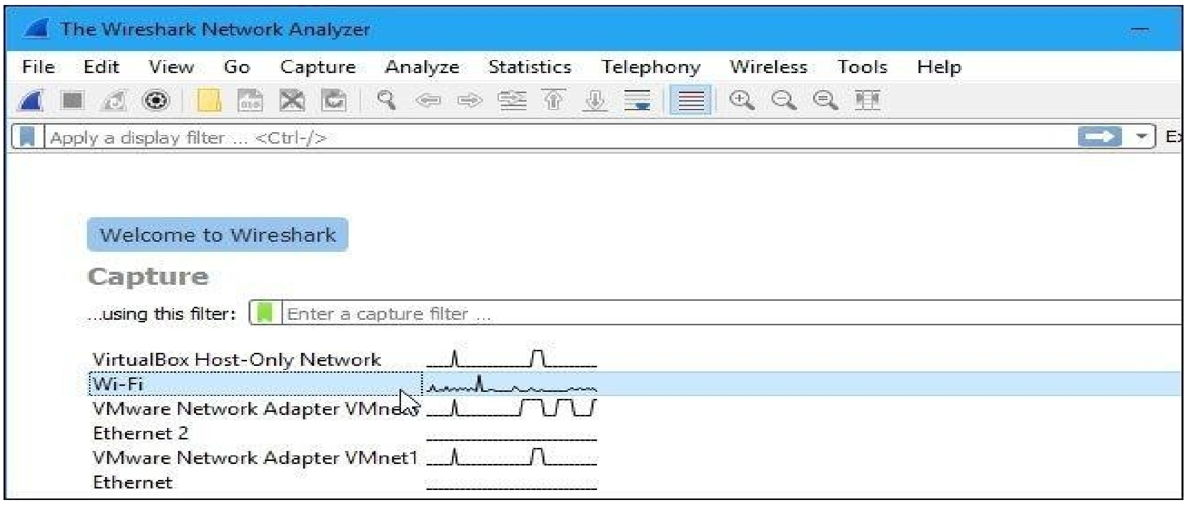
Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and displays them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets.
This tutorial will get you up to speed with the basics of capturing packets, filtering them, and inspecting them. You can use Wireshark to inspect a suspicious program's network traffic, analyze the traffic flow on your network, or troubleshoot network problems.

Getting Wireshark

You can download Wireshark for Windows or macOS from its official website. If you're using Linux or another UNIX-like system, you'll probably find Wireshark in its package repositories. For example, if you're using Ubuntu, you'll find Wireshark in the Ubuntu Software Center. Just a quick warning: Many organizations don't allow Wireshark and similar tools on their networks. Don't use this tool at work unless you have permission.

Capturing Packets

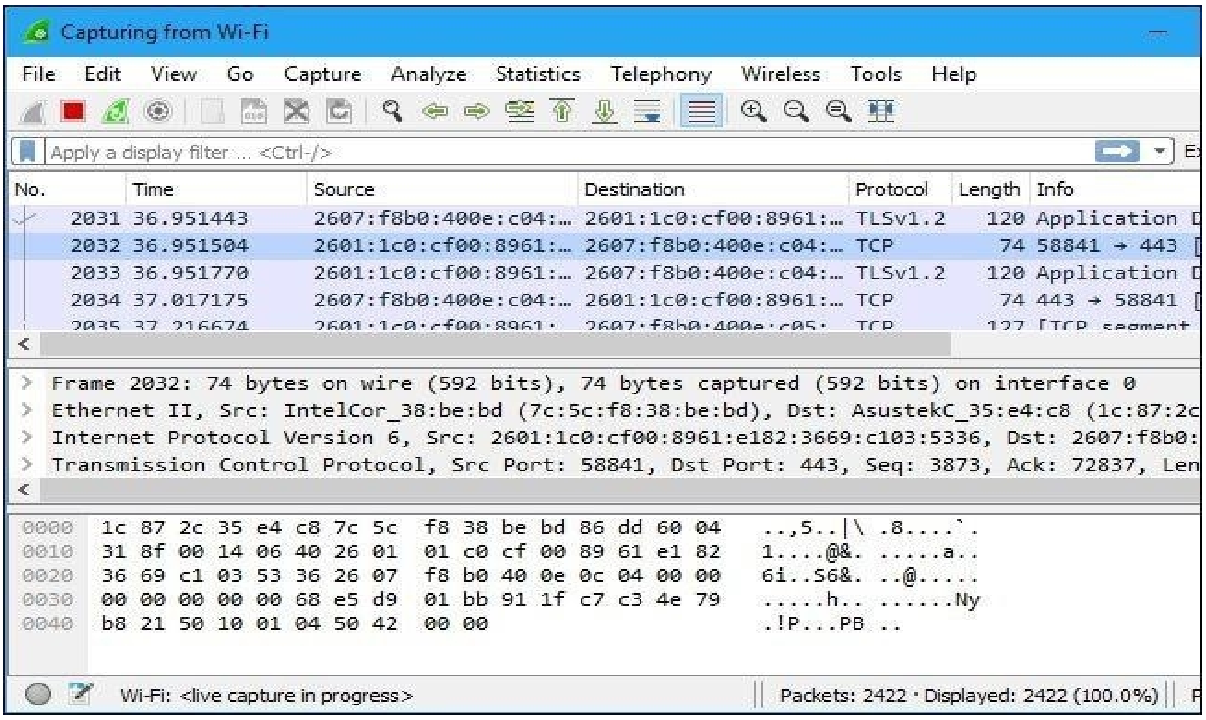
After downloading and installing Wireshark, you can launch it and double-click the name of a network interface under Capture to start capturing packets on that interface. For example, if you want to capture traffic on your wireless network, click your wireless interface. You can configure advanced features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


