4th

step 1 → open wireshark, click on "any"

step 2 → visit any website from browser, wait till it loads (wait for some time)

minimize browser

step 3 → goto wireshark (wait few secs) ~~stop the capturing~~ Assuming that the packets are been captured
stop the capturing.

// Wireshark : is a protocal analyser and packet analyser tool
packet consists of two components headers and payload
payload → user data
header consists of IP address of src and dest

step 4 → In the "Apply display filter" input bar, Type "dns" and select filter //→//

step 5 → Select the first Datagram request which is indicated with an incoming arrow "→" double click on it and right click on it and mark the packet
// Analyse the packet headers details in header area which is below the packet area //

// client port will always be greater than 1023 and server port will be less than 1023 //

RR → Resource records

Type A: Directly host name will be set

Type CNAME: Aliasing the to host name will be set

Step 6:- Unmark the previously marked packet and mark the first response packet & in color with outgoing arrows // ← //
and follow the same analysis as previous step

Step 7:- Again mark the first request packet and go to "Statics" menu in menu bar and select "flow Graph" in the dropdown

You will get the "flow graph" window and select the "limit to display" option and analyse the flow chart.

Save the analysis file

// To capture the local traffic

open the terminal first run the updated program
and follow all the same steps from 3.

To capture

Important

// 3 way handshake 5th Exp

same as 4th till minimising the browser

steps : Stop capturing

In the Apply filter bar type the following command

tcp.flags.syn==1 and click on "→"

// meaning of this command is it gets all the packets whose syn flag is set to 1, which means that that is tends connection request //

Step 4 : Mark the first request packet, indicated by incoming arrow "→"

and analyse the Header section.

Analyse the TCP

// https will be running on standard port 443

// window = 65536, ie total number of maximum packets that one can send.

Step 5: Find for first ~~Acknowledge~~ packet, indicated by outgoing arrow. "←"

// for that type the command

tcp.flags.syn==1 && tcp.flags.ack==1

in Apply filter bar and select Apply "→" //

// In TCP header,

you will find all flags

// Acknowledge number will be same as that of sequence number given to first packet //

Step 6 : for Both the packets, mark them one by one and ~~click~~ analyse the flowgraph from statistics window.

request → (tcp.flags.syn==1)
Ack (from server to client) (tcp.flags.syn==1 && tcp.flags.ack==1)
Ack (from client to server (tcp.flags.syn==0 && tcp.flags.ack==1)
(starts data)

Step 7: In the flow graphs of each of packet set the "flow type" as "TCP flows" and select the limit to display filter.

// for first packet i.e request packet let the typed code, ie tcp.flags.syn==1 be applied and select flow graph and analyse it

// for first ack packet let the typed code, ie tcp.flags.syn==1 && tcp.flags.syn==1 be applied and select flowgraph from statistics and analyse it

Step 8: Type "tcp.flags.syn==0 && tcp.flags.ack==1" in Apply filter bar and Apply
// which means that client has received the data and sends the acknowledgement //
finds for data packet and mark it and analyse the headers area //

Step 9: To show the connection termination type "tcp.flags.fin==1" in Apply filter and click Apply →"

Step 10: find the last packet or which is the "connection termination packet" and mark it and analyse it i.e headers and flow graph.

// There are two terminations
// last packet ie termination from client to server.

Expt 6

First place all the program files that you want to use into the "scratch" directory which is in ns3-allinone-3.28 folder → ns3-3.28

Step1 : Navigate to ns-3.28 and run the following command to run the program file ie "./waf --run scratch/First
(name of program file)

Step2 : You will obtain the data and analysis

Step2 : Open the program file within an editor (gedit first.cc) and add the following code to create animation which is to be place below the line
" clientApps.stop(seconds(10.0)) "

first add header file // " #include "ns3/netanim-module.h"

code to be added for animation :-

"
AnimationInterface anim ("first.xml");
anim.SetConstantPosition(nodes.Get(0), 1.0, 2.0);
anim.SetConstantPosition(nodes.Get(1), 4.0, 6.0);
AsciiTraceHelper ascii;
save the file

Step3 : Navigate to ns-3.28 and compile and run the code file using
../waf --run scratch/First.

