PROJECT REPORT

USRP DEVICES

Section:1

Network Scanner

OVERVIEW:

WiFi Scanning or Network scanning refers to the scanning of the whole network to which we are connected and try to find out what are all the clients connected to our network. We can identify each client using their IP and MAC address. We can use ARP ping to find out the alive systems in our network.

The network scanner will send the ARP request indicating who has some specific IP address, let's say "192.158.1.1", the owner of that IP address (the target) will automatically respond saying that he is "192.158.1.1", with that response, the MAC address will also be included in the packet, this allows us to successfully retrieve all network users' IP and MAC addresses simultaneously when we send a broadcast packet (sending a packet to all the devices in the network).

In our code, we are using the ***scapy*** library-

It is a powerful Python-based interactive packet manipulation program and library.

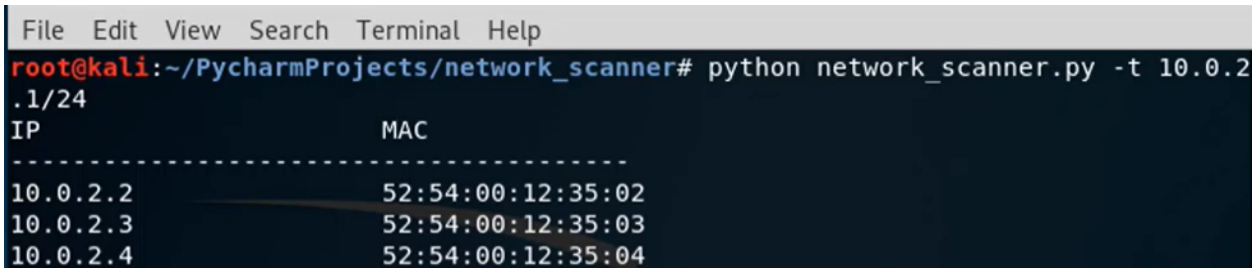
It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, store or read them using pcap files, match requests and replies, and much more. It is designed to allow fast packet prototyping by using default values that work.

Wifi scanner code

```
#!/usr/bin/env python
import scapy.all as scapy
import argparse

def get_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument("-t", "--target", dest="target", help="Specify
target ip or ip range")
    options = parser.parse_args()
    return  options

def scan(ip):
    arp_packet = scapy.ARP(pdst=ip)
    broadcast_packet = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_broadcast_packet = broadcast_packet/arp_packet
    answered_list = scapy.srp(arp_broadcast_packet, timeout=1,
verbose=False)[0]
    client_list = []

    for element in answered_list:
        client_dict = {"ip": element[1].psrc, "mac": element[1].hwsrc}
        client_list.append(client_dict)

    return client_list

def print_result(scan_list):
    print("IP\t\t\tMAC\n-----------------------------------------")
    for client in scan_list:
        print(client["ip"] + "\t\t" + client["mac"])

options = get_arguments()
result_list = scan(options.target)
print_result(result_list)
```

Steps to Run this code in linux:

- Download and install Scapy library in linux
- Open terminal and navigate to the folder having this python script
- Type and file name followed by the range of ip addresses you want to detect
  Eg. network_scanner.py -t // range of ip address
  network_scanner.py -t 10.0.2.4



Section: 2

Application of SDR

A software-defined radio (SDR) is a radio communication system where the major part of its functionality is implemented by means of software in a personal computer or embedded system. Such a design paradigm has the major advantage of producing devices that can receive and transmit widely different radio protocols based solely on the software used. This flexibility opens several application opportunities and one such application is discussed below.

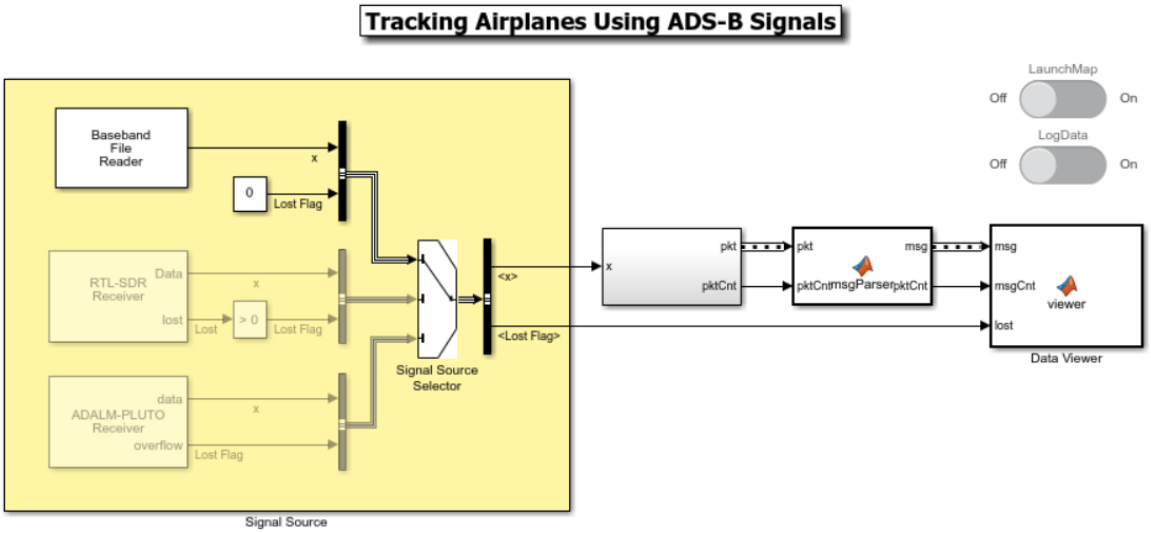
Airplane tracking using ADS-B signals

So basically we will track planes by processing Automatic Dependent Surveillance-Broadcast (ADS-B) signals using MATLAB® and Communications Toolbox™

