Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

School of Computer Engineering & Technology
Third Year B. Tech Computer Sc. & Engineering
Semester - V

WIRELESS AND MOBILE DEVICE SECURITY

LABORATORY MANUAL

COMPILED BY
SHIV SUTAR (COURSE COORDINATOR)
SCHOOL OF COMPUTER ENGINEERING & TECHNOLOGY, MIT-WPU PUNE - 411038

COURSE STRUCTURE

Course Code	**CET4004B**			
Course Category	**Professional Core Elective**			
Course Title	**Wireless and Mobile Device Security**			
Teaching Scheme and Credits	**L**	**T**	**Laboratory**	**Credits**
Weekly load hrs	**3**	**-**	**2**	**3 + 1**

Pre-requisites:

Basics of Computer Networks
Basics of Information Security

Course Objectives:

1. To understand wireless networks technologies and applications
2. To know wireless and ad-hoc network architecture, issues and challenges
3. To study about various network security attacks and key management in wireless networks

Course Outcomes: After completion of this course students will be able to

1. Demonstrate and interpret the underlying concept of existing wireless networks
2. Illustrate the knowledge of MAC protocols, routing protocols in ad-hoc and sensor networks
3. Examine attacks, security issues and key management issues in wireless network

Wireless and Mobile Security - Laboratory Assignments			
Lab No.	**Contents (Any Ten Laboratories)**	**Workload in Hrs**	
		Theory	**Lab**
1	Install and Configure Network Simulator tool such as Network Simulator 2 or NetSim or QualNet and study its components and eco system.	-	02
2	Write a program to simulate two node wireless network. You may use NetSim or NS2 or QualNet for this experiment. / To create a wireless network scenario and study the performance of CSMA/CA protocol through simulation tools like NetSim or NS2 or QualNet.	-	02
3	Write a program to simulate routing in mobile Ad-Hoc network with multiple nodes. You may use NetSim or NS2 or QualNet for this experiment.	-	02
4	Study the security permissions for applications in android phones. Either demonstrate Android security permission configurations or Write the android app to demonstrate permissions usage control in android phones.	-	02
5	Write an android program to encrypt and decrypt text file. Use Bouncy castle library API or Java cryptography API.	-	02
6	Write a program for user authentication application in Java or Python. Send OTP (one time passwords) to your mobile phones from this application and validate that OTP. It should tell of OTP is correct or wrong. Also add timing restriction in the application.	-	02
7	Configure access point and manage the access control for security. Access point is a networking hardware device that allows a Wi-Fi device to connect to a wired network.	-	02
8	Study, comparison and configuration of different types of Access points routers such CISCO, TP Link, DLink, Link Sys, NetGear. Study Technical specification of such a Wi-Fi routers.	-	02
9	Install, Configure and Demonstrate any one Wi-Fi traffic analyzer using sniffing tools such as WireShark, airCrack, AirSnort, etc.	-	02
10	Consider Android and iPhone device. Analyse, experiment all aspects of device security in these mobile devices. Compare and contrast pros and cons.	-	02
11	Write an Android Application to create secured mobile wallet with cryptographic algorithms	-	02

12	Home Automation (Monitoring and Control) with ZigBee Devices: Scenario Description: ZigBee allows small, low-cost devices to quickly transmit small amounts of data such as temperature readings for thermostats, on/off requests for light switches, or keystrokes for a wireless keyboard. The scenario shows an application of ZigBee technology for Home Automation. It demonstrates the monitoring and control capability that can be achieved with ZigBee.	-	02

Assignment No. 1

Aim: Install and configure network simulator tool such as Network Simulator 2 or NetSim or QualNet and study its functionality. Simulate basic wireless network scenario.

Theory:.

The Network Simulator (NS) has been distributed as free and open source software. The open source nature of the NS simulator has allowed thousands of students, engineers and researchers to contribute scripts and patches. The development of NS-2 has been funded by the DARPA VINT (Virtual Inter Network Testbed) project from 1997-2000, by DARPA SAMAN (Simulation Augmented by Measurement and Analysis for Networks) and NSF CONSER (Collaborative Simulation for Education and Research) from 2000-2004. The simulator NS-2 has had a smooth transition from the NS-1 version, which had a similar architecture; NS-2 was designed to be backward compatible with scripts written in NS-1. In contrast, the gap between the architectures of NS-2 and NS-3 is very large and NS-3 is not backward compatible. This suggests that NS-2 will remain for many years a useful tool, with an advantage of having huge amount of accessible open source software that had been developed during the last decade and not yet ported to NS-3. The open source nature of NS-2 and the community-based development practices of NS-2 which were one of the main sources for its rapid development are expected to continue with theNS-3 version.

The following figure shows the basic architecture of NS2. NS2 provides users with executable command ns which take on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns. In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle (e.g., n as a Node handle) is just a string (e.g.,_o10) in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may define its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively. Before proceeding further, the readers are encouraged to learn C++ and OTcl languages. NS2 provides a large number of built- in C++ objects. It is advisable to use these C++ objects to set up a simulation using a Tcl simulation script. However, advance users may find these objects insufficient. They need to develop their own C++ objects,
and use aOTcl configuration interface to put together these objects. After simulation, NS2 outputs either

text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behavior of the network, users can extract a relevant subset of text-based data and transform it to a more conceivable presentation.

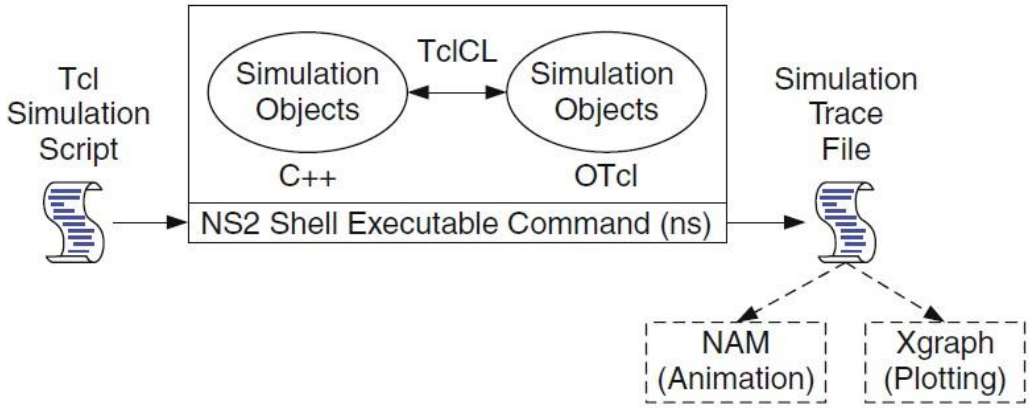


Fig: Basic Architecture of NS

Features of NS2:

- It is a discrete event simulator for networking research.
- It provides substantial support to simulate bunch of protocols like TCP, FTP, UDP, https and DSR.
- It simulates wired and wireless network.
- It is primarily Unix based.
- Uses TCL as its scripting language.
- Otcl: Object oriented support
- Tclcl: C++ and otcl linkage
- Discrete event scheduler

Tcl scripting:

Tcl is a general purpose scripting language. [Interpreter]

• Tcl runs on most of the platforms such as Unix, Windows, and Mac.

• The strength of Tcl is its simplicity.

• It is not necessary to declare a data type for variable prior to the usage.

Basics of TCL

Syntax: command arg1 arg2 arg3

Hello World!

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

Variables Command Substitution

set a 5 set len [string length foobar]
set b $a set len [expr [string length foobar] + 9]

Wired TCL Script Components:

Create the event scheduler
Open new files & turn on the tracing
Create the nodes Setup the links
Configure the traffic type (e.g., TCP, UDP, etc)
Set the time of traffic generation (e.g., CBR, FTP)
Terminate the simulation

NS Simulator Preliminaries:

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

Initialization and Termination of TCL Script in NS-2:

An ns simulation starts with the command: **set ns [new Simulator]**
Which is thus the first line in the tcl script. This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.
In order to have output files with data on the simulation (trace files) or files used for visualization (nam files), we need to create the files using - open command:

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

The above creates a dta trace file called out.tr and a nam visualization trace file called out.nam. Within the tcl script, these files are not called explicitly by their names, but instead by pointers that are declared above and called - tracefile1 and - namfile respectively. Remark that they begins with a # symbol. The second line open the file - out.tr to be used for writing, declared with the letter - w. The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go.

Define a "finish" procedure

```
Proc finish { } {
global ns tracefile1 namfile
$ns flush-trace
Close $tracefile1
Close $namfile
Exec nam out.nam &
Exit 0
}
```

Definition of a network of links and nodes

The way to define a node is

set n0 [$ns node]

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

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

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction.

To define a directional link instead of a bi-directional one, we should replace - duplex-link by -simplex-link.

In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

#set Queue Size of link (n0-n2) to 20

The wireless model essentially consists of the mobile node at the core, with additional supporting features that allows simulations of multi-hop ad-hoc networks, wireless LANs etc.

From the topology, a basic idea of the network to be simulated is obtained. For that topology, the node-config parameters are set first. This is shown below.

```
$ns  node-config    -adhocRouting       $val(rp) \
                    -llType             $val(ll) \
                    -macType            $val(mac) \
                    -ifqType            $val(ifq) \
                    -ifqLen             $val(ifqlen) \
                    -antType            $val(ant) \
                    -propType           $val(prop) \
                    -phyType            $val(netif) \
                    -channelType        $val(chan) \
                    -topoInstance       $topo \
                    -agentTrace         ON \
                    -routerTrace        OFF \
                    -macTrace           OFF \
                    -movementTrace      OFF
```

After setting the node configuration parameters, the actual nodes are created.

```
for { set j 0 } { $j < $val(nn)} {incr j} {
        set node_($j) [ $ns_ node ]
        $node_($i) random-motion 0 ;# disable random motion
        }
```

Thus the mobile nodes are created.

Setting Mobile Node Movements: The mobile node is designed to move in a three-dimensional topology. However, the third dimension (Z) is not used. That is the mobile node is assumed to move always on a flat terrain with Z always equal to 0. Thus the mobile node has X, Y, Z(=0) co-ordinates that is continually adjusted as the node moves. There are two mechanisms to induce movement in mobile nodes. In the first method, starting position of the node and its future destinations may be set explicitly. These directives are normally included in a separate movement scenario file.