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...

```
VirtualBox Host-Only Network
Wi-Fi
VMware Network Adapter VMnet
Ethernet 2
VMware Network Adapter VMnet1
Ethernet
```

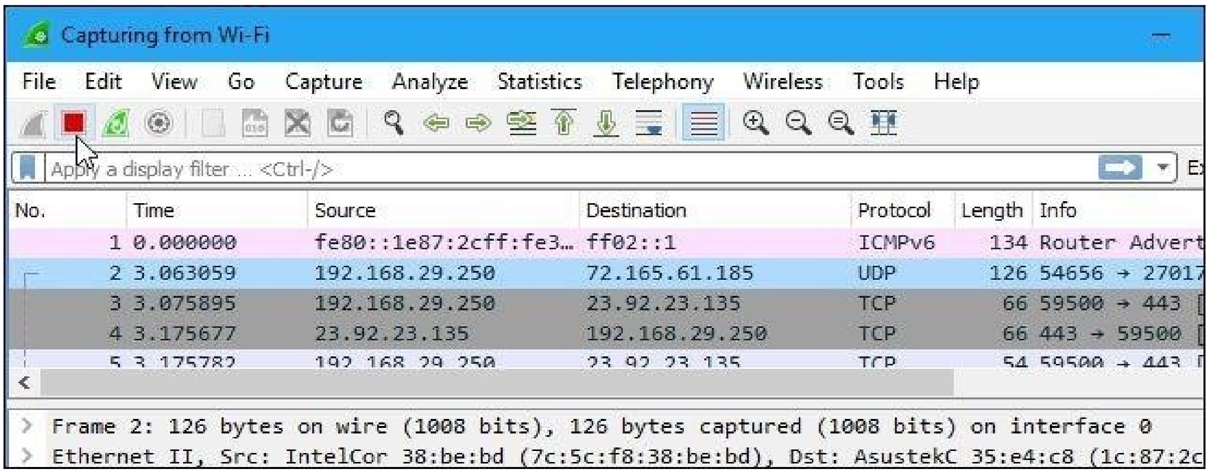
As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.

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.


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time       Source                 Destination            Protocol  Length Info
 2031 36.951443  2607:f8b0:400e:c04:…   2601:1c0:cf00:8961:…   TLSv1.2     120 Application D
 2032 36.951504  2601:1c0:cf00:8961:…   2607:f8b0:400e:c04:…   TCP          74 58841 → 443 [
 2033 36.951770  2601:1c0:cf00:8961:…   2607:f8b0:400e:c04:…   TLSv1.2     120 Application D
 2034 37.017175  2607:f8b0:400e:c04:…   2601:1c0:cf00:8961:…   TCP          74 443 → 58841 [
 2035 37.216674  2601:1c0:cf00:8961:    2607:f8b0:400e:c05:    TCP         127 [TCP segment

