AIM: SIMULATION OF WIRELESS NETWORK IN NS3

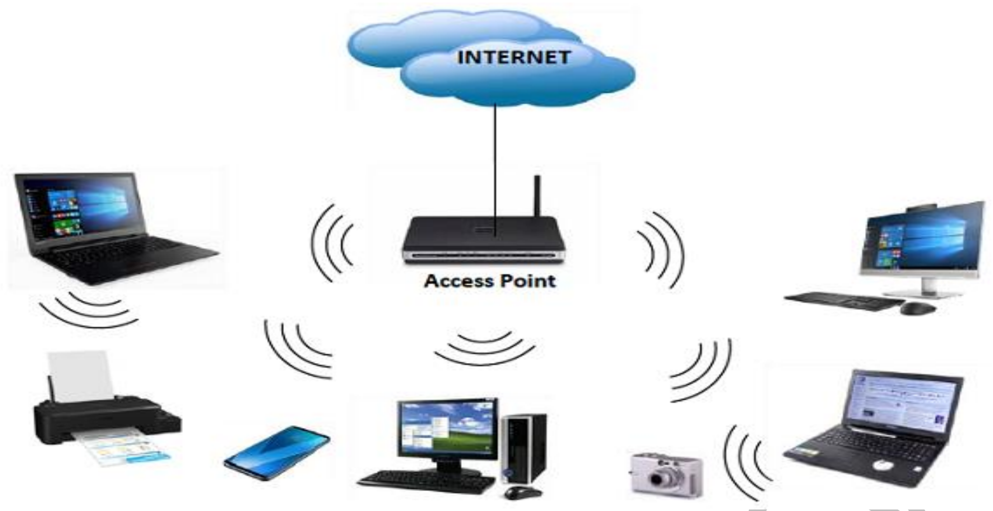
THEORY:

What Is a Wireless Network or Wi-Fi?
A wireless network refers to a computer network that makes use of Radio Frequency (RF) connections between nodes in the network. Wireless networks are a popular solution for homes, businesses, and telecommunications networks.
It is common for people to wonder “what is a wireless network” because while they exist nearly everywhere people live and work, how they work is often a mystery. Similarly, people often assume that all wireless is Wi-Fi, and many would be surprised to discover that the two are not synonymous. Both use RF, but there are many different types of wireless networks across a range of technologies (Bluetooth, ZigBee, LTE, 5G), while Wi-Fi is specific to the wireless protocol defined by the Institute of Electrical and Electronic Engineers (IEEE) in the 802.11 specification and it’s amendments.

Wired vs. Wireless Network: What Is the Difference?
At the most obvious, a wireless network keeps devices connected to a network while still allowing them the freedom to move about, unencumbered by wires. A wired network, on the other hand, makes use of cables that connect devices to the network. These devices are often desktop or laptop computers but can also include scanners and point-of-sale machines.
There are more subtle technology differences that come in to play between wired and wireless. Most modern wired networks are now “full duplex”, meaning that they can be transmitting/receiving packets in both directions simultaneously. In addition, most wired networks have a dedicated cable that runs to each end user device.
In a Wi-Fi network, the medium (the radio frequency being used for the network) is a shared resource, not just for the users of the network, but often for other technologies as well (Wi-Fi operates in what are called ‘shared’ bands, where many different electronic devices are approved to operate). This has several implications: 1) unlike a wired network, wireless can’t both talk and listen at the same time, it is “half duplex” 2) All users are sharing the same space must take turns to talk 3) everyone can ‘hear’ all traffic going on. This has forced Wi-Fi networks to implement various security measures over the years to protect the confidentiality of information passed wirelessly.

Examples of wireless networks:

- Mobile phone networks
- Wireless sensor networks
- Satellite communication networks
- Terrestrial microwave networks



Types of Wireless Connections
In addition to a LAN, there are a few other types of common wireless networks: personal-area network (PAN), metropolitan-area network (MAN), and wide-area network (WAN).

LAN
A local-area network is a computer network that exists at a single site, such as an office building. It can be used to connect a variety of components, such as computers, printers, and data storage devices. LANs consist of components like switches, access points, routers, firewalls, and Ethernet cables to tie it all together. Wi-Fi is the most commonly known wireless LAN.