The start-position and future destinations for a mobile node may be set by using the following APIs:

```
$node set X_ <x1>
```

$node set Y_ <y1>
$node set Z_ <z1>
$ns at $time $node setdest<x2><y2><speed>

At $time sec, the node would start moving from its initial position of (x1,y1) towards a destination (x2,y2) at the defined speed. In this method the node-movement-updates are triggered whenever the position of the node at a given time is required to be known. This may be triggered by a query from a neighboring node seeking to know the distance between them, or the setdest directive described above that changes the direction and speed of the node.

Topology Definition: Irrespective of the methods used to generate node movement, the topography for mobile nodes needs to be defined. It should be defined before creating mobile nodes. Normally flat topology is created by specifying the length and width of the topography using the following
primitive:
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
where val(x) and val(y) are the boundaries used in simulation

Following is a list of commands used in wireless simulations:
$ns_ node-config -addressing Type <usually flat or hierarchical used forwireless topologies>
-adhocRouting<adhocrotuing protocol like DSDV, DSR,
-llType<LinkLayer>
-macType<MAC type like Mac/802_11>
-propType<Propagation model likePropagation/TwoRayGround>
-ifqType<interface queue type likeQueue/DropTail/PriQueue>
-ifqLen<interface queue length like 50>
-phyType<network inteface type likePhy/WirelessPhy>
-antType<antenna type like Antenna/OmniAntenna>
-channelType<Channel type like Channel/WirelessChannel>
-topoInstance<the topography instance>
-wiredRouting<turning wired routing ON or OFF>
-mobileIP<setting the flag for mobileIP ON or OFF>
-energyModel<EnergyModel type>
-initialEnergy<specified in Joules>
-rxPower<specified in W>
-txPower<specified in W>

-agentTrace<tracing at agent level turned ON or OFF>
-routerTrace<tracing at router level turned ON or OFF>
-macTrace<tracing at mac level turned ON or OFF>
-movementTrace<mobilenode movement logging turnedON or OFF>

Algorithm:

1. Initialize variables
2. Create a simulator object
3. Create tracing and animation file
4. Create topography
5. Create GOD - General Operations Director
6. Create nodes
7. Create channel (Communication PATH)
8. Position of the nodes (Wireless nodes needs a location)
9. Any mobility codes (if the nodes are moving)
10. Run the simulation

NS2 INSTALLATION ON UBUNTU 16.04: The following are the steps to install NS-2 on Ubuntu 16.04:

1) Download 'ns-allinone-2.35' from:
http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/ns-allinone-2.35.tar.gz/ download
2) Copy the downloaded zip file 'ns-allinone-2.35.tar.gz file' to home Folder and extract.
3) A) Type following commands on terminal:
   i) sudo apt-get update
   ii) sudo apt-get install build-essential autoconfautomake
   iii) sudo apt-get install tcl8.5-dev tk8.5-dev
   iv) sudo apt-get install perlxgraphlibxt-dev libx11-dev libxmu-dev
4) Now open file ls.h from /home/ns-allinone-2.35/ns-2.35/linkstatein the file go to line no. 137 (void eraseAll() { erase(baseMap::begin(), baseMap::end()); })
Replace with voideraseAll() { **this**->erase(baseMap::begin(), baseMap::end()); }
5) Now change your directory (here i have already extracted the downloaded files to desktop, so my location is desktop) type the following codes in the command window to install NS2.
6) cd ns-allinone-2.35
7) ./install
The installation procedure will take a few minutes.....
8) After completing the installation type the following command in the command window
gedit ~/.bashrc

9) Now an editor window appears, please copy and paste the following codes **in the end of the text** file (note that '/home/c05l0202/ns-allinone-2.35/otcl-1.14' in each line in the below code should be replaced with your location where the 'ns-allinone-2.35.tar.gz'file is extracted)

```
# LD_LIBRARY_PATH
OTCL_LIB=/home/c05l0202/ns-allinone-2.35/otcl-1.14

NS2_LIB=/home/c05l0202/ns-allinone-2.35/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
ExportLD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$X11_LIB:$USR_L
OCAL_LIB
# TCL_LIBRARY
TCL_LIB=/home/c05l0202/ns-allinone-2.35/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
# PATH

XGRAPH=/home/c05l0202/ns-allinone-2.35/bin:/home/c05l0202/ns-allinone-
2.35/tcl8.5.10/unix:/home/c05l0202/ns-allinone-2.35/tk8.5.10/unix
NS=/home/c05l0202/ns-allinone-2.35/ns-2.35/
NAM=/home/c05l0202/ns-allinone-2.35/nam-1.15/
PATH=$PATH:$XGRAPH:$NS:$NAM
```

10) Save and close the text editor and then type the following command on the terminal

```
source ~/.bashrc
```

11) Close the terminal window and start a new terminal window and now change the directory to ns-2.35and validate ns-2.35 by executing the following command

```
cd ns-allinone-2.35
cd ns-2.35
./validate
```

12) If the installation is successful, then you will be able to see % at the command prompt while typing the following command

```
ns
```

13) Checknam, xgraph too

14) Now type Exit

Conclusion: Thus, we have studied, installed and configured NS2 on ubuntu machine and also simulate wireless scenario.

FAQs:

1. Distinguish emulation and simulation with example.
2. Compare complier and interpreter.
3. List the different simulation tools and compare it with network simulator-2 (NS-2).
4. Why two language (oTCL and C++) used by NS2?
5. What is the role of GOD?

Assignment No. 2

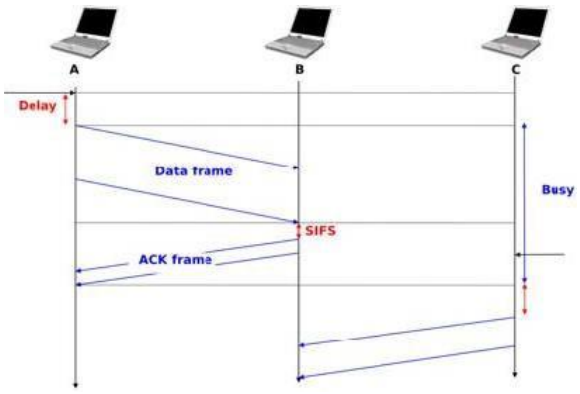
Aim: To create a wireless network scenario and study the performance of CSMA/CA protocol through simulation tools like NetSim or NS2 or QualNet.

Theory:

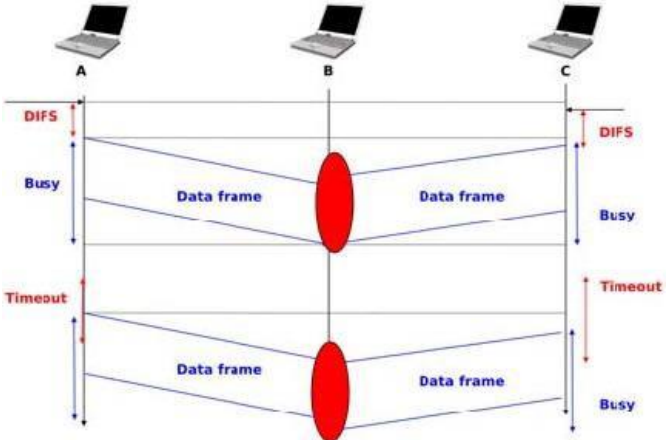
The Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) Medium Access Control algorithm was designed for the popular WiFi wireless network technology [802.11]. CSMA/CA also senses the transmission channel before transmitting a frame. Furthermore, CSMA/CA tries to avoid collisions by carefully tuning the timers used by CSMA/CA devices. CSMA/CA uses acknowledgements like CSMA. Each frame contains a sequence number and a CRC. The CRC is used to detect transmission errors while the sequence number is used to avoid frame duplication. When a device receives a correct frame, it returns a special acknowledgement frame to the sender. CSMA/CA introduces a small delay, named Short Inter Frame Spacing (SIFS), between the reception of a frame and the transmission of the acknowledgement frame. This delay corresponds to the time that is required to switch the radio of a device between the reception and transmission modes.

Compared to CSMA, CSMA/CA defines more precisely when a device is allowed to send a frame. First, CSMA/CA defines two delays: DIFS and EIFS. To send a frame, a device must first wait until the channel has been idle for at least the Distributed Coordination Function Inter Frame Space (DIFS) if the previous frame was received correctly. However, if the previously received frame was corrupted, this indicates that there are collisions and the device must sense the channel idle for at least the Extended Inter Frame Space (EIFS), with SIFS < DIFS < EIFS. The exact values for SIFS, DIFS and EIFS depend on the underlying physical layer.

The figure below shows the basic operation of CSMA/CA devices. Before transmitting, host A verifies that the channel is empty for a long enough period. Then, its sends its data frame. After checking the validity of the received frame, the recipient sends an acknowledgement frame after a short SIFS delay. Host C, which does not participate in the frame exchange, senses the channel to be busy at the beginning of the data frame. Host C can use this information to determine how long the channel will be busy for. Note that as SIFS < DIFS < EIFS, even a device that would start to sense the channel immediately after the last bit of the data frame could not decide to transmit its own frame during the transmission of the acknowledgement frame.

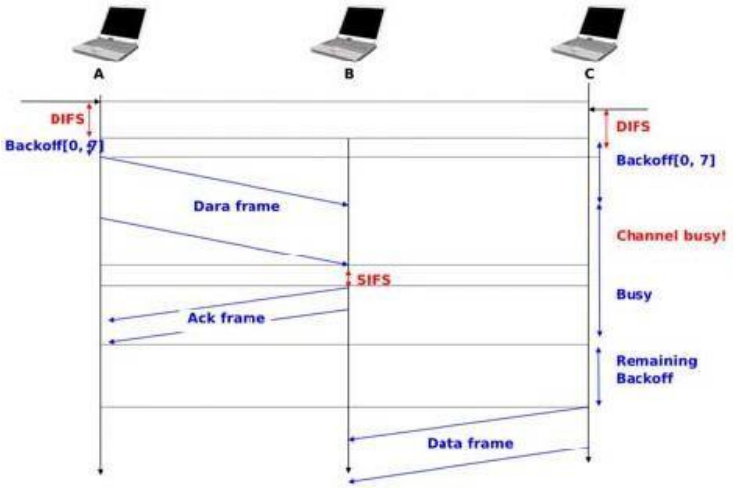


The main difficulty with CSMA/CA is when two or more devices transmit at the same time and cause collisions. This is illustrated in the figure below, assuming a fixed timeout after the transmission of a data frame. With CSMA/CA, the timeout after the transmission of a data frame is very small, since it corresponds to the SIFS plus the time required to transmit the acknowledgement frame.