> Frame 2032: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
> Ethernet II, Src: IntelCor_38:be:bd (7c:5c:f8:38:be:bd), Dst: AsustekC_35:e4:c8 (1c:87:2c
> Internet Protocol Version 6, Src: 2601:1c0:cf00:8961:e182:3669:c103:5336, Dst: 2607:f8b0:
> Transmission Control Protocol, Src Port: 58841, Dst Port: 443, Seq: 3873, Ack: 72837, Len

0000  1c 87 2c 35 e4 c8 7c 5c  f8 38 be bd 86 dd 60 04   ..,5..|\ .8....`.
0010  31 8f 00 14 06 40 26 01  01 c0 cf 00 89 61 e1 82   1....@&. .....a..
0020  36 69 c1 03 53 36 26 07  f8 b0 40 0e 0c 04 00 00   6i..S6&. ..@.....
0030  00 00 00 00 00 68 e5 d9  01 bb 91 1f c7 c3 4e 79   .....h.. ......Ny
0040  b8 21 50 10 01 04 50 42  00 00                     .!P...PB ..

Wi-Fi: <live capture in progress>        Packets: 2422 · Displayed: 2422 (100.0%)
```

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


Capturing from Wi-Fi

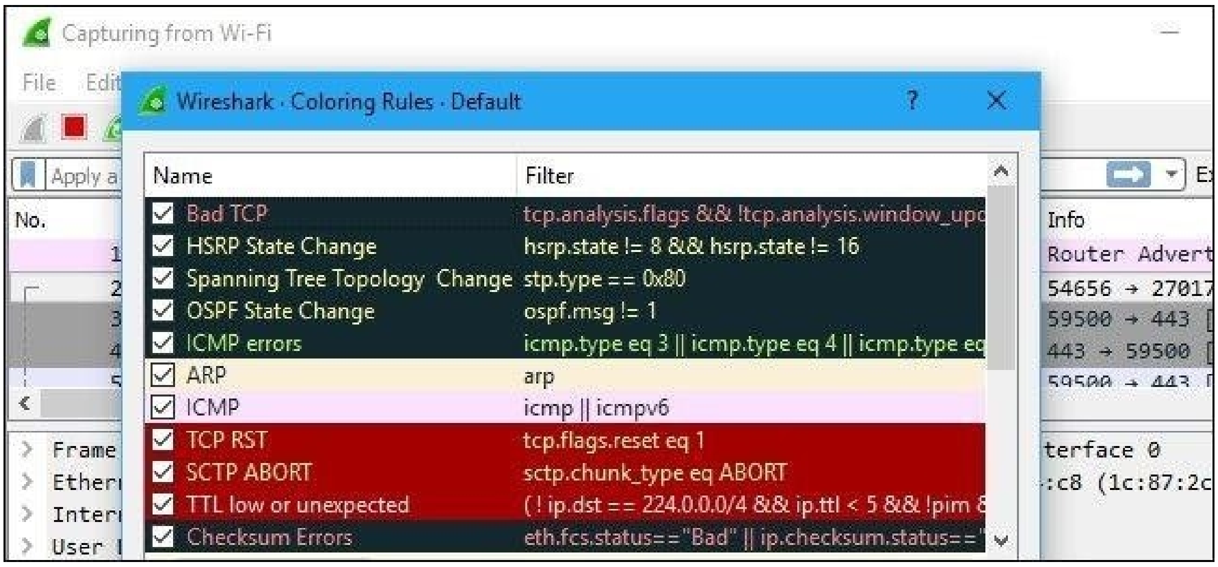
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	fe80::1e87:2cff:fe3…	ff02::1	ICMPv6	134	Router Advert
2	3.063059	192.168.29.250	72.165.61.185	UDP	126	54656 → 27017
3	3.075895	192.168.29.250	23.92.23.135	TCP	66	59500 → 443 [
4	3.175677	23.92.23.135	192.168.29.250	TCP	66	443 → 59500 [
5	3.175782	192.168.29.250	23.92.23.135	TCP	54	59500 → 443 [