PAN
A personal-area network consists of a network centralized around the devices of a single person in a single location. A PAN could have computers, phones, video game consoles, or other peripheral devices. They are common inside homes and small office buildings. Bluetooth is the most commonly known wireless PAN.

MAN
A metropolitan-area network is a computer network that spans across a city, small geographical area, or business or college campus. One feature that differentiates a MAN from a LAN is its size. A LAN usually consists of a solitary building or area. A MAN can cover several square miles, depending on the needs of the organization.
Large companies, for example, may use a MAN if they have a spacious campus and need to manage key components, such as HVAC and electrical systems.

WAN
A wide-area network covers a very large area, like an entire city, state, or country. In fact, the internet is a WAN. Like the internet, a WAN can contain smaller networks, including LANs or MANs. Cellular services are the most commonly known wireless WANs.

The Components of a Wireless Network

Several components make up a wireless network's topology:

1) **Clients:** What we tend to think of as the end user devices are typically called 'clients'. As the reach of Wi-Fi has expanded, a variety of devices may be using Wi-Fi to connect the network, including phones, tablets, laptops, desktops, and more. This gives users the ability to move about the area without sacrificing their bridge to the network. In some instances, mobility within an office, warehouse, or other work area is necessary. For example, if employees have to use scanners to register packages due to be shipped, a wireless network provides the flexibility they need to freely move about the warehouse.
2) **Access Point (AP):** An access point (AP) consists of a Wi-Fi that is advertising a network name (known as a Service Set Identifier, or SSID). Users who connect to this network will typically find their traffic bridged to a local-area network (LAN) wired network (like Ethernet) for communication to the larger network or even the internet.

SOURCE CODE:

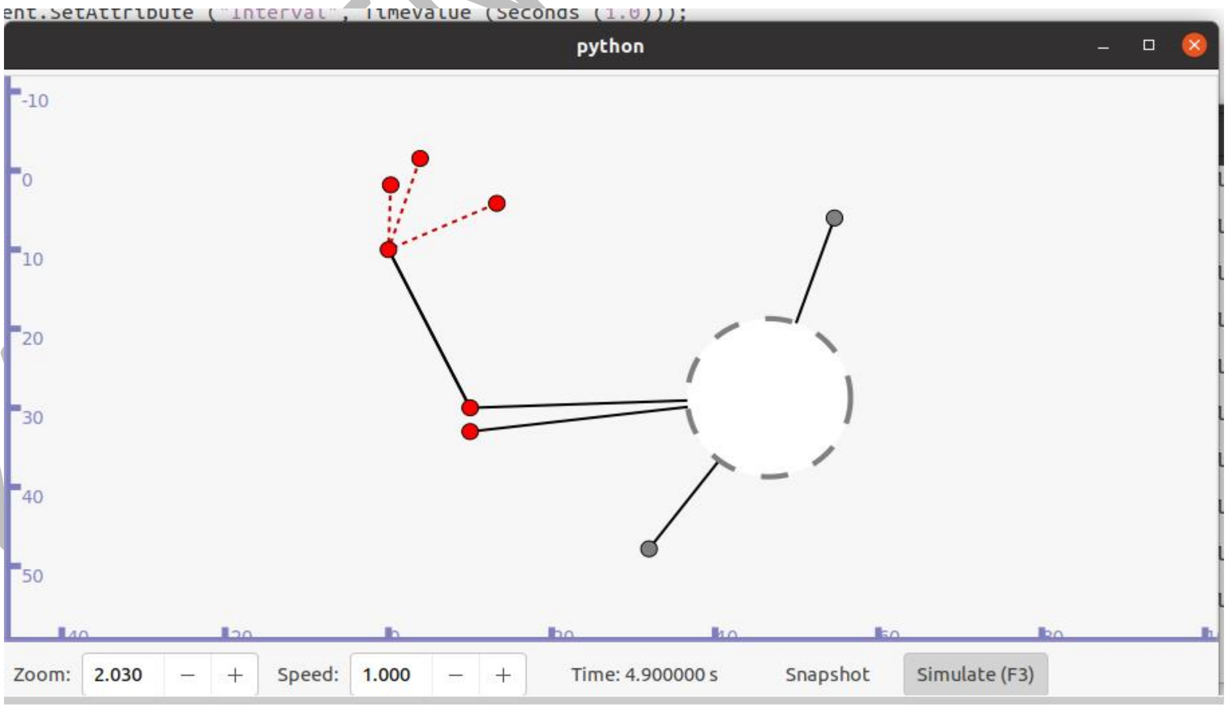
```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
#include "ns3/netanim-module.h"

// Default Network Topology
//
//   Wifi 10.1.3.0
//              AP
//  *    *    *    *
//  |    |    |    |   10.1.1.0
// n5   n6   n7   n0 -------------- n1   n2   n3   n4
//                point-to-point  |    |    |    |
//                                ================
//                                  LAN 10.1.2.0

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 (__FILE__);
  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);

  // The underlying restriction of 18 is due to the grid position
  // allocator's configuration; the grid layout will exceed the
  // bounding box if more than 18 nodes are provided.
  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;
```

