PROJECT #3

Modern Networking Concepts

Evaluate MAC random transmission protocol using NS-2

Objective and Learnings:

Ns is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multi-cast protocols over wired and wireless (local and satellite) networks. The use of Network Simulator, in this project was targeted at increasing the comprehension and working of a MAC protocol's implementation for Sensor Networks. Our learning objective is to modify a protocol for achieving a desired simulation for a network topology. Furthermore, to enable us with an understanding of Tcl, Otcl, and C++ and use them to program on the network level.

Key Implementation:

MAC Protocol File(.cc and .h) :

The MAC\Simple protocol provided enables NS-2 to emulate sending a single copy of packet to the destination. In lieu of our project requirement, to send multiple copies of the packet, the mac-simple.cc and mac-simple.h were edited. The mac-simple.cc file contains 3 methods that assist in sending a packed that is received from the upper layer to the destination.

1) waitHandler()
2) sendHandler()
3) send()

We maintain a counter (repeat_count) to help keep a track of number of copies to the destination. The number of repeat transmission per packet copy (repeatTrans) was bound and used to generate 'X' random copies per sending interval. To avoid collision of packets from different nodes, a random jitter was added before sending a particular copy of the packet.

MAC Header File

The variables that were added to the mac Header file were:

1) repeatTrans: The number of copies required to be sent for a single packet.
2) repeat_count: The current count of the packet being sent(of X copies).
3) jitter: The jitter value to be added to prevent collision between packets.
4) induced_jitter: generated sub-time interval in the packet interval to send X copies.

NS-2 Topology File

To simulate the MAC random transmission protocol, we create a Tcl which helps in simulating a network topology which otherwise would have taken considerable resources to experience.

We created a wireless transmission environment. The number of sender nodes we kept at 10, 20, 50, and 100 with one sink node. To increase the chances of transmission, multiple copies of the same packet were sent.

A node will generate a packet every 0.02 seconds and send it to the destination sink node. Thus in every 0.02 seconds a sender node will send X copies of the packet to the sink node.

To simulate MAC random transmission protocol, a topology file was designed for defining the configuration. The NS-2 topology file defines the following parameters

1) Number of nodes
2) Terrain dimension
3) Simulation time
4) Packet size
5) Packet generation interval
6) Number of packet copies

AWK Scripting

I wrote a small script to analyze generated trace file(1.5GB!!!), which narrows down the number of sent, received, dropped packets along with the probability of transmission defined as (No. of (MAC/cbr)packets received / Number of total (MAC/cbr) packets generated).

Process

To run each simulation, the ns nsmac.tcl (topology file name) command is run from within the ns2 root folder (where the tcl file resides).

The network simulation generates a trace file with the name of simulation.tr. This simulation file is then analyzed using awk script. The awk script generated the total number of packets sent during the simulation, no of packets received.