```
> Frame 2: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface 0
> Ethernet II, Src: IntelCor_38:be:bd (7c:5c:f8:38:be:bd), Dst: AsustekC_35:e4:c8 (1c:87:2c
```

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 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.

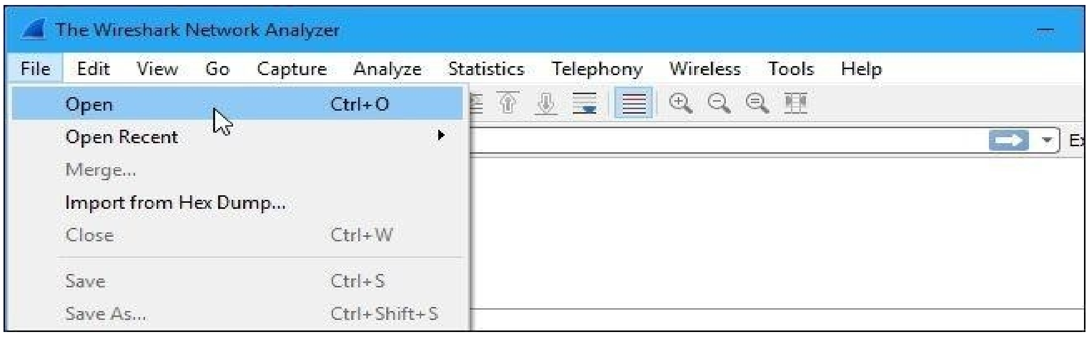

Wireshark · Coloring Rules · Default

Name	Filter
✓ Bad TCP	tcp.analysis.flags && !tcp.analysis.window_upd
✓ HSRP State Change	hsrp.state != 8 && hsrp.state != 16
✓ Spanning Tree Topology Change	stp.type == 0x80
✓ OSPF State Change	ospf.msg != 1
✓ ICMP errors	icmp.type eq 3 || icmp.type eq 4 || icmp.type eq
✓ ARP	arp
✓ ICMP	icmp || icmpv6
✓ TCP RST	tcp.flags.reset eq 1
✓ SCTP ABORT	sctp.chunk_type eq ABORT
✓ TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim &
✓ Checksum Errors	eth.fcs.status=="Bad" || ip.checksum.status=="

Sample Captures

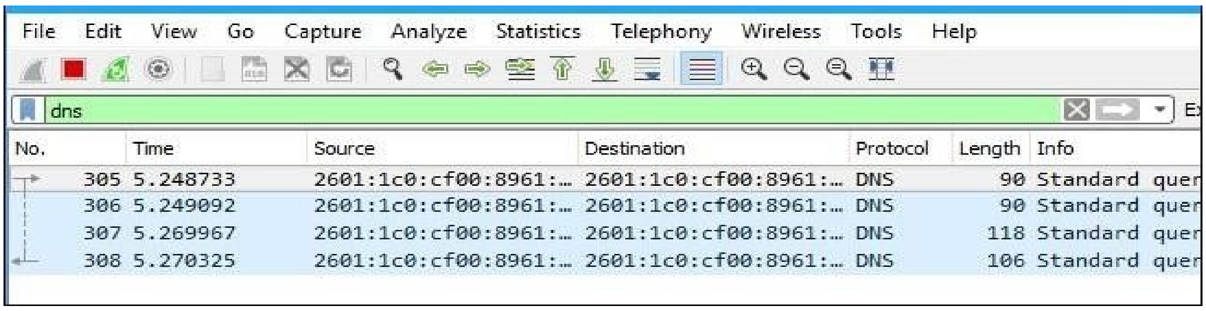
If there's nothing interesting on your own network to inspect, Wireshark's wiki has you covered. The wiki contains a page of sample capture files that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one.

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


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Open Ctrl+O
Open Recent
Merge...