Step 4 :- To run the Animation simulation file ie First.xml open the netanim simulator

which is in "all" folder

navigate which is located in navigate to this folder and run the following g cmd to run Animation.

./NetAnim.

Step 5: In Nt Animation window, open your xml file ie "First.xml" which is within the ns3.28 folder, select Nt file and open

Analyse the nodes sender and receiver present in "run Nt animation"

Analyse the "stats" and "Packets"

↓ Structure of nodes

↓ movement of packets and delay in between.

Step 6: Generate the trace file:

To generate the trace file add the following code below the animation code added previously

open the code file ie gedit first.cc which is in scratch

code to be included:

// AsciiTraceHelper ascii;

pointToPoint.EnableAsciiAll(ascii.CreateFileStream("first.tr"));

pointToPoint.EnablePcapAll("first"); //

save the file

and run command // cd..

"./waf --run scratch/first" to compile and run

step 7: To open the trace file with "first.tr" which is in ns-3.28 using command

gedit first.tr

Analyse all the data:

// enque : Adding packets one by one to buffer
// dequeue : removing " " " from "
// Timings of sender and receiver are different

step 8: You can make the necessary changes to the parameters within the code.

open the program file:
gedit first.cc

- You can increase the number of nodes (minimum node is 2)
- You can change the delay ie eg:("2ms")
- you can change the server and client start and stop seconds // but server should always start first
- You can change "Max packets
"Interval
"Packetsize".

You can change the position.

Exp 7

Step 1 :- Copy the third.cc from example -> Tutorial folder and place it in ns-3.28 / scratch and open the Third.cc file

Step 2 :- Navigate to ns-3.28 and run ./waf --run scratch/third.

Step 3 :- open Third.cc file.

and add the appropriate header code and other code for animation.

#include "ns3/netanim-module.h"

and add the following code below "Simulator::Stop (Seconds(10.0))"

Code to be included :-

AnimationInterface anim ("7.xml");

save the code and run ./waf --run scratch/third and analyse the

Step 4 :- navigate to netanim-3.108 and run the command

./NetAnim

Step 5 :- load the XML file "7.xml" and run the simulation and analyse the animation and stats and packets and stats and packets

come out of ns3 dir (cd ..)

Step 6 :- Navigate to ns-3.28 and go to scratch
Open the third.cc file

make these changes into code for generating trace file

```
bool tracing = false;
```
↓ to
```
bool ascii tracing = false;
```

add

```
CMD.AddValue ("tracing", "Enable pcap tracing", tracing);
```
↓ to
```
cmd.AddValue ("asciitracing", "Enable ascii tracing",
                                    asciitracing);
```

add the below line of code after
```
AnimationInterface anim ("1.xml");
```

codes to add :

in "if" statement append
ascii to tracing == true
which makes it (asciitracing == true)

and within "if"
replace "pcap" with "ascii" in the
line "PointToPoint.EnableAscii ("third",

and replace "Pcap" with "Ascii" in the
second and third line within if
and remove "true" from third line.

change the present "if" portion ie

```
if (tracing == true)
{
    pointToPoint.EnablePcapAll ("third");
    phy.EnablePcap ("third", apDevices.Get(0));
    csma.EnablePcap ("third", csmaDevices.Get(0),
                                        true);
}
```

to

```
if (asciitracing = true)
{
    AsciiTraceHelper ascii;
    pointToPoint.EnableAsciiAll(ascii.CreateFileStream ("7a.tr"));
    phy.EnableAscii ("7b", apDevices.Get(0));
    csma.EnableAscii ("7c", csmaDevices.Get(1));
}
```

Save the code

Step 7 :- navigate to ns-3.28 and run

```
./waf --run "scratch/third --asciitracing
                                    =true"
```

and analyse.

step 8 :- open all three trace files one by one
i.e "7a" "7b" "7c" and analyse
gedit 7a.tr gedit 7b.tr gedit 7c.tr

7a -> is for point to point network
7b -> is for wireless network
7c -> is for csma network