To deal with this problem, CSMA/CA relies on a backoff timer. This backoff timer is a random delay that is chosen by each device in a range that depends on the number of retransmissions for the current frame. The range grows exponentially with the retransmissions as in CSMA/CD. The minimum range for the backoff timer is [0, 7 ∗ slotT ime] where the slotTime is a parameter that depends on the underlying physical layer. Compared to CSMA/CD ' s exponential backoff, there are two important differences to notice. First, the initial range for the backoff timer is seven times larger. This is because it is impossible in CSMA/CA to detect collisions as they happen. With CSMA/CA, a collision may affect the entire frame while with CSMA/CD it can only affect the beginning of the frame. Second, a CSMA/CA device must regularly sense
the transmission channel during its back off timer. If the channel becomes busy (i.e. because another device

is transmitting), then the back off timer must be frozen until the channel becomes free again. Once the channel becomes free, the back off timer is restarted. This is in contrast with CSMA/CD where the back off is recomputed after each collision. This is illustrated in the figure below. Host A chooses a smaller backoff than host C. When C senses the channel to be busy, it freezes its backoff timer and only restarts it once the channel is free again.



Another problem faced by wireless networks is often called the hidden station problem. In a wireless network, radio signals are not always propagated same way in all directions. For example, two devices separated by a wall may not be able to receive each other's signal while they could both be receiving the signal produced by a third host. The other example, two devices that are on different sides of a hill may not be able to receive each other's signal while they are both able to receive the signal sent by a station at the top of the hill. Furthermore, the radio propagation conditions may change with time.

To avoid collisions in these situations, CSMA/CA allows devices to reserve the transmission channel for some time. This is done by using two control frames: Request To Send (RTS) and Clear To Send (CTS). Both are very short frames to minimize the risk of collisions. To reserve the transmission channel, a device sends a RTS frame to the intended recipient of the data frame. The RTS frame contains the duration of the requested reservation. The recipient replies, after a SIFS delay, with a CTS frame which also contains the duration of the reservation. As the duration of the reservation has been sent in both RTS and CTS, all hosts that could collide with either the sender or the reception of the data frame are informed of the reservation. They can compute the total duration of the transmission and defer their access to the transmission channel until then. The utilization of the reservations with CSMA/CA is an optimization that is useful when collisions are frequent. If there are few collisions, the time required to transmit the RTS and CTS frames can become significant and in particular when short frames are exchanged. Some devices only turn on RTS/CTS after transmission errors.

Algorithm:

1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create multiple nodes that forms a network numbered from 0 to ∞ (infinity)
5. Create duplex links between the nodes and add Orientation to the nodes for setting a LAN topology
6. Setup TCP Connection between some nodes
7. Apply FTP Traffic over TCP
8. Setup UDP Connection between some nodes
9. Apply CBR Traffic over UDP
10. Apply CSMA/CA and CSMA/CD mechanisms and study their performance
11. Schedule events and run the program

Conclusion: Thus, we have studied the performance of CSMA/CA protocol through simulation.

FAQs:

1. Compare CSMA/CA and CSMA/CD.
2. What is the function of MAC layer?
3. Draw the structure of trace files and explain it briefly.

Assignment No. 3

Aim: Write a program to implement routing protocol for ad-hoc network with multiple nodes using simulation tools like NetSim or NS2 or QualNet

Theory:

The wireless model essentially consists of the mobile node at the core, with additional supporting features that allows simulations of multi-hop ad-hoc networks, wireless LANs etc. For creating the wireless network, we shall describe the internals of mobile node, its routing mechanisms, the routing protocols dsdv, aodv, tora and dsr, creation of network stack allowing channel access in mobile node. Nodes are configured with the components of channel, networking interface, radio propagation model, MAC protocol, adhoc routing protocol, interface queue, link layer, topography object, and antenna type.

The Ad hoc On-Demand Distance Vector (AODV) routing protocol offers an ability of quick adaption to dynamic link conditions, low processing and memory overhead, low network utilization and determines unicast routes to destinations within the ad-hoc network. It uses destination sequence numbers to ensure the elimination of loops, and consequently the counting to infinity problem, at all times, thus avoiding related problems associated with classical distance vector protocols. In AODV protocol, route discovery is done by broadcasting the RREQ message to neighbor nodes with the requested destination address and a sequence number, which prevents the old messages to be forwarded to nodes. The sequence number acts as timestamps and prevents this AODV protocol from the loop problem. The destination sequence number for each possible destination host is stored in the routing table. The destination sequence numbers are updated in the routing table when the host receives the message with the greater sequence number. The route request message (RREQ) is sent when the host does not know the route to the needed destination host. When a host receives RREQ message, it checks the time period between the last RREQ messages from the same host and discards the message if it is under the specified limit. Next host increases the hop count by one in the RREQ message and makes update in own routing table basing on the sequence number and the requested host's address. Also the hop count is copied from the RREQ message. Finally the host decrease request message TTL field in the IP header and broadcasts it. Each route request message increases path hop count and passed hosts make update in their own routing table about the requested host. This information helps the destination reply to be easily routed back to the requested host. The route reply uses RREP message that can be only be generated by the destination host or the hosts who have the information that the destination host is alive. The host can generate the route reply message (RREP) if the destination is the host itself or if the route to the destination is valid and has the same or greater destination sequence number, but only if the D field is not set. D field in the RREQ message indicates that only the destination host can reply to the RREQ message. When generating the RREP message host copies the destination address and the requested host's sequence number to the corresponding RREP message's fields. If the receiver is the destination host
then its own sequence number is incremented and copied to the destination sequence number field. In

addition, the hop count is set to zero and the lifetime field of the RREP message is set to the initial Page |18 timeout value of the host. If the receiver is the intermediate host, then it just copies the destination sequence number from the routing table and adds the host address from where it has received RREQ message to the destination address field. When the RREP message is created it is sent to the requested hop in order to be delivered to the requested host. The hop count metric is incremented along the path, so at the end, it corresponds to the actual distance between the hosts. Although AODV is a reactive protocol it uses the Hello messages periodically to inform its neighbors that the link to the host is alive. The Hello messages are broadcasted with TTL equals to 1, so that the message will not be forwarded further. When host receives the Hello message it updates the lifetime of the host information in the routing table. If the host does not get information from the host's neighbor for specified amount of time, then the routing information in the routing table is marked as lost.

For that topology, the node-config parameters are set first. This is shown below.

```
$ns  node-config    -adhocRouting       $val(rp) \
                      -llType             $val(ll) \
                      -macType            $val(mac) \
                      -ifqType            $val(ifq) \
                      -ifqLen             $val(ifqlen) \
                      -antType            $val(ant) \
                      -propType           $val(prop) \
                      -phyType            $val(netif) \
                      -channelType        $val(chan) \
                      -topoInstance       $topo \
                      -agentTrace         ON \
                      -routerTrace        OFF \
                      -macTrace           OFF \
                      -movementTrace      OFF
```

After setting the node configuration parameters, the actual nodes are created.

```
for { set j 0 } { $j < $val(nn)} {incr j} {
        set node_($j) [ $ns_ node ]
        $node_($i) random-motion 0 ;# disable random motion
        }
```

Thus the mobile nodes are created.

$ns_ node-config -addressing Type <usually flat or hierarchical used for wireless topologies>
-adhocRouting<adhocrotuing protocol like DSDV, DSR,
-llType<LinkLayer>
-macType<MAC type like Mac/802_11>
-propType<Propagation model likePropagation/TwoRayGround>
-ifqType<interface queue type likeQueue/DropTail/PriQueue>
-ifqLen<interface queue length like 50>
-phyType<network inteface type likePhy/WirelessPhy>
-antType<antenna type like Antenna/OmniAntenna>
-channelType<Channel type like Channel/WirelessChannel>
-topoInstance<the topography instance>
-wiredRouting<turning wired routing ON or OFF>
-mobileIP<setting the flag for mobileIP ON or OFF>
-energyModel<EnergyModel type>
-initialEnergy<specified in Joules>
-rxPower<specified in W>
-txPower<specified in W>
-agentTrace<tracing at agent level turned ON or OFF>
-routerTrace<tracing at router level turned ON or OFF>
-macTrace<tracing at mac level turned ON or OFF>
-movementTrace<mobilenode movement logging turnedON or OFF>

After the basic network setup is done, the next thing to do is to setup traffic agents such as TCP and UDP, traffic sources such as FTP and CBR, and attach them to nodes and agents respectively.

- **set *tcp* [new *Agent/TCP*]:** This line shows how to create a TCP agent. But in general, users can create any agent or traffic sources in this way. Agents and traffic sources are in fact basic objects (not compound objects), mostly implemented in C++ and linked to OTcl. Therefore, there are no specific Simulator object member functions that create these object instances. To create agents or traffic sources, a user should know the class names these objects (Agent/TCP, Agnet/TCPSink, Application/FTP and so on).

- **$ns attach-agent node agent:** The attach-agent member function attaches an agent object created to a node object. Actually, what this function does is call the attach member function of specified node, which attaches the given agent to itself. Therefore, a user can do the same thing by, for example, $n0 attach $tcp. Similarly, each agent object has a member function attach-agent that attaches a traffic

source object to itself.

- **$ns connect agent1 agent2:** After two agents that will communicate with each other are created, the next thing is to establish a logical network connection between them. This line establishes a network connection by setting the destination address to each others' network and port address pair.

Assuming that all the network configuration is done, the next thing to do is write a simulation scenario (i.e. simulation scheduling). The Simulator object has many scheduling member functions. However, the one that is mostly used is the following:

- **$ns at time "string":** This member function of a Simulator object makes the scheduler (scheduler_ is the variable that points the scheduler object created by [new Scheduler] command at the beginning of the script) to schedule the execution of the specified string at given simulation time. For example, $ns at 0.1 "$cbr start" will make the scheduler call a start member function of the CBR traffic source object, which starts the CBR to transmit data. In NS, usually a traffic source does not transmit actual data, but it notifies the underlying agent that it has some amount of data to transmit, and the agent, just knowing how much of the data to transfer, creates packets and sends them.