Import from Hex Dump...
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S

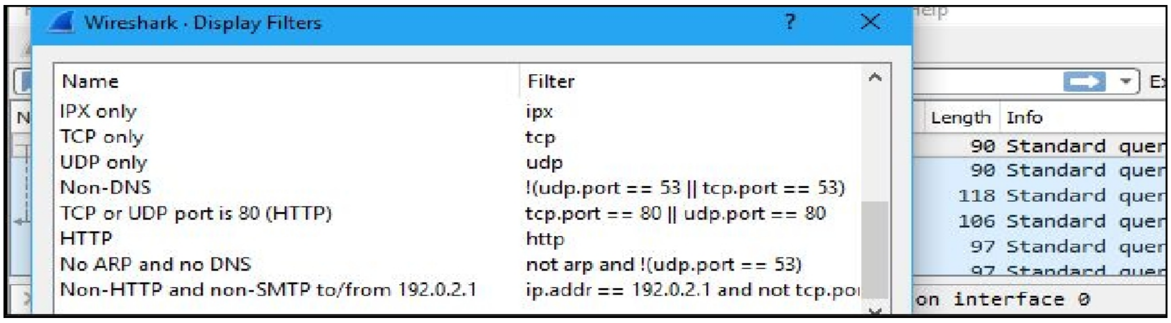
Filtering Packets

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 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.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns

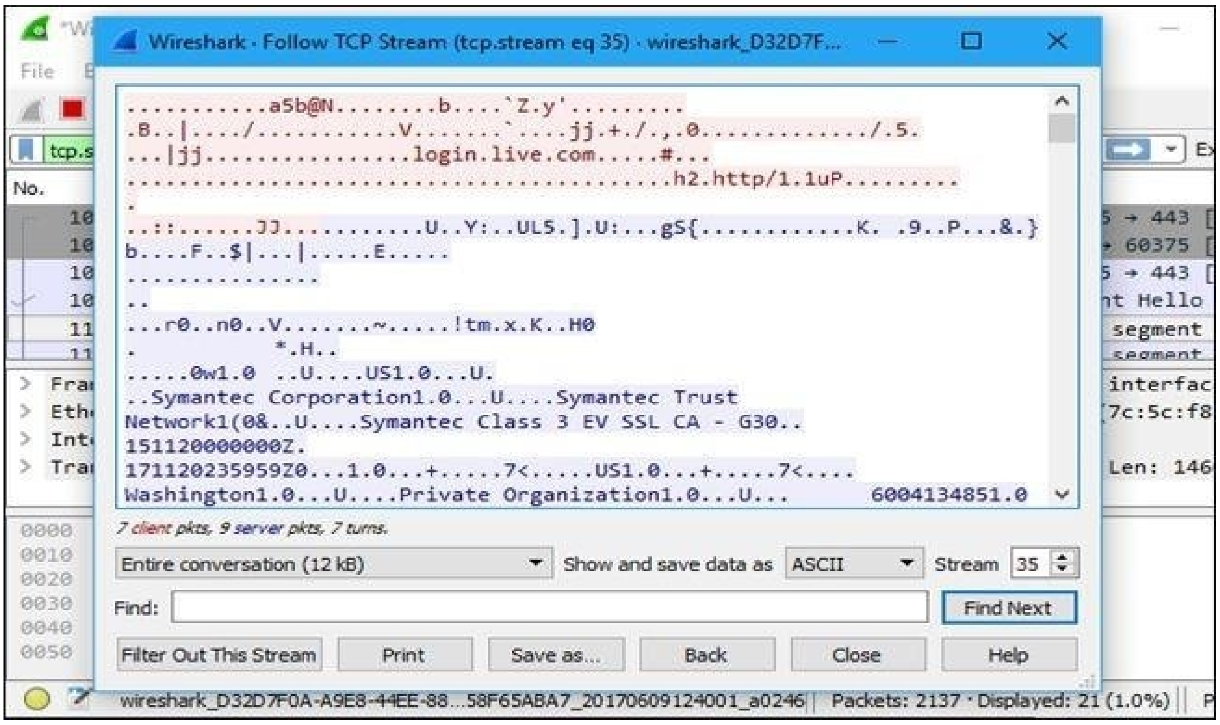
```
No.   Time       Source                 Destination            Protocol  Length Info
  305 5.248733   2601:1c0:cf00:8961:…   2601:1c0:cf00:8961:…   DNS         90 Standard quer
  306 5.249092   2601:1c0:cf00:8961:…   2601:1c0:cf00:8961:…   DNS         90 Standard quer
  307 5.269967   2601:1c0:cf00:8961:…   2601:1c0:cf00:8961:…   DNS        118 Standard quer
  308 5.270325   2601:1c0:cf00:8961:…   2601:1c0:cf00:8961:…   DNS        106 Standard quer
```

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.



Name	Filter
IPX only	ipx
TCP only	tcp
UDP only	udp
Non-DNS	!(udp.port == 53 || tcp.port == 53)
TCP or UDP port is 80 (HTTP)	tcp.port == 80 || udp.port == 80
HTTP	http
No ARP and no DNS	not arp and !(udp.port == 53)
Non-HTTP and non-SMTP to/from 192.0.2.1	ip.addr == 192.0.2.1 and not tcp.po

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



Wireshark · Follow TCP Stream (tcp.stream eq 35) · wireshark_D32D7F...

