

PRACTICAL-

1

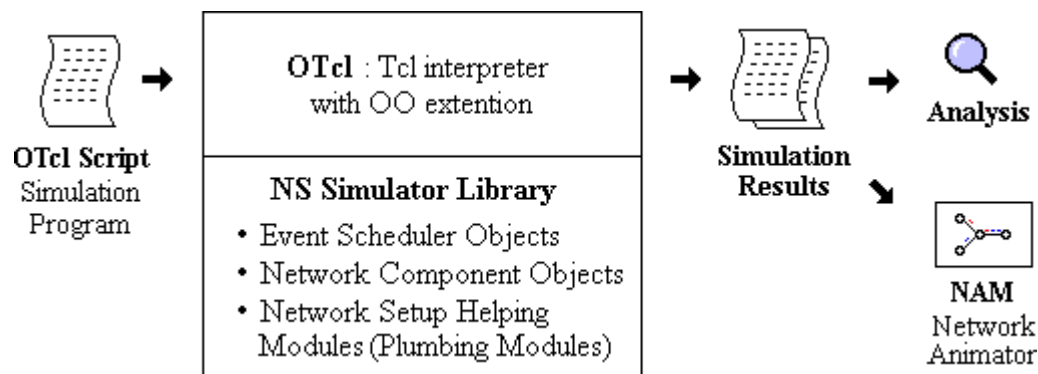
Title: Implementation of Mobile Network using Network Simulator (NS2): Create a Mobile Ad hoc network

Objective: To study Routing in MANET

Pre-Requisite: Basic knowledge of wireless networking

Description:

NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Berkely written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. It implements network protocols such as TCP and UDP, traffic source behavior such as FTP, Telnet, Web, CBR and VBR, router queue management mechanism such as Drop Tail, RED and CBQ, routing algorithms such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations.



Program description:

Each agent keep track of what messages it has seen and only forwards those which it has seen and only forwards those which it hasn't seen before. Each message is of the form "ID:DATA" where ID is some arbitrary message identifier and DATA is the payload. In order to reduce memory usage, the agent store only the messageID.

Steps:

1. Set the following configuration for each node's interface
 - Type of channel -WirelessChannel
 - Type of propagation -TwoRayGround
 - Physical Layer -Wireless
 - Mac Layer - MAC802.11
 - Type of Queue -DropTail/PriQueue
 - LinkLayer -LL
 - Type of Antenna-OmniAntenna
 - Maximum Packet in Queue -50
2. Open Trace file in write mode
3. Open NAM file in write mode.
4. Create a topology containing 6 groups each having 4 nodes. Use Flat Grid topology

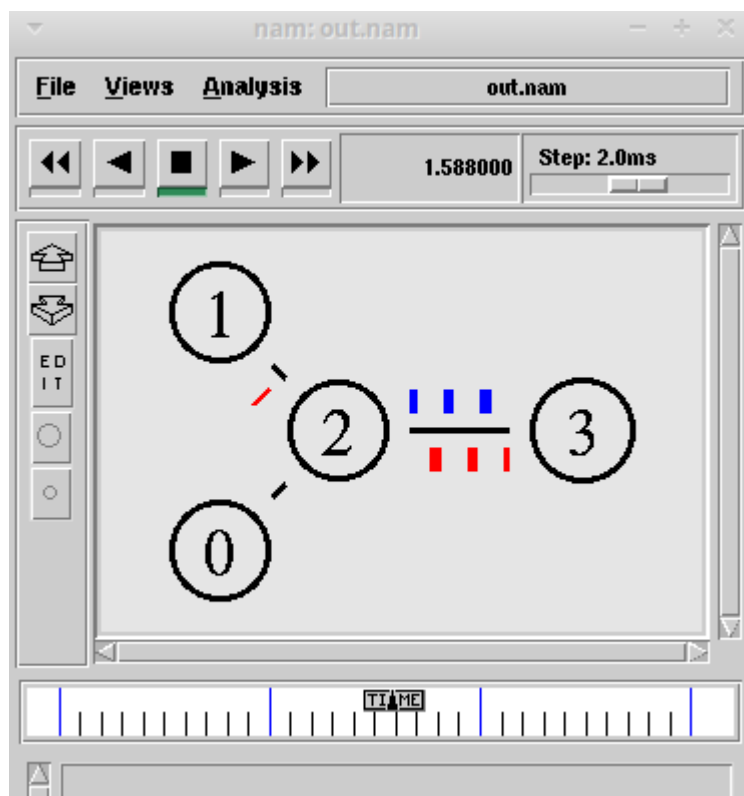
5. Configure each node using the configuration set in step1.
6. Create a simple Message Passing/Flooding agent
7. Create Receive procedure that receives each packet and maintain list of unseen messages
8. Create send procedure that broadcasts message.
9. Create Message Passing/Flooding agent and attach it with every node.
10. Set up some events.
11. Write finish procedure.