The following is description of Tcl Commands used in program:

1. Initialize the network configuration parameters of the network to be simulated using the 'set' keyword and the 'val()' keyword.
2. While initializing, the super class of the particular class is also denoted. The example of an initialization is "set val(chan) Channel/WirelessChannel"
3. The values of link layer type, interface queue length, the number of nodes and the routing protocol variables can be directly given without any superclass.

ex.: set val(ll) LL
set val(nn) 6

4. Create an object (say, ns) for the Simulator class using the 'set' keyword, and 'new' keyword.

"set ns [new Simulator]"

5. Create and open a trace file with write mode (i.e., w) using 'set' and 'open' keywords.

"settf [open wireless.tr w] "

6. Using the 'trace-all' function, set the ns object to trace all the events and write the trace file created.
7. Create a nam file in write mode and using 'namtrace-all-wireless' function, with arguments as the nam file handler and the topography values (here, 500 x 500), the events traceds are put in nam file.
8. Create an object for the Topography class and give the values (500 x 500) in the 'load_flatgrid' function.
9. Pass the value of the number of nodes to the 'create-god' function.

10. Set the values for the arguments of the 'node-config' function with all the configuration parameter values.
11. Create the nodes 0, 1, 2, 3, 4 and 5 using the 'set' and '$ns' keywords shown.
    "set node0 [$ns node]"
12. Set the position of the nodes in the X_, Y_ and Z_ variables using 'set' keyword.
13. Create the TCP agent and attach it to node0 using the 'attach-agent' function.
14. Similarly create the object for FTP Application and attach it to the TCP object using 'attach-agent' function.
15. Create the object for the TCP Sink and attach it to the node5 using 'attach-agent' function.
16. Connect the TCP object to the TCPSink object using the 'connect' function.
17. Set the movement of node1 using 'at' keyword, specifying the time in seconds and the position in x, y, z values using 'setdest' function.
18. The application start time and stop time are specified using the 'at' keyword along with the 'start' and 'stop' keywords.
19. The Simulation stop time is indicated using the 'at' keyword and call the 'finish' procedure.
20. Inside the 'finish' procedure, set the values ns, tracefile and tracenam as 'global'.
21. Call the 'flush-trace' function to write all the events traced in the trace files.
22. Close the trace file using 'close' function.
23. Execute the nam file using 'exec' keyword and put & at the end of the line.
24. Call the 'exit' function with parameter '0', and close the 'finish' procedure.
25. Run the simulation using the 'run' keyword.
26. Close the editor and run the .tcl file to get the output. Thus the simulation of adhoc networks is made.

Conclusion: Thus, we have studied and simulated multiple wireless nodes with mobility and send the packet from source node to destination node.

FAQs:

1. Explain TCP and UDP agent with example.
2. List different applications of ad-hoc wireless network. Explain any two.
3. Explain issues in designing a routing protocol for ad hoc wireless networks

Assignment No. 4

Aim: Study the security permissions for applications in android phones. Either demonstrates Android security permission configurations or Write the android app to demonstrate permissions usage control in android phones.

Theory:

1.0 The purpose of permission is to protect the privacy of an Android user. Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features(such as camera and internet). Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.

A central design point of the Android security architecture is that no app, by default, has permission to perform any operations that would adversely impact other apps, the operating system, or the user. This includes reading or writing the user's private data (such as contacts or emails), reading or writing another app's files, performing network access, keeping the device and so on.

Permission approval example

An app must publicize the permissions it requires by including <uses-permission>tags in the app manifest. For example, an app that needs to send SMS messages would have this line in the manifest:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.snazzyapp">
    <uses-permission android:name="android.permission.SEND_SMS"/>