```
...........a5b@N........b....`Z.y'.........
.B..|..../...........V.......`....jj.+./.,.0.............../.5.
...|jj................login.live.com.....#...
.......................................h2.http/1.1uP.........
.
..::......JJ...........U..Y:..UL5.].U:...gS{............K. .9..P...&.}
b....F..$|...|.....E.....
...............
..
...r0..n0..V.......~.....!tm.x.K..H0
.          *.H..
.....0w1.0  ..U....US1.0...U.
..Symantec Corporation1.0...U....Symantec Trust
Network1(0&..U....Symantec Class 3 EV SSL CA - G30..
151120000000Z.
171120235959Z0...1.0...+.....7<.....US1.0...+.....7<....
Washington1.0...U....Private Organization1.0...U...      6004134851.0
```

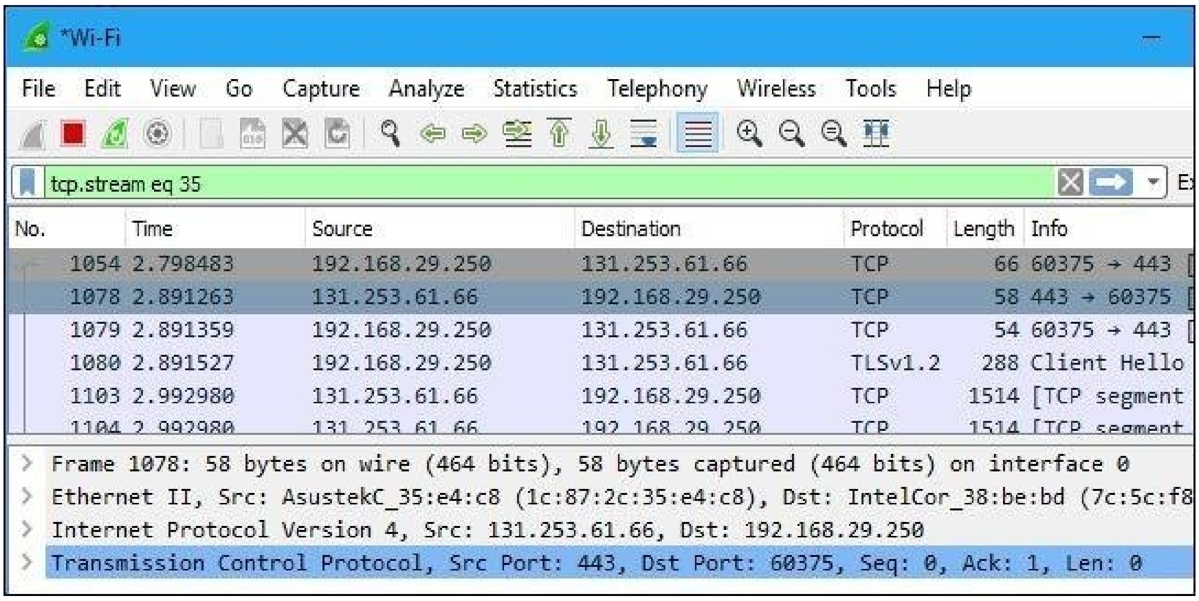
7 client pkts, 9 server pkts, 7 turns.

Entire conversation (12 kB) | Show and save data as: ASCII | Stream: 35

Find:

wireshark_D32D7F0A-A9E8-44EE-88...58F65ABA7_20170609124001_a0246 | Packets: 2137 · Displayed: 21 (1.0%)

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



*Wi-Fi

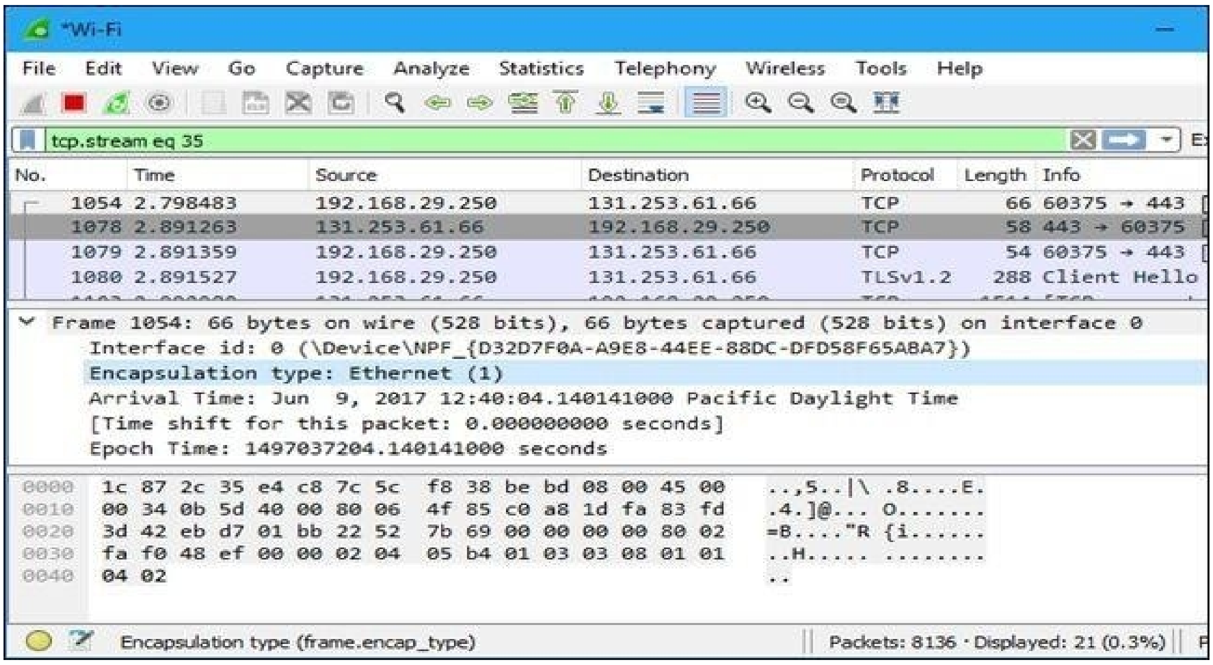
tcp.stream eq 35