***Note:** Data packets are sent after an initial time where the ARP packets are sent to resolve addresses.

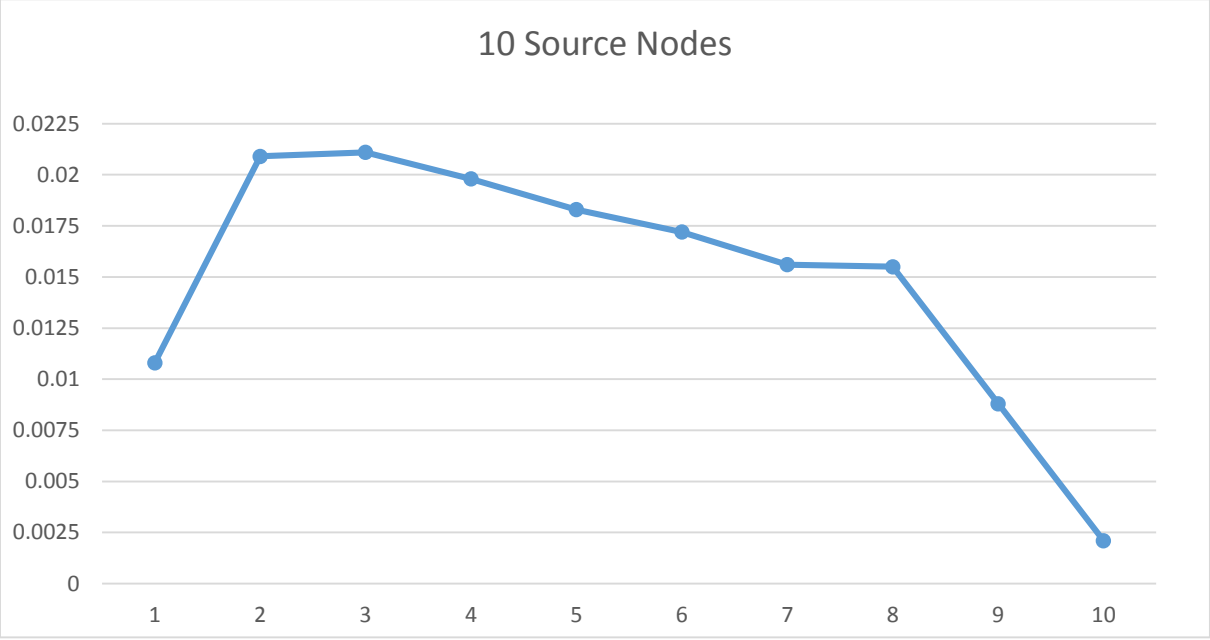
ANALYSIS

Reading *

To analyze the simulation, for a fixed number of nodes, the number of packet copies were varied from 1-10. The probability of number of received packets to the number of sent packets was calculated and plotted on the graph.

1.) Number of Source Nodes: 10 Simulation Time: 100.0 sec

No of Copies Sent per Interval	Probability (recv/send)
1	**0.0108**
2	**0.0209**
3	**0.0211**
4	**0.0198**
5	**0.0183**
6	**0.0172**
7	**0.0156**
8	**0.0155**
9	**0.0088**
10	**0.0021**