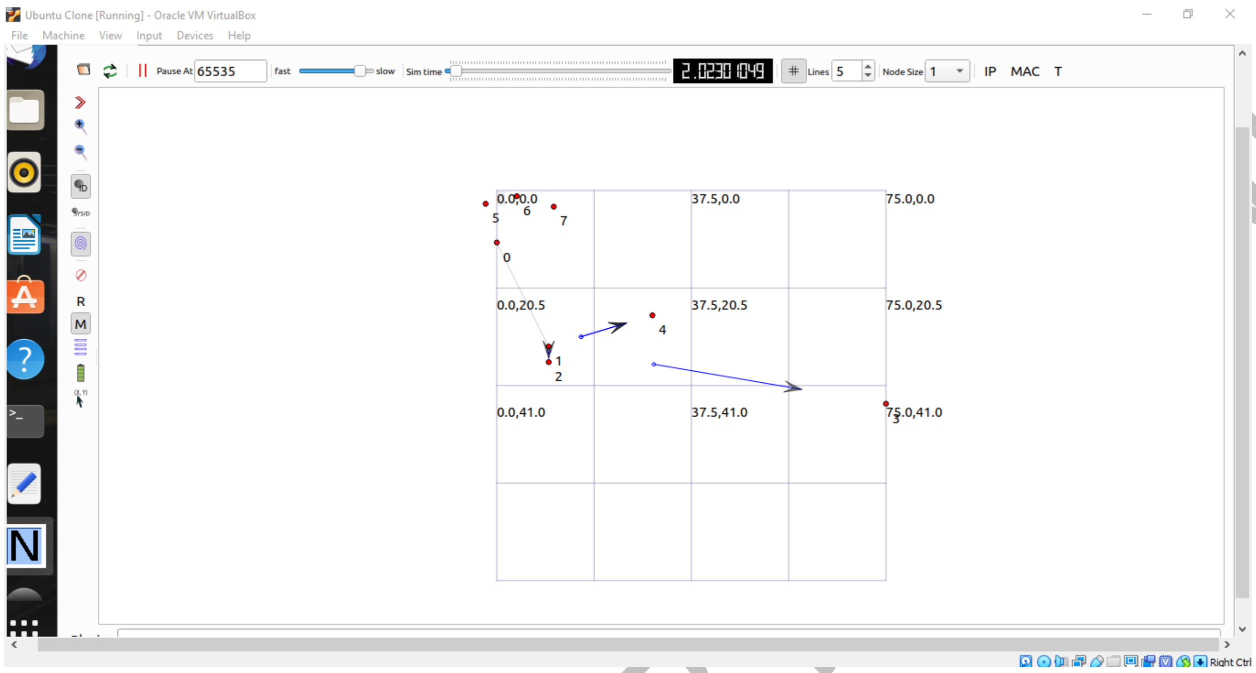
```
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsma);

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);

AnimationInterface::SetConstantPosition (p2pNodes.Get (1), 10, 30);
AnimationInterface::SetConstantPosition (csmaNodes.Get (1), 10, 33);

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);
    }