```
No.   Time      Source          Destination     Protocol Length Info
 1054 2.798483  192.168.29.250  131.253.61.66   TCP        66 60375 → 443 [
 1078 2.891263  131.253.61.66   192.168.29.250  TCP        58 443 → 60375 [
 1079 2.891359  192.168.29.250  131.253.61.66   TCP        54 60375 → 443 [
 1080 2.891527  192.168.29.250  131.253.61.66   TLSv1.2   288 Client Hello
 1103 2.992980  131.253.61.66   192.168.29.250  TCP      1514 [TCP segment
 1104 2.992980  131.253.61.66   192.168.29.250  TCP      1514 [TCP segment
```

```
> Frame 1078: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface 0
> Ethernet II, Src: AsustekC_35:e4:c8 (1c:87:2c:35:e4:c8), Dst: IntelCor_38:be:bd (7c:5c:f8
> Internet Protocol Version 4, Src: 131.253.61.66, Dst: 192.168.29.250
> Transmission Control Protocol, Src Port: 443, Dst Port: 60375, Seq: 0, Ack: 1, Len: 0
```

Inspecting Packets

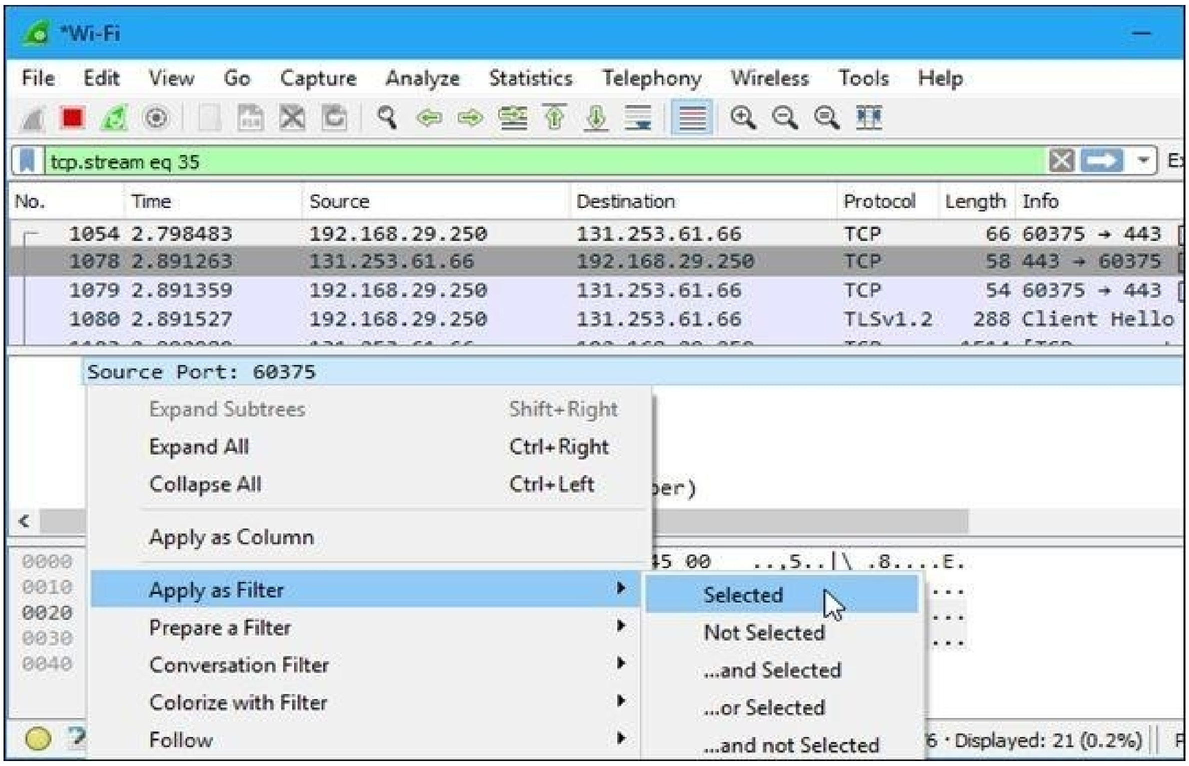
Click a packet to select it and you can dig down to view its details.



*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 35