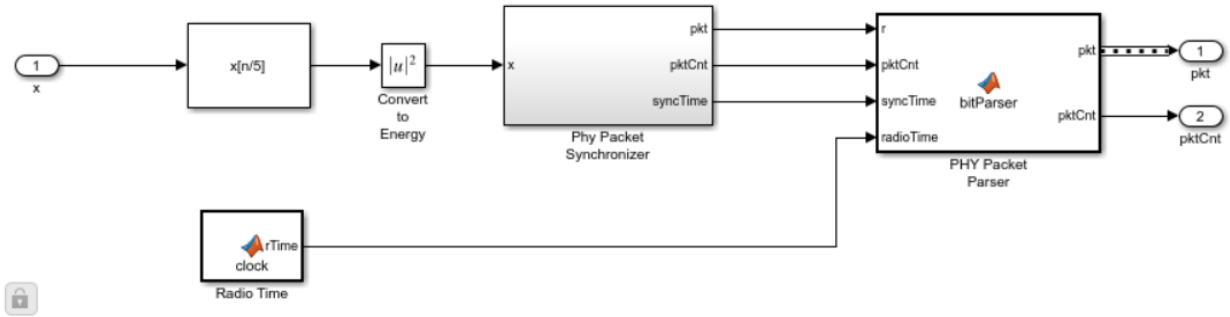
ADS-B is a major component of the FAA's(and other countries) NextGen Airspace overhaul program to make flying safer ,more secure and less of a hassle from an operators standpoint.

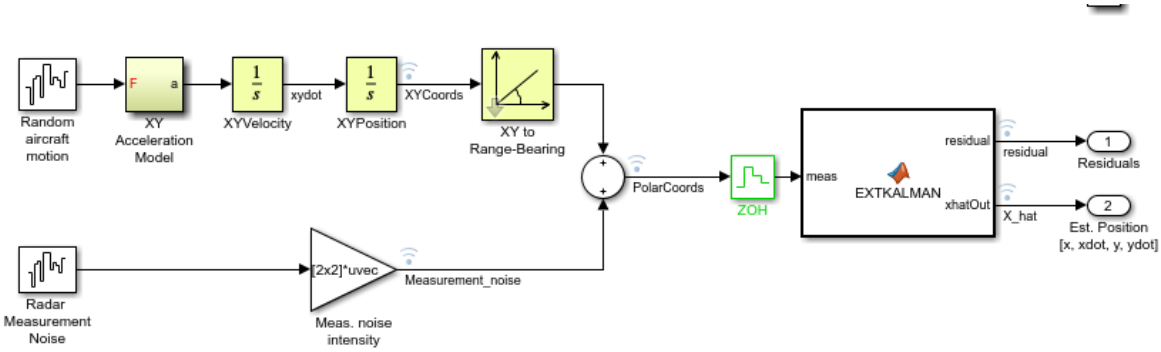
- ❏ It provides air to air and air to ground surveillance capability.
- ❏ Provides surveillance to remote or inhospitable areas that do not currently have coverage with radar
- ❏ Reduces the cost of infrastructure needed to operate the national airspace system.
- ❏ Allows for reduced separation and greater predictability in departure and arrival time thus,reducing airline delays.
- ❏ ADS-B will help our environment by allowing improved routes and will save fuel consumption as well.

Simulink Models:



Physical layer:





Matlab code for processing a given input ADS-B signal file:

```
%For the option to change default settings, set |cmdlineInput| to 1.
cmdlineInput = 0;
if cmdlineInput
    % Request user input from the command-line for application parameters
    userInput = helperAdsbUserInput;
else
    load('defaultinputsADSB.mat');
end

% Calculate ADS-B system parameters based on the user input
[adsbParam,sigSrc] = helperAdsbConfig(userInput);

% Create the data viewer object and configure based on user input
viewer = helperAdsbViewer('LogFileName',userInput.LogFilename, ...
    'SignalSourceType',userInput.SignalSourceType);
if userInput.LogData
  startDataLog(viewer);
end
if userInput.LaunchMap
  startMapUpdate(viewer);
end

% Create message parser object
msgParser = helperAdsbRxMsgParser(adsbParam);

% Start the viewer and initialize radio time
start(viewer)
radioTime = 0;
```

```
% Main loop
while radioTime < userInput.Duration


     if adsbParam.isSourceRadio
        if adsbParam.isSourcePlutoSDR
            [rcv,~,lostFlag] = sigSrc();
        else
            [rcv,~,lost] = sigSrc();
            lostFlag = logical(lost);
        end
    else
        rcv = sigSrc();
        lostFlag = false;
     end

  % Process physical layer information (Physical Layer)
  [pkt,pktCnt] = helperAdsbRxPhy(rcv,radioTime,adsbParam);

  % Parse message bits (Message Parser)
  [msg,msgCnt] = msgParser(pkt,pktCnt);

  % View results packet contents (Data Viewer)
  update(viewer,msg,msgCnt,lostFlag);

  % Update radio time
  radioTime = radioTime + adsbParam.FrameDuration;
end

% Stop the viewer and release the signal source
stop(viewer)
release(sigSrc)
```

PROJECT MEMBERS:
VIVEK SHARMA- 19116068
RAVINA BISHNOI- 19115098

MENTORS:
PRIYANSHU SINGH
LAKSHAY MADAAN