  AnimationInterface anim ("narender-wireless-animation.xml");

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

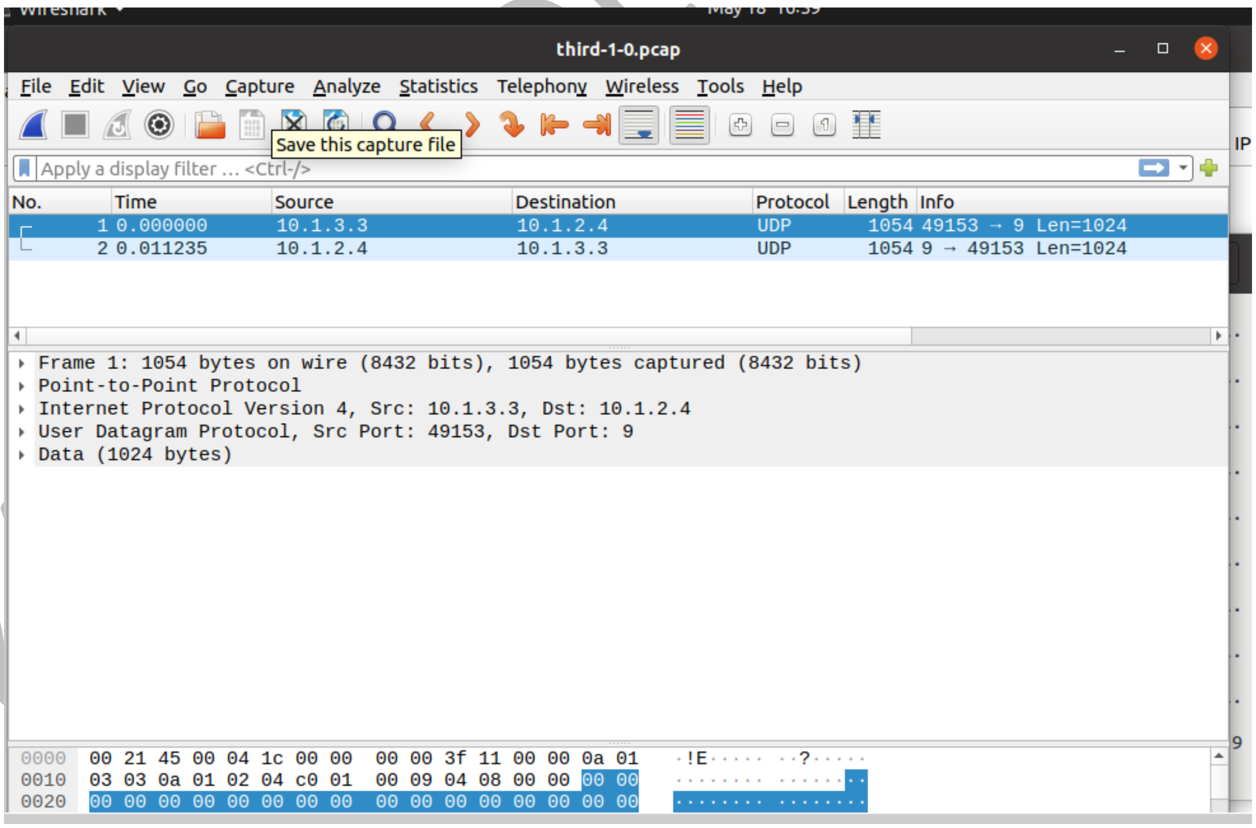
OUTPUT:



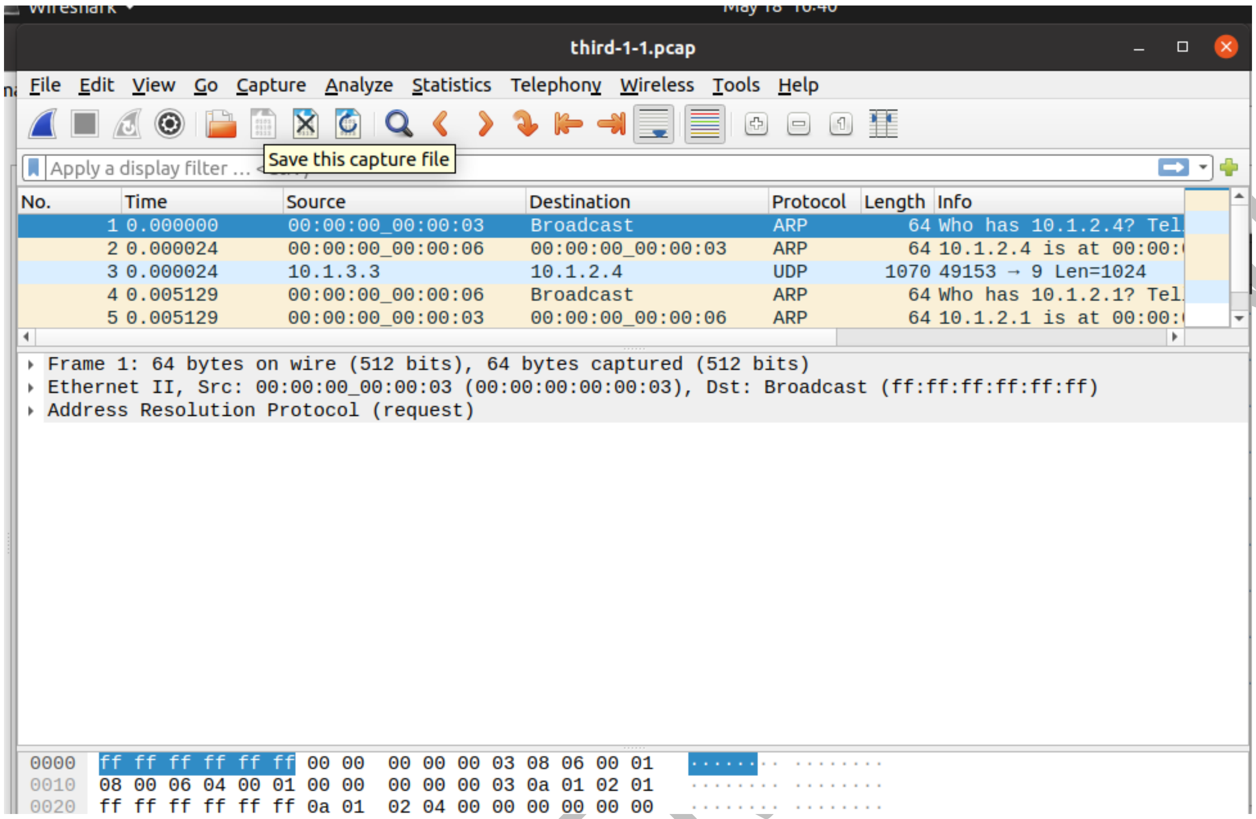
SIMULATION:



WIRESHARK:


third-1-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Save this capture file
Apply a display filter … <Ctrl-/>