    <application ...>
        ...
    </application>
  </manifest>
```

If your app lists *normal* permissions in its manifest (that is, permissions that don't pose much risk to theuser's privacy or the device's operation), the system automatically grants those permissions to your app.

If your app lists *dangerous* permissions in its manifest (that is, permissions that could potentially affectthe user's privacy or the device's normal operation), such as the SEND_SMS permission above, the usermust explicitly agree to grant those permissions.

2.0 Android Security permissions best practices:

Permission requests protect sensitive information available from a device and should only beused when access to information is necessary for the functioning of your app. This documentprovides tips on ways you might be able to achieve the same (or better) functionality withoutrequiring access to such information; it is not an exhaustive discussion of how permissions workin the Android operating system.

Android permissions recommendations

#1: Only use the permissions necessary for your app to work. Depending on how you are usingthe permissions, there may be another way to do what you need (system intents, identifiers,backgrounding for phone calls) without relying on access to sensitive information.

#2: Pay attention to permissions required by libraries. When you include a library, you alsoinherit its permission requirements. You should be aware of what you're including, thepermissions they require, and what those permissions are used for.

#3: Be transparent. When you make a permissions request, be clear about what you're accessing, and why, so users can make informed decisions. Make this information availablealongside the permission request including install, runtime, or update permission dialogues.

#4: Make system accesses explicit. Providing continuous indications when you access sensitivecapabilities (for example, the camera or microphone) makes it clear to users when you're collecting data and avoids the perception that you're collecting data surreptitiously.

3.0 List of Permissions and meaning

1. Make phone calls

Permission Type: Software
URI: android.permission.CALL_PHONE
Risk: HIGH
Protection level: DANGEROUS
Official Description:
Allows an application to initiate a phone call without going through the Dialer user interface for the userto confirm the call being placed.
Details:
This permission is of high importance. This could let an application call a 1-900 number and charge youmoney.

2. Send SMS or MMS

Permission Type: Software
URI: android.permission.SEND_SMS
Risk: HIGH
Protection level: DANGEROUS

Official Description:
Allows an application to send SMS messages.
Details:
This permission is of high importance. This could let an application send an SMS on your behalf, andmuch like the phone call permission, it could cost you money by sending SMS to for-pay numbers.

3. Modify/delete SD card contents
Permission Type: Software
URI: android.permission.WRITE_EXTERNAL_STORAGE
Risk: MEDIUM
Protection level: DANGEROUS
Official Description:
Allows an application to write to external storage.
Details:
This permission is of high importance. This will allow applications to read, write, and delete anythingstored on your phone's SD card. This includes pictures, videos, mp3s, documents, and even data writtento your SD card by other applications.

4. Read Contacts
Permission Type: Software
URI: android.permission.READ_CONTACTS
Risk: MEDIUM-HIGH
Protection level: DANGEROUS
Official Description:
Allows an application to read the user's contacts data.
Details:
This permission is of high importance. Unless an app explicitly states a specific feature that it would useyour contact list for, there isn't much of a reason to give an application this permission.

5. Write contact data
Permission Type: Software
URI: android.permission.WRITE_CONTACTS
Risk: MODERATE-HIGH
Protection level: DANGEROUS
Official Description
Allows an application to write (but not read) the user's contacts data.

Details:
This permission is of high importance. Unless an app explicitly states a specific feature that it would useyour contact list for, there isn't much of a reason to give an application this permission.

6. Read calendar data
Permission Type: Software
URI: android.permission.READ_CALENDAR
Risk: MEDIUM
Protection level: DANGEROUS
Official Description:
Allows an application to read the user's calendar data.
Details:
This permission is of moderate to high importance. While most people would consider their calendarinformation slightly less important than their list of contacts and friends, this permission should still betreated with care when allowing applications access.

7. Write calendar data
Permission Type: Software
URI: android.permission.WRITE_CALENDAR
Risk: MEDIUM
Protection level: DANGEROUS
Official Description:
Allows an application to write (but not read) the user's calendar data.
Details:
This permission is of moderate to high importance. While most people would consider their calendarinformation slightly less important than their list of contacts and friends, this permission should still betreated with care when allowing applications access.

8. Read browser history & bookmarks
Permission Type: Software
URI: com.android.browser.permission.READ_HISTORY_BOOKMARKS
Risk: MEDIUM-HIGH
Protection level: DANGEROUS
Official Description:
Allows an application to read (but not write) the user's browsing history and bookmarks.
Details:

This permission is of medium-high importance. Browsing habits are often tracked through regularcomputers, but with this permission, you'd be giving access to more than just browsing habits.

9. Write browser history & bookmarks

Permission Type: Software
URI: com.android.browser.permission.WRITE_HISTORY_BOOKMARKS
Risk: MODERATE-HIGH
Protection level: DANGEROUS
Official Description:
Allows an application to write (but not read) the user's browsing history and bookmarks.
Details:
This permission is of medium-high importance. Browsing habits are often tracked through regularcomputers, but with this permission, you'd be giving access to more than just browsing habits.

10. Read sensitive logs

Permission Type: Software
URI: android.permission.READ_LOGS
Risk: VERY-HIGH
Protection level: DEVELOPMENT
Official Description:
Allows an application to read the low-level system log files.
Details:
This permission is of high importance. This allows the application to read what any other applicationshave logged.

11. Modify global system settings

Permission Type: Hardware
URI: android.permission.WRITE_SETTINGS
Risk: MEDIUM
Protection level: DANGEROUS
Official Description:
Allows an application to read or write the system settings
Details:
This permission is pretty important but only has the possibility of moderate impact. Global settings arepretty much anything you would find under Android's main 'settings' window. However, a lot of thesesettings may be perfectly reasonable for an application to change.

12. Read sync settings

Permission Type: Hardware
URI: android.permission.READ_SYNC_SETTINGS
Risk: LOW-MODERATE
Protection level: UNKNOWN
Official Description:
Allows applications to read the sync settings
Details:
This permission is of low to medium importance. It mostly allows the application to know if you havebackground data sync (such as for Facebook or Gmail) turned on or off.

13. Automatically start at boot

Permission Type: Hardware
URI: android.permission.RECEIVE_BOOT_COMPLETED
Risk: MODERATE-HIGH
Protection level: UNKNOWN
Official Description:
Allows an application to receive the ACTION_BOOT_COMPLETED that is broadcast after the systemfinishes booting.
Details:
This permission is of low to moderate impact. It will allow an application to tell Android to run theapplication every time you start your phone. While not a danger in and of itself, it can point to applications intent.

14. Restart other applications

Permission Type: Hardware
URI: android.permission.RESTART_PACKAGES
Risk: HIGH
Protection level: UNKNOWN
Official Description:
This constant is deprecated. The restart Package(String) API is no longer supported.
Details:
This permission is of low to moderate impact. It will allow an application to tell Android to ‘kill’ theprocess of another application. However, any app that is killed will likely get restarted by the Android OSitself.

15. Retrieve running applications

Permission Type: Hardware
URI: android.permission.GET_TASKS
Risk: MEDIUM-HIGH
Protection level: DANGEROUS
Official Description:
Allows an application to get information about the currently or recently running tasks: a thumbnailrepresentation of the tasks, what activities are running in it, etc.
Details:
This permission is of moderate importance. It will allow an application to find out what other applicationsare running on your phone. While not a danger in and of itself, it would be a useful tool for someonetrying to steal your data.

16. Display system-level alerts

Permission Type: Hardware
URI: android.permission.SYSTEM_ALERT_WINDOW
Risk: HIGH
Protection level: DANGEROUS
Official Description:
Allows an application to open windows using the type TYPE_SYSTEM_ALERT, shown on top ofallother applications.
Details:
This permission is of high importance. This permission allows an app to show a “popup” window aboveall other apps, even if the app is not in the foreground. A malicious developer/advertiser could use it toshow very obnoxious advertising.

17. Control vibrator

Permission Type: Hardware
URI: android.permission.VIBRATE
Risk: LOW
Protection level: UNKNOWN
Official Description:
Allows access to the vibrator
Details:
This permission is of low importance. As it states, it lets an app control the vibrate function on yourphone. This includes incoming calls and other events.

18. Take pictures and videos

Permission Type: Hardware
URI: android.permission.CAMERA
Risk: MODERATE-HIGH
Protection level: DANGEROUS
Official Description:
Required to be able to access the camera device.
Details:
This permission is of moderate importance. As it states, it lets an app control the camera function on yourphone. In theory, this could be used maliciously to snap unsuspecting photos, but it would be unlikely anddifficult to get a worthwhile picture or video. However, it is not impossible to make malicious use ofcameras.

19. Access location extra commands

Permission Type: Network
URI: android.permission.ACCESS_LOCATION_EXTRA_COMMANDS
Risk: MEDIUM-HIGH
Protection level: UNKNOWN
Official Description:
Allows an application to access extra location provider commands
Details:
The specifics of the extra commands here are a bit unclear. However, the usage of this permissionindicates that an app wants to know detailed information about your location, and respond accordingly.

20. Access mock location

Permission Type: Network
URI: android.permission.ACCESS_MOCK_LOCATION
Risk: MODERATE
Protection level: DANGEROUS
Official Description:
Allows an application to create mock location providers for testing
Details:
This is permission used for development of apps that make use of location-based services. By creating“mock” (fake) locations, apps can test if their code works correctly depending on your location.

21. Battery stats

Permission Type: Hardware
URI: android.permission.BATTERY_STATS
Risk: LOW
Protection level: UNKNOWN
Official Description:
Allows an application to collect battery statistics
Details:
This permission is of little to no importance.

22. Bluetooth Admin

Permission Type: Software
URI: android.permission.BLUETOOTH_ADMIN
Risk: MEDIUM
Protection level: DANGEROUS
Official Description:
Allows applications to discover and pair Bluetooth devices
Details:
Bluetooth Wikipedia is a technology that lets your phone communicate wirelessly over short distances. Itis similar to Wi-Fi in many ways. It itself is not a danger to your phone, but it does enable a way for anapplication to send and receive data from other devices.

23. Broadcast Sticky (Intents)

Permission Type: Hardware
URI: android.permission.BROADCAST_STICKY
Risk: LOW-MEDIUM
Protection level: UNKNOWN
Official Description:
Allows an application to broadcast sticky intents. These are broadcasts whose data is held by the systemafter being finished so that clients can quickly retrieve that data without having to wait for the nextbroadcast.
Details:
The permission has to do with how applications “talk” to each other using a communication methodcalled “Intents”. While this permission is highly technical it is relatively low importance. There are noknow obvious malicious uses for this permission.

24. Change Configuration

Permission Type: Hardware
URI: android.permission.CHANGE_CONFIGURATION
Risk: MEDIUM-HIGH
Protection level: DANGEROUS
Official Description:
Allows an application to modify the current configuration, such as locale.
Details:
This is a permission that generally should not be granted to regular apps. Other than changing the locale(i.e. language), it is unclear what configuration changes this permission allows. As such, it should betreated with considerable caution.

25. Clear app cache
Permission Type: Hardware
URI: android.permission.CLEAR_APP_CACHE
Risk: LOW
Protection level: DANGEROUS
Official Description:
Allows an application to clear the caches of all installed applications on the device.
Details:
This permission is of low importance. It allows an app to clear the cache of apps on the phone or tablet. Acache is a place that app stores recently used data for faster access. Clearing the cache can sometimes(very rarely) fix bugs related to those files.

26. Disable Keyguard (lock screen)
Permission Type: Hardware
URI: android.permission.DISABLE_KEYGUARD
Risk: MEDIUM-HIGH
Protection level: DANGEROUS
Official Description:
Allows applications to disable the keyguard
Details:
This permission is of medium-high importance. It allows an app to disable the “lock screen” that mostphones go into after going to sleep and been turned on again. This lock screen can sometimes be apassword screen, or a PIN screen, or just a “slide to unlock” screen.

27. Expand status bar

Permission Type: Hardware
URI: android.permission.EXPAND_STATUS_BAR
Risk: MEDIUM-HIGH
Protection level: UNKNOWN
Official Description:
Allows an application to expand or collapse the status bar.
Details:
This appears to be system permission — not for use by regular applications. If you come across thispermission I would beware of any app requesting it that is not an Android system app.

28. Flashlight
Permission Type: Hardware
URI: android.permission.FLASHLIGHT
Risk: LOW
Protection level: UNKNOWN
Official Description:
Allows access to the flashlight
Details:
This allows apps to turn on or off the LED “flash” light used by the camera. This is a handy tool butusually of no risk itself.

29. Get package size
Permission Type: Hardware
URI: android.permission.GET_PACKAGE_SIZE
Risk: LOW-MODERATE
Protection level: UNKNOWN
Official Description:
Allows an application to find out the space used by any package.
Details:
This permission does not seem to have any risk associated with it.

30. Modify audio settings
Permission Type: Hardware
URI: android.permission.MODIFY_AUDIO_SETTINGS
Risk: LOW
Protection level: DANGEROUS

Official Description:
Allows an application to modify global audio settings
Details:
This permission is of low importance. Audio settings pose little to no risk to the device.

31. Format file systems
Permission Type: Software
URI: android.permission.MOUNT_FORMAT_FILESYSTEMS
Risk: MEDIUM
Protection level: DANGEROUS
Official Description:
Allows formatting file systems for removable storage.
Details:
The primary danger with this permission is that it could be used to erase data from an SD card or othersimilar storage in your phone. This is also not permitted by any normal app should need.

32. NFC (Near Field Communication)
Permission Type: Software
URI: android.permission.NFC
Risk: MEDIUM
Protection level: DANGEROUS
Official Description:
Allows applications to perform I/O operations over NFC
Details:
NFC stands for Near Field Communication. This is a technology like Bluetooth that enables short-rangecommunication between two devices or the reading of NFC “tags”.

33. Process outgoing calls
Permission Type: Software
URI: android.permission.PROCESS_OUTGOING_CALLS
Risk: VERY-HIGH
Protection level: DANGEROUS
Official Description:
Allows an application to monitor, modify, or abort outgoing calls.
Details:

This permission is of high importance. This would allow an app to see what numbers are called and otherpersonal info. Generally, this permission should only be seen on apps for VOIP (Voice Over InternetProtocol) like Google Voice or dialer replacement type apps.

34. Record audio

Permission Type: Software
URI: android.permission.RECORD_AUDIO
Risk: MODERATE-HIGH
Protection level: DANGEROUS
Official Description:
Allows an application to record audio
Details:
While this permission is not typically dangerous, it is a potential tool for eavesdropping. Howeverrecording audio has legitimate uses such as note taking apps or voice search apps. As a side noterecording audio is typically a significant drain on the battery.

35. Set alarm

Permission Type: Hardware
URI: android.permission.SET_ALARM
Risk: LOW
Protection level: UNKNOWN
Official Description:
Allows an application to broadcast an Intent to set an alarm for the user.
Details:
This permission seems to be of low risk because it doesn't allow the setting of the alarm directly. Rather itallows the opening of the alarm app on the phone.

36. Read profile

Permission Type: Software
URI: android.permission.READ_PROFILE
Risk: MEDIUM-HIGH
Protection level: DANGEROUS
Official Description:
Allows an application to read the user's personal profile data.
Details:

This a new permission that relates to a special new "Me" contact you can create in your phone or tablet asyour own profile.

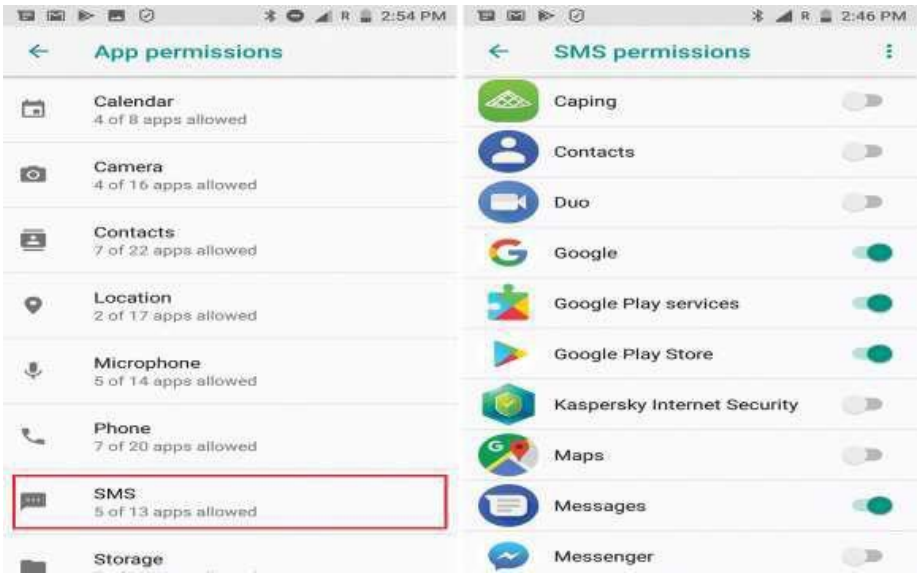
37. Read external storage

Permission Type: Software
URI: android.permission.READ_EXTERNAL_STORAGE
Risk: LOW
Protection level: UNKNOWN
Official Description:
Allows an application to read from external storage.
Details:
This permission is granted to all apps by default.

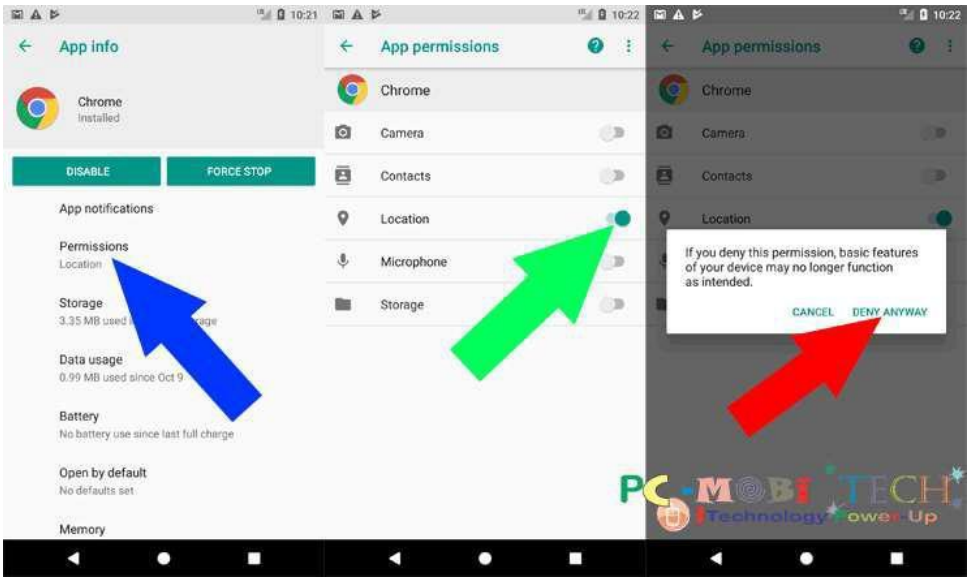
4.0 How to toggle permission on android phones

1. To start, head to Settings > App and find an app which you want to work with.
2. Select the app.
3. Tap App Permissions on the App Info screen.
4. You will see a list of permissions the app requests, and whether those permissions are toggled on or off.
5. Tap the toggle to customize the setting of this app.

Sms applications permission Settings:



Chrome applications permission settings example:



Conclusion: Thus, we have studied security permissions for applications in android phones.

FAQs:

1. How do I stop an app from accessing my contacts?
2. Can apps steal your photos?
3. What are app protection policies?

Assignment No. 5

Aim: Write an android program to encrypt and decrypt text file. Use bouncy castle library API or java cryptography API.

Theory:

There are different cryptographic providers other than native java libraries used for cryptography. BouncyCastle is a third party Java library that complements the default Java Cryptographic Extension (JCE). Bouncy castle libraries are preferred to use in mobile phones since they are light weight. Here, steps are listed along with code to show how to use BouncyCastle to perform cryptographic operations, such as encryption, decryption.

1. Maven Configuration (Eclipse or Netbeans)

Note: You may also try jdk instead of IDE

Before we start working with the library, we need to add the required dependencies to our *pom.xml* file:

```
<dependency>
<groupId>org.bouncycastle</groupId>
<artifactId>bcpkix-jdk15on</artifactId>
<version>1.58</version>
</dependency>
```

2. Setup Unlimited Strength Jurisdiction Policy Files

The standard Java installation is limited in terms of strength for cryptographic functions, this is due to policies prohibiting the use of a key with a size that exceeds certain values e.g. 128 for AES. To overcome this limitation, we need to configure the unlimited strength jurisdiction policy files. In order to do that, we first need to download the package by following this oracle link. This is optional step as already these files are available in installed jdk. Afterwards, we need to extract the zipped file into a directory of our choice – which contains two jar files:

- local_policy.jar
- US_export_policy.jar

Finally, we need to look for the {JAVA_HOME}/lib/security folder and replace the existing policy files with the ones that we've extracted here.

Note that **in Java 9, we no longer need to download the policy files package,** setting the crypto.policy property to unlimited is enough:

```
Security.setProperty("crypto.policy", "unlimited");
```

Once done, we need to check that the configuration is working correctly:

```
int maxKeySize = javax.crypto.Cipher.getMaxAllowedKeyLength("AES");
```

2. System.out.println("Max Key Size for AES : " + maxKeySize);

As a result:

1. Max Key Size for AES : 2147483647

Based on the maximum key size returned by the getMaxAllowedKeyLength() method, we can safely say that the unlimited strength policy files have been installed correctly.

If the returned value is equal to 128, we need to make sure that we've installed the files into the JVM where we're running the code.

3. Cryptographic Operations

3.1. Preparing Certificate and Private Key

Before we jump into the implementation of cryptographic functions, we first need to create a certificate and a private key.

For test purposes, we can use these resources:

- Baeldung.cer
- Baeldung.p12 (password = “password”)

Baeldung.cer is a digital certificate that uses the international X.509 public key infrastructure standard, while the Baeldung.p12 is a password-protected PKCS12 Keystore that contains a private key.

Let's see how these can be loaded in Java:

1. Security.addProvider(new BouncyCastleProvider());
2. CertificateFactory certFactory= CertificateFactory .getInstance("X.509", "BC");
3. X509Certificate certificate = (X509Certificate) certFactory .generateCertificate(new FileInputStream("Baeldung.cer"));
4. char[ ] keystorePassword = "password".toCharArray();
5. char[ ] keyPassword = "password".toCharArray();
6. KeyStore keystore = KeyStore.getInstance("PKCS12");
7. keystore.load(new FileInputStream("Baeldung.p12"), keystorePassword);
8. PrivateKey key = (PrivateKey) keystore.getKey("baeldung", keyPassword);

First, we've added the BouncyCastleProvider as a security provider dynamically using the addProvider() method. This can also be done statically by editing the {JAVA_HOME}/jre/lib/security/java.security file, and adding this line:

1. security.provider.N = org.bouncycastle.jce.provider.BouncyCastleProvider

Once the provider is properly installed, we've created a CertificateFactory object using the getInstance() method.

The getInstance() method takes two arguments; the certificate type “X.509”, and the security provider “BC”.

The certFactory instance is subsequently used to generate an X509Certificate object, via the generateCertificate() method.
In the same way, we've created a PKCS12 Keystore object, on which the load() method is called.
The getKey() method returns the private key associated with a given alias.
Note that a PKCS12 Keystore contains a set of private keys, each private key can have a specific password, that's why we need a global password to open the Keystore, and a specific one to retrieve the private key.
The Certificate and the private key pair are mainly used in asymmetric cryptographic operations:

- Encryption
- Decryption

3.2 Encryption and Decryption

In asymmetric encryption cryptography, each communication requires a public certificate and a private key. The recipient is bound to a certificate that is publicly shared between all senders.
Simply put, the sender needs the recipient's certificate to encrypt a message, while the recipient needs the associated private key to be able to decrypt it.
Let's have a look at how to implement an encryptData( ) function, using an encryption certificate:

```
1. public static byte[ ] encryptData(byte[ ] data,
2. X509Certificate encryptionCertificate)
3. throws CertificateEncodingException, CMSException, IOException {
4. byte[ ] encryptedData = null;
5. if (null != data && null != encryptionCertificate) {
6. CMSEnvelopedDataGenerator cmsEnvelopedDataGenerator = new
   CMSEnvelopedDataGenerator();
7. JceKeyTransRecipientInfoGenerator jceKey
   = new JceKeyTransRecipientInfoGenerator(encryptionCertificate);
8. cmsEnvelopedDataGenerator.addRecipientInfoGenerator(transKeyGen);
9. CMSTypedData msg = new CMSProcessableByteArray(data);
10. OutputEncryptor encryptor
    = new JceCMSContentEncryptorBuilder(CMSAlgorithm.AES128_CBC)
11. .setProvider("BC").build();
12. CMSEnvelopedData cmsEnvelopedData = cmsEnvelopedDataGenerator
13. .generate(msg,encryptor);
14. encryptedData = cmsEnvelopedData.getEncoded();
15. }
16. return encryptedData;
17. }
```

We've created a JceKeyTransRecipientInfoGenerator object using the recipient's certificate. Then, we've created a new CMSEnvelopedDataGenerator object and added the recipient information generator into it. After that, we've used the JceCMSContentEncryptorBuilder class to create an OutputEncrytor object, using the AES CBC algorithm.

The encryptor is used later to generate a CMSEnvelopedData object that encapsulates the encrypted message.

Finally, the encoded representation of the envelope is returned as a byte array.

Now, let's see what the implementation of the decryptData() method looks like:

```
1. public static byte[ ] decryptData(
2. byte[ ] encryptedData,
3. PrivateKey decryptionKey)
4. throws CMSException {
5. byte[ ] decryptedData = null;
6. if (null != encryptedData && null != decryptionKey) {
7. CMSEnvelopedData envelopedData = new CMSEnvelopedData(encryptedData);
8. Collection<RecipientInformation> recipients
   = envelopedData.getRecipientInfos().getRecipients();
9. KeyTransRecipientInformation recipientInfo
   = (KeyTransRecipientInformation) recipients.iterator().next();
10. JceKeyTransRecipient recipient
    = new JceKeyTransEnvelopedRecipient(decryptionKey);
11. return recipientInfo.getContent(recipient);
12. }
13. return decryptedData;
14. }
```

First, we've initialized a CMSEnvelopedData object using the encrypted data byte array, and then we've retrieved all the intended recipients of the message using the getRecipients() method.

Once done, we've created a new JceKeyTransRecipient object associated with the recipient's private key.

The recipientInfo instance contains the decrypted/encapsulated message, but we can't retrieve it unless we have the corresponding recipient's key.

Finally, given the recipient key as an argument, the getContent() method returns the raw byte array extracted from the EnvelopedData this recipient is associated with.

Driver code in main()

Let's write a simple test to make sure everything works exactly as it should:

```
1. String secretMessage = "My password is 123456Seven";
2. System.out.println("Original Message : " + secretMessage);
```

3. byte[ ] stringToEncrypt = secretMessage.getBytes();
4. byte[ ] encryptedData = encryptData(stringToEncrypt, certificate);
5. System.out.println("Encrypted Message : " + new String(encryptedData));
6. byte[ ] rawData = decryptData(encryptedData, privateKey);
7. String decryptedMessage = new String(rawData);
8. System.out.println("Decrypted Message : " + decryptedMessage);

As a result during running the program.

1. Original Message : My password is 123456Seven
2. Encrypted Message : 0�*�H��...
3. Decrypted Message : My password is 123456Seven

Conclusion: Thus, we have studied and implemented encryption using bouncy castle API.

FAQs:

1. What is bouncy castle used for?
2. What do you mean by message digest? List different algorithms..

Reference: https://www.baeldung.com/java-bouncy-castle

Assignment No. 9

Aim: Install, configure and demonstrate any one Wi-Fi traffic analyzer using sniffing tools such as Wireshark, AirCrack, AirSnort, etc.

Theory:

Wireless networking, such as Wi-Fi - is a technique of receiving broadband internet without wires. Wi-Fi allows us to connect a number of computers to the same broadband internet at a time. People have a laptop or any wifi supported gadget can be used within the coverage zone of wifi network. Extra phone lines or cables is not required if Wi-Fi is installed.

It is easy to create Wi-Fi network with several computers and a wireless access point. Router is also needed to build a wireless network. This is a single unit that contains: a port to connect with cable or DSL modem, a router, An Ethernet hub, a firewall, a wireless access point. A wireless router allows us to use wireless signals or Ethernet cables to connect computers and mobile devices to one another, to a printer and to the Internet. Most routers provide coverage for about 100 feet (30.5 meters) in all directions, although walls and doors can block the signal. If the home or the area where network will be set up is very large, it is important to buy reasonable priced range extenders or repeaters to increase working router's range.

With the rapid growth of Wi-Fi arises the number of hackers. The major concern of users at Wi-Fi hotspots is security. These types of wireless networks are primarily unsecure. This is because encryption methods such as WEP and WPA, which are usually used to protect private wireless networks, aren't implemented due to the complexities of supporting users. Moreover, using WEP or WPA means you'll have to advertise the "private" encryption key(s). So now there is no point in using encryption since the eavesdroppers will have the key(s) to quickly decode the Wi-Fi hotspot traffic. Wi-Fi hotspots are usually susceptible to hackers. Once a free Wi-Fi hotspot is launched, suddenly a couple of more "free" Wi-Fi networks emerges that promise to allow you to access the Internet for free.

To obtain awareness against different attacks, there are few open source tools available to monitor and secure the Wi-Fi network. However, end-to-end security still cannot be assumed; just enabling Wi-Fi encryption will not secure the applications running on the wireless network. Wi-Fi technologies, products, and attacks will continue to sprout. Security admin still need to keep updated on new threats, assess their business risk, and take appropriate action. Some Wi-Fi open source tools available for the safety of Wi-Fi zone are mentioned below-

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named

Ethereal, in May 2006 the project was renamed Wireshark due to trademark issues. Wireshark is cross-platform, using the GTK+ widget toolkit to implement its user interface, and using pcap to capture packets; it runs on various Unix-like operating systems including Linux, Mac OS X, BSD, and Solaris, and on

Microsoft Windows. There is also a terminal-based (nonGUI) version for Linux called TShark. Wireshark, and the other programs distributed with it such as TShark, are free software, released under the terms of the GNU General Public License.

Kismet Linux fans know that Kismet is a Wi-Fi Swiss Army knife--it discovers APs and clients, captures Wi-Fi packets from local NICa or remote drones, and can generate alerts for fingerprinted recon activities. Kismet is a versatile client/server tool that can be paired with any RFMON-capable adapter--even on OS X or Cygwin. Using Kismet, one can identify discovered APs and clients which will help spot policy violations like mis-configured APs or misbehaving clients.

Ettercap is a suite for man in the middle attacks on LAN. It features sniffing of live connections, content filtering on the fly and many other interesting tricks. It supports active and passive dissection of many protocols (even ciphered ones) and includes many features for network and host analysis. These entire features are integrated with a easy-touse and pleasurefulncurses/gtk interfaces.

PRTG is a open source network monitor where PRTG stands for Paessler Router Traffic Grapher. It is a network monitoring software from Paessler AG.PRTG runs on Windows and monitors network availability and network usage using SNMP, Packet Sniffing, WMI, IP SLAs and Netflow and various other protocols.

Nagios is the definitive open source network monitoring solution. It can be used from simply checking to see if a network host is still up, all the way up to monitoring specific services on remote hosts, and even to trigger corrective action if a problem is detected. And tell you about all that by mail, phone, fax, pager, sirenes and flashing lights, and possibly also by carrier pigeon.

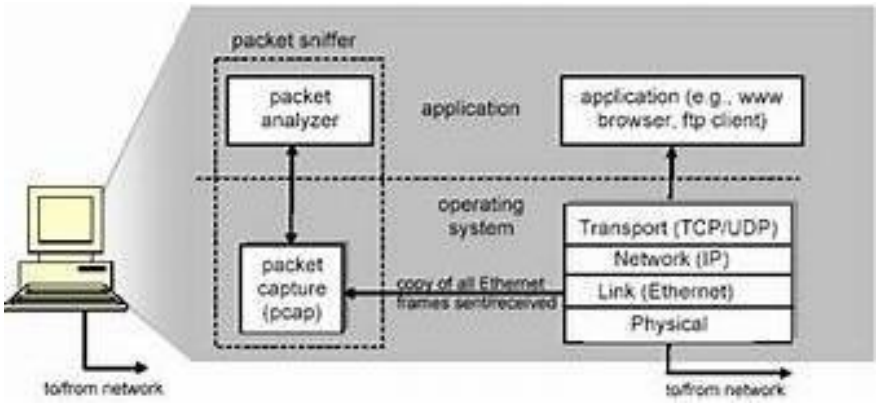
The easy-creds script is a bash script that leverages ettercap and other tools to obtain credentials during penetration testing. Menu driven, it attack with basic ARP spoofing, one-way ARP spoofing and DHCP spoofing and the setup of a Fake AP. Moreover, it has an SSLStrip log file parser that leverages a definition file to give one the compromised credentials and the site they have come from.

Aircrack-ng is an 802.11 WEP and WPA-PSK keys cracking program that can recover keys once enough data packets have been captured. It implements the standard FMS attack along with some optimizations like KoreK attacks, as well as the all-new PTW attack, thus attack compared to other WEP cracking tools. In fact, Aircrack-ng is a set of tools for auditing wireless networks.

