EXPERIMENT NO. 6

AIM: To setup & configuration of Wireless Access Point (AP). Analyse the Wi-Fi communication range in the presence of the access point (AP) and the base station (BS).

THEORY:

In computer networking, a wireless access point (WAP), or more generally just access point (AP), is a networking hardware device that allows other Wi-Fi devices to connect to a wired network. The AP usually connects to a router (via a wired network) as a standalone device, but it can also be an integral component of the router itself. An AP is differentiated from a hotspot, which is the physical location where Wi-Fi access to a WLAN is available.

The network simulator is discrete event packet level simulator. NS is popularly used in the simulation of routing and multicast protocols among others and is heavily used in ad-hoc networking research. NS is a discrete event simulator targeted at networking research.

The network simulator covers a very large number of applications of different kind of protocols of different network types consisting of different network elements and traffic models. Network simulator is a package of tools that simulates behaviour of networks such as creating network topologies, log events that happen under any load, analyse the events, and understand the network. It is popular in academia for its extensibility and plentiful online documentation.

NS-2 is licensed for use under version 2 of the GNU General public license.

In 1996-97, ns version 2 (ns-2) was initiated based on a refactoring by Steve McCanne. Use of Tcl was replaced by MIT's Object Tcl (OTcl), an object-oriented dialect Tcl. The core of ns-2 isalso written in C++, but the C++ simulation objects are linked to shadow objects in OTcl and variables can be linked between both language realms. Simulation scripts are written in the OTcl language, an extension of the Tcl scripting language.

Presently, ns-2 consists of over 300,000 lines of source code, and there is probably a comparable amount of contributed code that is not integrated directly into the main distribution. It runs on GNU/Linux, FreeBSD, Solaris, Mac OS X and Windows versions that support Cygwin.

Steps for generic NS_2 Script:

- Create Simulator object [Turn on tracing]
- Create topology [Setup packet loss, link dynamics]
- Create routing agents
- Create application and/or traffic sources
- Post-processing procedures (i.e. nam)
- Start simulation

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive (at to_node)
+ : enqueue (at queue)              src_addr : node.port (3.0)
- : dequeue (at queue)              dst_addr : node.port (0.0)
d : drop    (at queue)
```

```
r 1.3556 3 2 ack 40 ------- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
```

STEP 1: CREATE SIMULATOR OBJECT

- Create event scheduler

  a) set ns [new Simulator]

STEP 2: TRACING

- Insert immediately after scheduler!
- Trace packets on all links

  set nf [open out.nam w]

  $ns trace-all $nf

  $ns namtrace-all $nf

STEP 3 : CREATE NETWORK

Nodes

1. set n0 [$ns node]

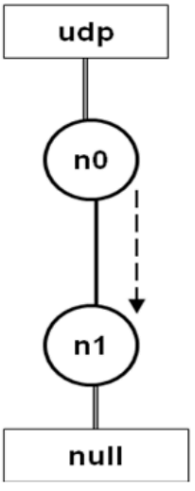
2. set n1 [$ns node]

Links and queuing

1.$ns duplex-link $n0 $n1 1Mb 10ms RED

2.$ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>

3.<queue_type>: DropTail, RED, etc.



STEP 4: NETWORK DYNAMICS

Link failures

1.Hooks in routing module to reflect routing changes

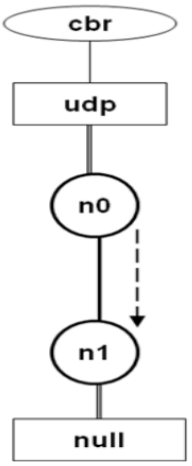
$ns rtmodel-at <time> up|down $n0 $n1

For example: $ns rtmodel-at 1.0 down $n0 $n1

$ns rtmodel-at 2.0 up $n0 $n1

STEP 5: CREATING UDP CONNECTION

- set udp [new Agent/UDP]
- set null [new Agent/Null]
- $ns attach-agent $n0 $udp
- $ns attach-agent $n1 $null
- $ns connect $udp $null



STEP 6: CREATING TRAFFIC (On Top of UDP)

CBR

1.set cbr [new Application/Traffic/CBR]

2.$cbr set packetSize_ 500

3.$cbr set interval_ 0.005

4.$cbr attach-agent $udp

STEP 7:

Add a 'finish' procedure that closes the trace file and starts nam.

```
proc finish {}
{
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}
```

RUN SIMULATION:

Schedule Events

$ns at <time> <event>

<event>: any legitimate ns/tcl commands

$ns at 0.5 "$cbr start"

$ns at 4.5 "$cbr stop“

Call ‘finish’

$ns at 5.0 "finish"

Run the simulation

$ns run

VISUALIZATION TOOLS:

nam-1 (Network AniMator Version 1)

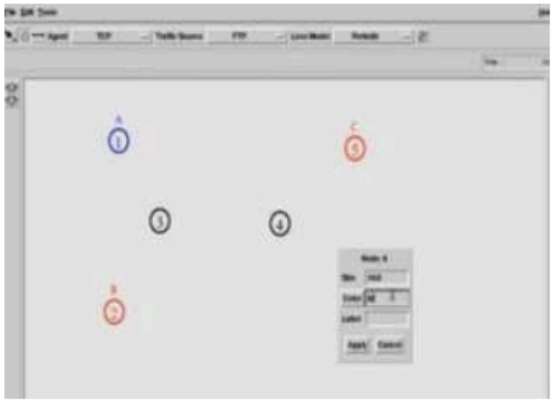
1.Packet-level animation

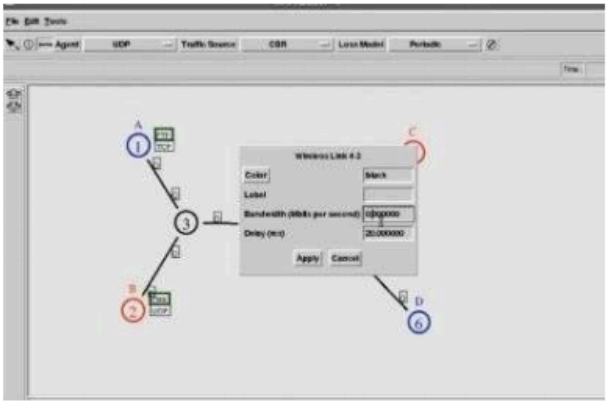
2.Well supported by ns

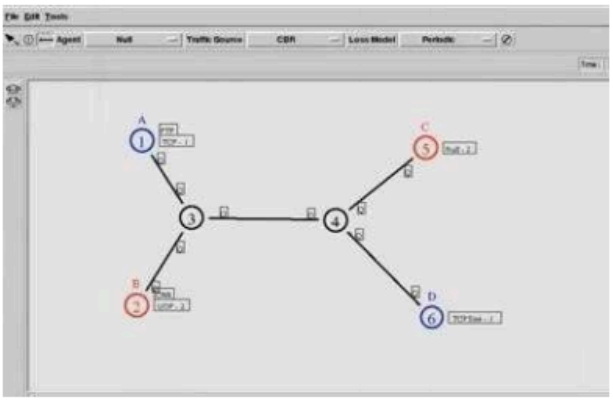
xgraph

1.Simulation result

Results:







Conclusion:

This experiment is to make us familiar with routing protocols used in ad-hoc network and see the performance of DSDV protocols. Thus, we have successfully demonstrated simulation of discrete packet travelling between two nodes in ad-hoc network.