```
No.  Time      Source    Destination Protocol Length Info
   1 0.000000  10.1.3.3  10.1.2.4    UDP        1054 49153 → 9  Len=1024
   2 0.011235  10.1.2.4  10.1.3.3    UDP        1054 9 → 49153  Len=1024

▸ Frame 1: 1054 bytes on wire (8432 bits), 1054 bytes captured (8432 bits)
▸ Point-to-Point Protocol
▸ Internet Protocol Version 4, Src: 10.1.3.3, Dst: 10.1.2.4
▸ User Datagram Protocol, Src Port: 49153, Dst Port: 9
▸ Data (1024 bytes)

0000  00 21 45 00 04 1c 00 00  00 00 3f 11 00 00 0a 01   ·!E·····  ··?·····
0010  03 03 0a 01 02 04 c0 01  00 09 04 08 00 00 00 00   ········  ········
0020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········  ········
```


third-1-1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter …
Save this capture file

```
No.  Time      Source               Destination          Protocol Length Info
   1 0.000000  00:00:00_00:00:03    Broadcast            ARP        64 Who has 10.1.2.4? Tel
   2 0.000024  00:00:00_00:00:06    00:00:00_00:00:03    ARP        64 10.1.2.4 is at 00:00:
   3 0.000024  10.1.3.3             10.1.2.4             UDP      1070 49153 → 9 Len=1024
   4 0.005129  00:00:00_00:00:06    Broadcast            ARP        64 Who has 10.1.2.1? Tel
   5 0.005129  00:00:00_00:00:03    00:00:00_00:00:06    ARP        64 10.1.2.1 is at 00:00:

 Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
 Ethernet II, Src: 00:00:00_00:00:03 (00:00:00:00:00:03), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 Address Resolution Protocol (request)

0000  ff ff ff ff ff ff 00 00  00 00 00 03 08 06 00 01   ........ ........
0010  08 00 06 04 00 01 00 00  00 00 00 03 0a 01 02 01   ........ ........
0020  ff ff ff ff ff ff 0a 01  02 04 00 00 00 00 00 00   ........ ........
```

CONCLUSION:

From this practical, I have learned how to simulate the wireless networks