Mitmjws is a basic script to automate man-in-the-middle attacks. The script calls airbase, ettercap, sslstripper and driftnet, requires aircrack-ng with experimental software. So, before test this program, you need install all dependence tools and libraries. This project's source code is released under GNU General Public License v3 (GPLv3).

The basic tool for observing the messages exchanged between executing protocol entities is called a packet sniffer. Packet sniffer captures (“sniffs”) messages being received from computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the

packet sniffer. Instead, a packet sniffer receives a copy of packets that are sent by application and protocols executing on machine. The following figure shows the structure of a packet sniffer. At the right of figure are the protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on your computer. The packet sniffer, shown within the dashed rectangle in figure is an addition to the usual software in your computer, and consists of two parts. The packet capture library receives a copy of every link-layer frame that is sent from or received by your computer. The messages exchanged by higher layer protocols such as HTTP, FTP, TCP, UDP, DNS, or IP all are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In Figure 2.a, the assumed physical media is an Ethernet, and so all upper layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you all messages sent/received from/by all protocols and applications executing in your computer.

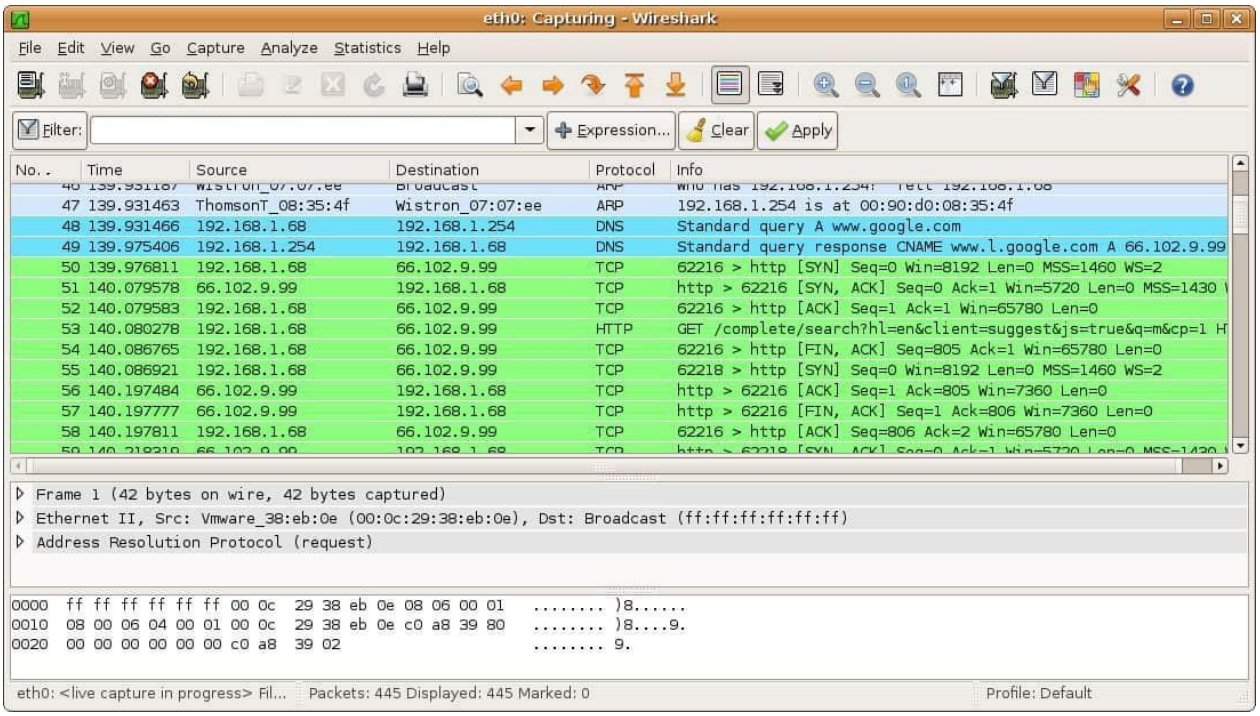


The second component of a packet sniffer is the packet analyzer, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must “understand” the structure of all messages exchanged by protocols.

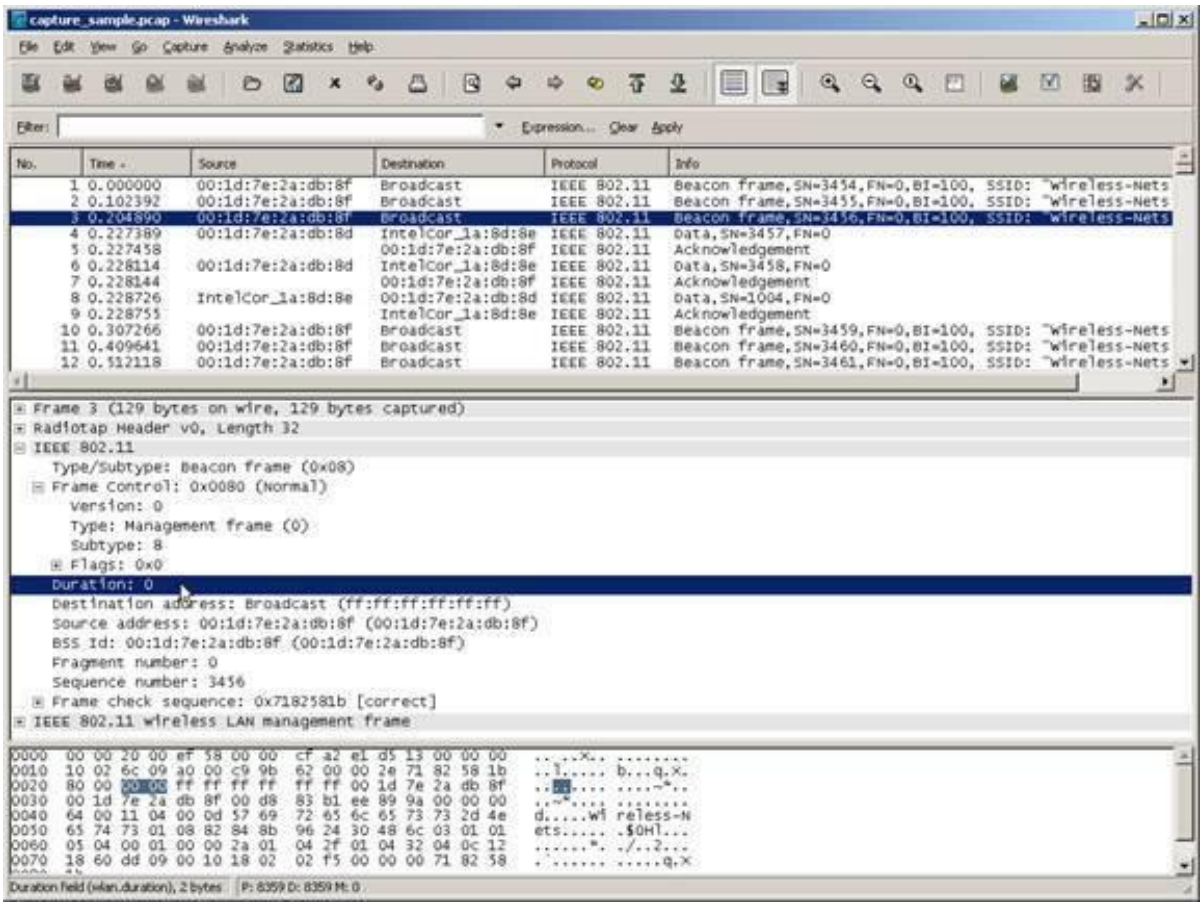
There are many packet capturing and analyzing tools available in market but there is a tool wireshark that leads the rest. Wireshark is by far the best GUI based open source packet analyzer.
Wireshark is a tool that can capture network packets (both incoming and outgoing) and present them in a GUI providing detailed information about each packet captured. This tool is extremely helpful for network administrators to know details like which all computers are trying to communicate with a machine. Also, while debugging any connectivity related issue, the details provided by wireshark capture is very useful.
This tool is also used by protocol implementers to test whether particular protocol packets are being correctively formed or not. Wireshark is also used in case of debugging by software developers in case they want to know how a packet arrived on wire and whether it was changed by an application or not?

Using Wireshark is not rocket science. A couple of configuration steps can help Wireshark to capture packets. Here are the few steps to get your Wireshark up and capturing in a basic mode: Select the network interface on which we want to capture the packets. This can be done through Capture->Interfaces or can select the interface from the list as shown below. Please start this tool with administrator privileges otherwise you will not see any interface in the list.

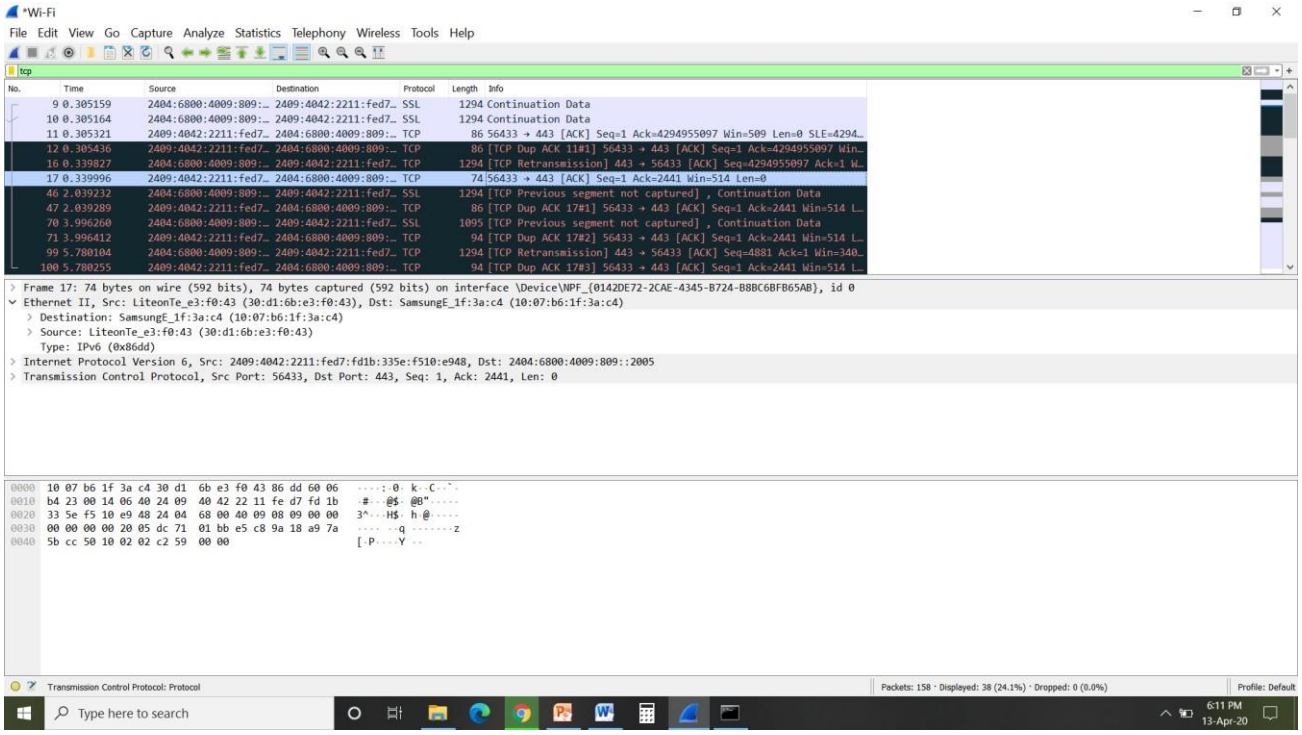
Once the interface is selected, Wireshark will start capturing all packets arriving and leaving the selected network interface.



If we click on any packet, we will see detailed information about that packet in the lower half of the Wireshark GUI.



We can filter the packets based on various filters that are available with Wireshak. For example, if we want to highlight only TCP packets, just type tcp in the filter box and hit enter.



To stop the capture, need to hit the stop button present in GUI. Always stop Wireshark once we are done with capture otherwise it will keep capturing packets and will consume significant amount of system memory that may slow down the system.
We can save the capture for future reference using File->save
So that, by following those steps, we can use Wireshark easily.

A Protocol Analyzer: It is mostly a tool for seeing the bits and bytes flowing from end to end to a network in human understandable form. Without it, understanding a network communication exchange would be almost impossible. Network protocol is broken down into 7-layers. The part that WireShark deals with is layer 2 up to 7. Most well-known protocols can be decoded by WireShark.

Learn Network Protocols: One of the most obvious applications of WireShark is the ability to capture network traffic and look at it from the perspective of learning. For instance, if we are learning how the TCP protocol works, capture traffic from our own computer when we visit a web site. In the captured trace file, you will see every detail of the network communication exchange including the details of the well-known 3-way connection handshake.

Solve Network Problems: While "black box" method to network troubleshooting doesn't does not work, it is time to use WireShark. For example, at work, we had an issue where a computer was powerless to connect to a specific address on the Internet. We patterned the setup again. The Internet configuration 29 was OK because people can get to it from outside of our network, but from within out network, they could not reach this particular site. Normal troubleshooting method didn't change it. Using WireShark the network traffic being exchanged by our computer and the network could be captured. The capture revealed that our computer was getting a tcp check thus the connection would not go through. As it turns out, out company web filter was sending a TCP RESET to block us from reaching that particular site! Without WireShark, there was no way we could have figured this out. Solving network issues is probably the best use of WireShark.

Wireshark Misconceptions: There is a combine misconception about wireshark to recognize what wireshark is not.

1. Wireshark is not a packet generator or packet dropper. It is only capture packets and analyze them. Through this, it is easy to configure some filters for wireshark to display only the packets but nothing can be done with actual packet.
2. Wireshark will never advise if any suspicious packets or mischievous connections. Thus, it cannot use as an alarm or notification for packets.

Conclusion: Thus, we have studied, configured wireshark for Wi-Fi traffic.

FAQs:

1. List the different open source tool to capture packet. Also, write its features.
2. Which mode NIC uses for Ethereal/packet sniffing?
3. Which wireshark filter can be used to monitor outgoing packets from a specific system on the network?

THANK YOU