```
No.     Time       Source          Destination     Protocol  Length Info
  1054 2.798483   192.168.29.250  131.253.61.66   TCP          66 60375 → 443 [
  1078 2.891263   131.253.61.66   192.168.29.250  TCP          58 443 → 60375 [
  1079 2.891359   192.168.29.250  131.253.61.66   TCP          54 60375 → 443 [
  1080 2.891527   192.168.29.250  131.253.61.66   TLSv1.2     288 Client Hello

Frame 1054: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
    Interface id: 0 (\Device\NPF_{D32D7F0A-A9E8-44EE-88DC-DFD58F65A8A7})
    Encapsulation type: Ethernet (1)
    Arrival Time: Jun  9, 2017 12:40:04.140141000 Pacific Daylight Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1497037204.140141000 seconds

0000  1c 87 2c 35 e4 c8 7c 5c  f8 38 be bd 08 00 45 00   ..,5..|\ .8....E.
0010  00 34 0b 5d 40 00 80 06  4f 85 c0 a8 1d fa 83 fd   .4.]@... O.......
0020  3d 42 eb d7 01 bb 22 52  7b 69 00 00 00 00 80 02   =B...."R {i......
0030  fa f0 48 ef 00 00 02 04  05 b4 01 03 03 08 01 01   ..H..... ........
0040  04 02                                              ..

Encapsulation type (frame.encap_type)        Packets: 8136 · Displayed: 21 (0.3%)
```

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.



```
No.     Time       Source          Destination     Protocol  Length Info
  1054 2.798483   192.168.29.250  131.253.61.66   TCP          66 60375 → 443 [
  1078 2.891263   131.253.61.66   192.168.29.250  TCP          58 443 → 60375 [
  1079 2.891359   192.168.29.250  131.253.61.66   TCP          54 60375 → 443 [
  1080 2.891527   192.168.29.250  131.253.61.66   TLSv1.2     288 Client Hello

Source Port: 60375
```

Bhayva Khanna
00696302717