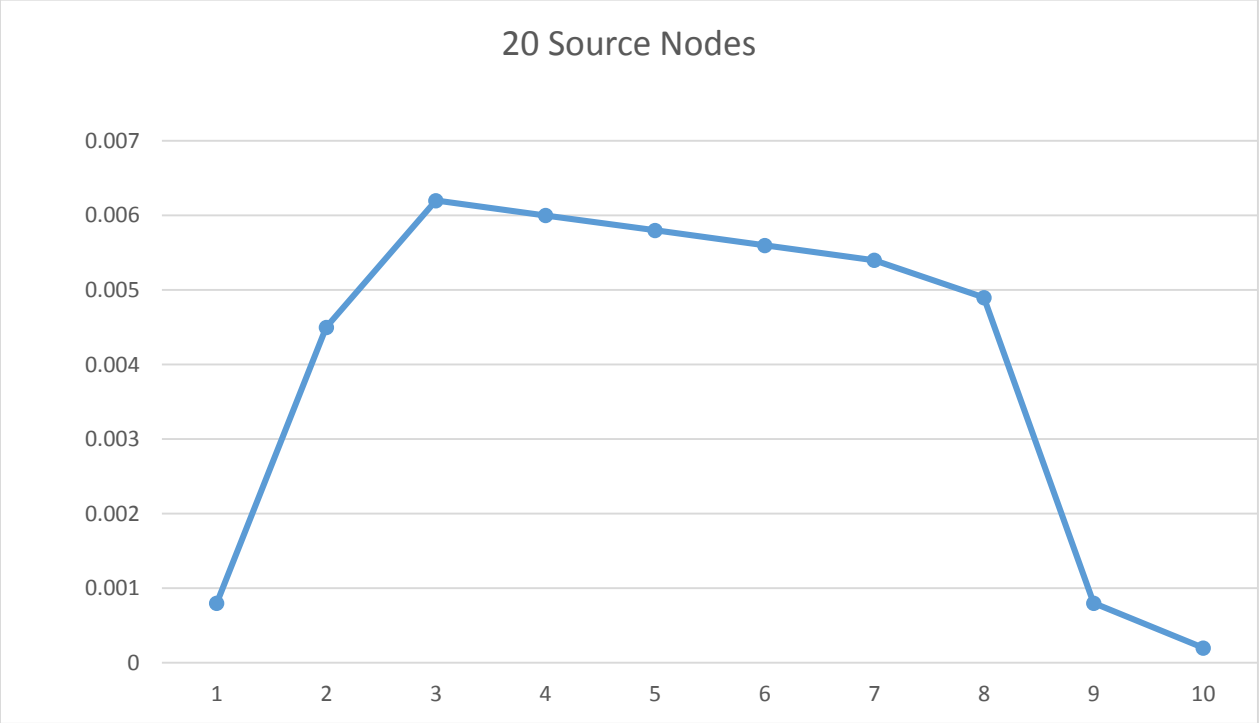


***The output folder has been attached which contains the details of the analysis including number of packets sent, received, dropped and probability calculated.**

2.) Number of Source Nodes: 20 **Simulation Time: 100.0 sec**

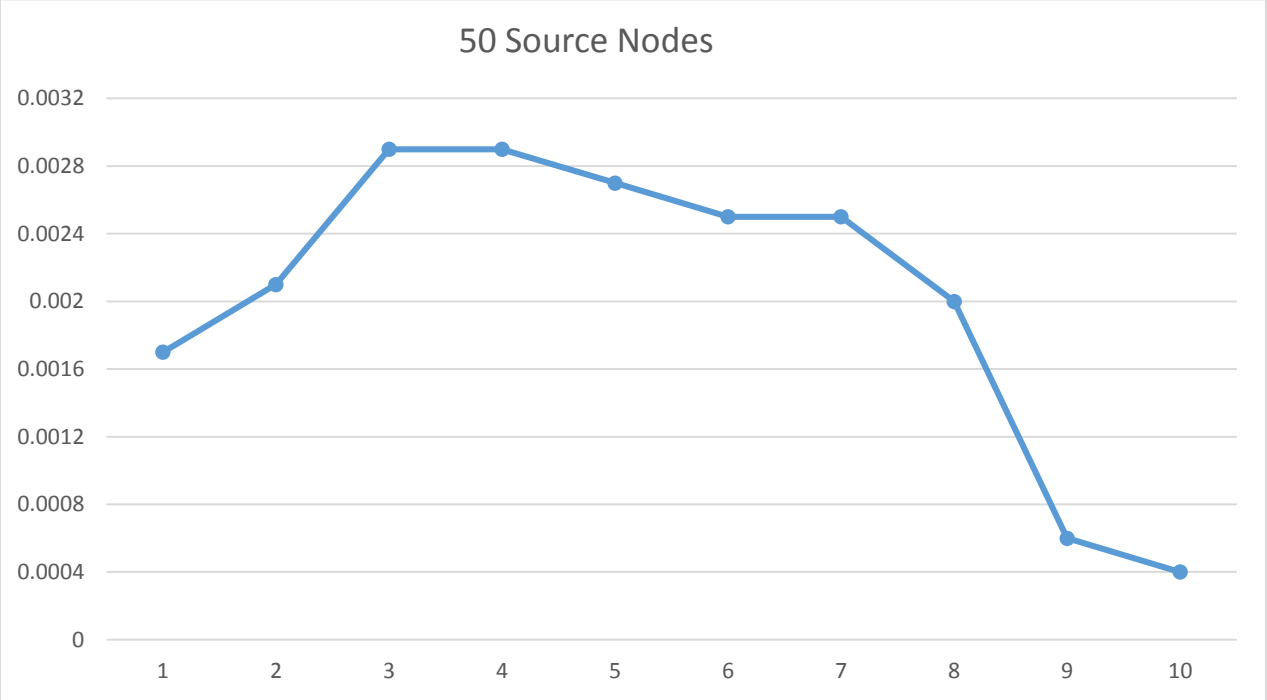
No of Copies Sent per Interval	Probability (recv/send)
1	**0.0008**
2	**0.0045**
3	**0.0062**
4	**0.006**
5	**0.0058**
6	**0.0056**
7	**0.0054**
8	**0.0049**
9	**0.0008**
10	**0.0002**