Conclusion: Mobile networks using NS2 has been studied and implemented successfully.

```

universe@lenovo18:~/Desktop/ ns simple.tcl
210
0.0037499999999999999
running nam...
universe@lenovo18:~/Desktop/ awk -f exp.awk out.tr
137140 105840 1.00 3.00
universe@lenovo18:~/Desktop/ █

```



```
set ns [new Simulator]
```

```
$ns color 0 blue
```

```
$ns color 1 red
```

```
$ns color 2 white
```

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
```

```
set f [open out.tr w]
$ns trace-all $f
set nf [open out.nam w]
$ns namtrace-all $nf
```

```
$ns duplex-link $n0 $n2 5Mb 2ms DropTail
$ns duplex-link $n1 $n2 5Mb 2ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link $n2 $n4 1.5Mb 10ms DropTail
```

```
$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n2 $n4 orient right-down
```

```
$ns duplex-link-op $n2 $n3 queuePos 0.5
```

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
```

```
set udp1 [new Agent/UDP]
$ns attach-agent $n3 $udp1
$udp1 set class_ 1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
```

```
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
```

```
set null1 [new Agent/Null]
$ns attach-agent $n1 $null1
```

```
set null2 [new Agent/Null]
$ns attach-agent $n4 $null2
```

```
$ns connect $udp0 $null0
$ns connect $udp1 $null1
```

```
$ns at 1.0 "$cbr0 start"
$ns at 1.1 "$cbr1 start"
```

```

set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.2 "$ftp start"

$ns at 1.35 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

puts [$cbr0 set packetSize_]
puts [$cbr0 set interval_]

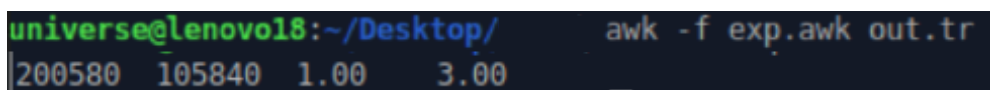
$ns at 3.0 "finish"

proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf

    puts "running nam..."
    exec nam out.nam &
    exit 0
}

$ns run

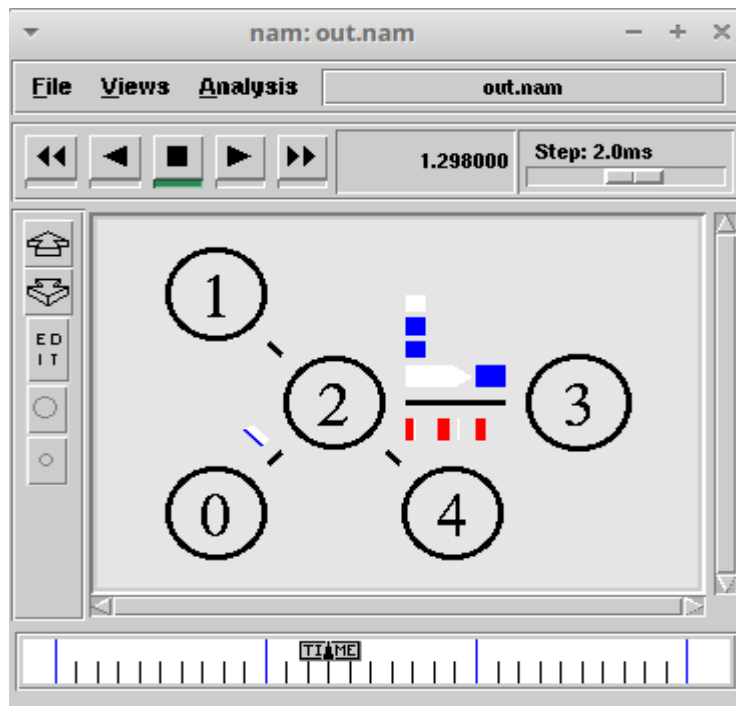
```



```

universe@lenovo18:~/Desktop/      awk -f exp.awk out.tr
200580 105840 1.00 3.00

```



Post Lab Questions:

1. Describe your observations about output.

The TCL script uses ns-2 to simulate a 4-node network with duplex links and TCP traffic. It generates trace and visualization files for analysis using nam. Users can customize parameters to observe network behavior, including packet transmissions and link utilization, in the resulting trace files.

2. Explain the working of DSDV protocol.

Destination Sequenced Distance Vector Routing protocol is a modified version of Bellman Ford Algorithm and is based upon the concepts of Distance Vector Routing.

In Distance Vector Routing(DVR), each node broadcasts a table containing its distance from nodes which are directly connected and based upon this, other nodes broadcasts the updated routing. Those nodes which are unreachable directly are labelled as "infinite".

But, this updation of routing tables keeps on happening and an infinite loop is generated which is commonly known as Count-To-Infinity problem.

To overcome this problem of count to infinity by generating sequence number in the routing table, every time the routing table is updated. The process of DSDV is same as that of Distance Vector Routing but an extra attribute of sequence number is added.

DSDV protocol uses and maintains a single table only, for every node individually. The table contains the following attributes.

Routing Table : It contains the distance of a node from all the neighboring nodes along with the sequence number(SEQ No means the time at which table is updated).

Format :

Node	Destination	Next Hop	Distance	SEQ No
------	-------------	----------	----------	--------

Destination Sequenced Distance Vector Routing : Format

This table is updated on every step and ensures that each node broadcast as well as receives correct information about all the nodes including their distance and sequence number.