You can make any necessary changes in parameters

like 6th expt

instead of "ParfwifiManager"

you can use "constantRatManager" or RoadWifiManager

You can make active property to "true"

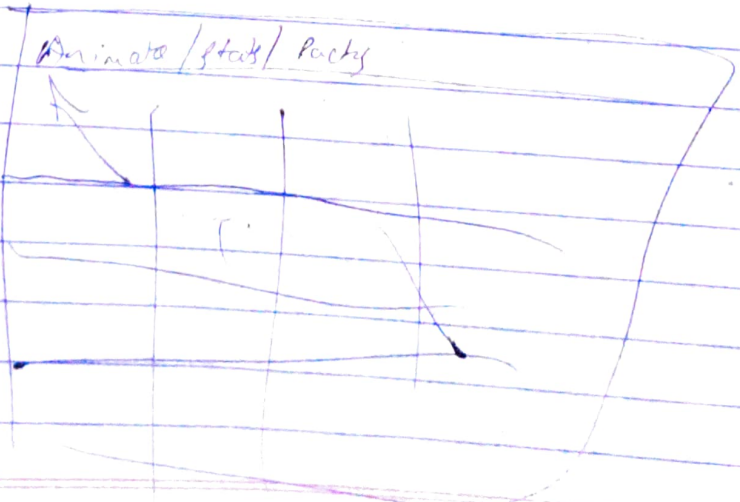
Expt 8

make as much possible changes in the parameters within 7th program ie

eg: ParfwifiManager to constantRatManager

and all changes necessary

- in other parameters



Expt 9

ubuntu cooja or Ubuntu 06

Step 1 :- navigate to cooja ; ie
cd contiki-ng/tools/cooja/
and run the command
ant run
to open the simulator

Step 3 :- In the cooja simulator
goto file → new simulation →
give the name for simulation (Si)
select the radio medium as usual
and leave everything as it is and click
create.

Step 4 → Go to motes in menu bar →
Add motes → create new mote type
→ sky mote

Step 5 → click on Browse and goto examples
folder and → libs → ipv6-hooks →
select ipv6-hooks.c and click open.
and then compile → create

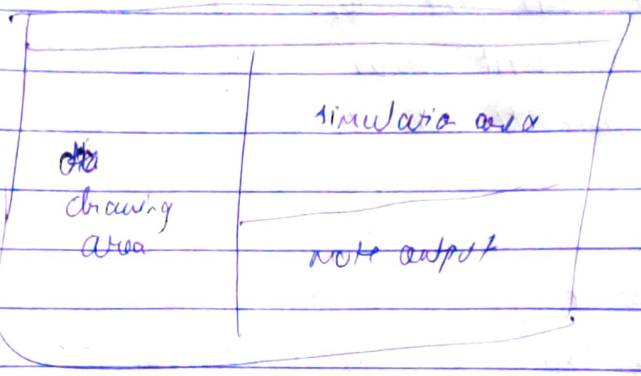
Step 6 → In add motes windows
enter the number of new motes eg (4)
→ add motes

Step 7 :- Place the motes closely

step 8 :- Goto view and select "Address : IP or Link

step 9 :- To simulation and click start.

analyse the "Mote output area"



step 10 :- pause the simulation

and analyse the "Mote output"

Progto.

- CSMA
- Routing Protocol :- RPL low power lossy networks for IOT
- Why IPV6

Expt: 10

Step 1 → /cooja → run ant run

Step 2 → Files → new simulation → name it as S2 → select radio medium as Constant Loss or any other or keep as it is → Create.

~~Step 3 →~~

Step 3 → Motes → add motes → create new mote? type → Sky mote → Browse → Examples folder → RPL-udp folder → select udp-client.c → open → compile → Create

~~enter~~ enter number of motes that should be atleast 2, max 4. → add motes.

place motes close to each other

Step 4 → Motes → add mote → create new motes → Sky mote click Browse → Examples folder → RPL-udp folder → select udp-server.c → open → compile → Create.

enter number of motes for server ie 1 and → add motes.

goto view → select Addresses ~~and radio~~

Step 5 → click start.

Step 6 → pause simulation and analyse