***Please notice the value units on the y-axis are subject to change**



3.) Number of Source Nodes: 50 Simulation Time: 100.0 sec

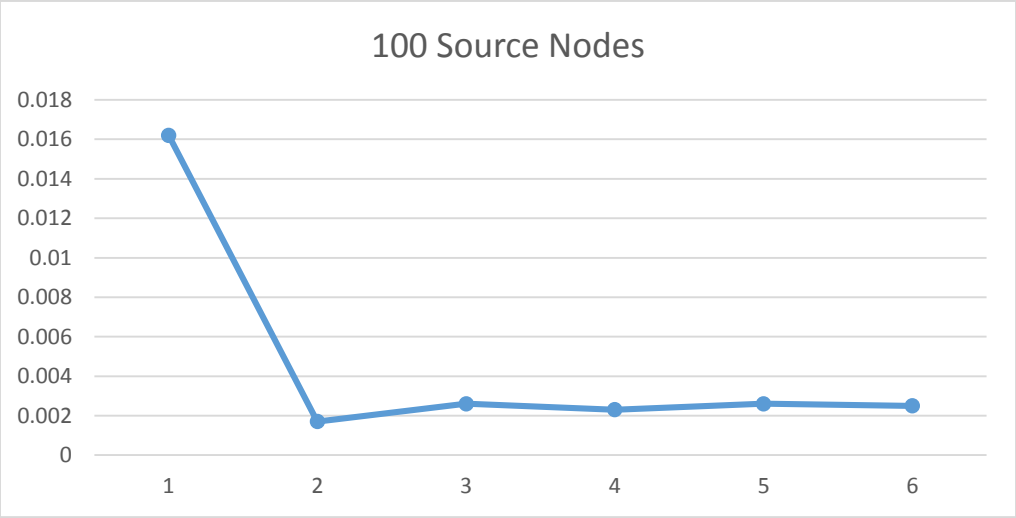
No of Copies Sent per Interval	Probability (recv/send)
1	**0.0017**
2	**0.0021**
3	**0.0029**
4	**0.0029**
5	**0.0027**
6	**0.0025**
7	**0.0025**
8	**0.002**
9	**0.0006**
10	**0.0004**



4.) Number of Source Nodes: 100 **Simulation Time: 100.0 sec**

No of Copies Sent per Interval	Probability (recv/send)
1	0.0162
2	0.0017
3	0.0026
4	0.0023
5	0.0026
6	0.0025
7	System crash
8	System crash
9	System crash
10	System crash

***Behavior uncertain, as system crashed several times so might not be proper**



Observations

1) It has been observed that, during the initial time of the simulation, all nodes send ARP packets to resolve address. This is the reason why the total number of data packets sent are less than the total number expected.
2) The more are the number of nodes in a system, more will be the collisions.
3) The probability initially increases with the increases in the number of copies transmitted. After a certain threshold, the probability reduces again. This is primarily due to the increase in the number of collisions and increased number of drop packets (empirically observed).
4) The protocol behaves like a pure aloha protocol in which any packet ready for transmission is sent immediately(although we are adding a random amount of wait time)
5) The protocol implementation is different from slotted aloha as the protocol does not wait for the start of a time slot for transmission of a packet.

SUMMARY

I learned how to modify a protocol for achieving a desired simulation for a network topology. Use of Tcl helped in creating network simulations which otherwise would have taken considerable resources to experience. I also learned to script awk for analyzing trace files. The analysis showed the MAC behaved like the graph Aloha protocol.