PROJECT REPORT

NS-2 Simulator and MAC-Simple Protocol

Submitted By:

Srikanth Reddy Malipatel - Person#50169097

Anuj Rastogi – Person#50134324

NS-2 Simulator

NS (from **network simulator**) is a name for series of discrete event network simulators, specifically **ns-1**, **ns-2** and **ns-3**. All of them are discrete-event computer network simulators.

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 is also 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 (many forks of ns-2 exist, both maintained and unmaintained).

In our project, the sole purpose behind using NS-2 was to implement and evaluate MAC (Medium Access Protocol) for sensor networks. There is the use of radio transition frequency to transmit packets. Also, we basically built the simulation on 100 nodes which acted as source and there was one sink.

Various Layers of the network and the MAC layer

In practicalities we have the following layers in network which are at the core of any network and which make up the entire network. These are:

1. Application Layer
2. Transport Layer
3. Network Layer
4. Link Layer
5. Physical Layer

So, seeing this hierarchy we can we can conclude the point that MAC Protocol is a link layer protocol and it basically supports wireless network between the end systems. There are several versions of MAC protocol which are present but here we are going to deal with MAC-Simple protocol. The MAC protocol is basically a Wi-Fi-Protocol.

For this project we will be using existing protocols available in ns-2. For application layer we are using CBR - constant bitrate generator, For transport layer we are using UDP and Network layer DSDV and link layer is Phy/WirelessPhy. Using these protocols we will be implementing our own Mac protocol with below features.

The target of the project, hence, is to demonstrate and analyze the nature of the Wi-Fi network which sends packets without sensing for an input signal. Hence, in the project there are 100 nodes with one sink. All of them begin transmissions at t=0 and continue transmission till T seconds. In this period they send many number of packets with random back-off time and at the end of simulation the results are obtained, which project as to how many unique packets are received by the client owing to the duplicate packets sent by any source.

Hence, we have the following assumptions:

1. There are 100 sources
2. There is one sink
3. Each node transmits N number of packets after a random time interval for a few seconds.
4. Therefore in the project we have developed a solution, a .tcl file, through which were able to achieve such a scenario and also we were will be able to accommodate the MAC-Simple protocol in such a requirement. The requirements of the project were:
5. Terrain Dimension: 50m x 50m
6. Simulation Time: 50 s
7. Packet size: 128 bits (including a header)
8. Packet generation interval (T): 0.02 s

Design

We have taken mac-simple protocol and have extended the functionality. mac-simple protocol sends packets after every 0.02 secs but as per our requirement we need send X duplicate packets in this interval.

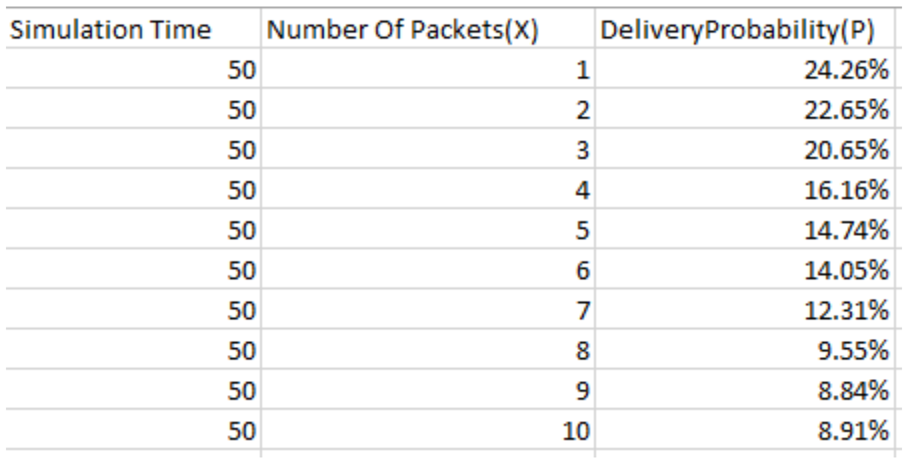
For this we have added two variables to the existing mac-simple protocol. One is for specifying number of duplicate packets and other is for specifying the interval.

Now when we get a packet from above layer to mac layer, we are generating X number of random times in the interval range. After this we sort all the intervals using insertion sort and then at the first time interval we start scheduling the packets in send(), which is the first original packet. When the packets are being sent out by sendHandler() function we start creating a copy packet of this original packet and then schedule X-1 packets.

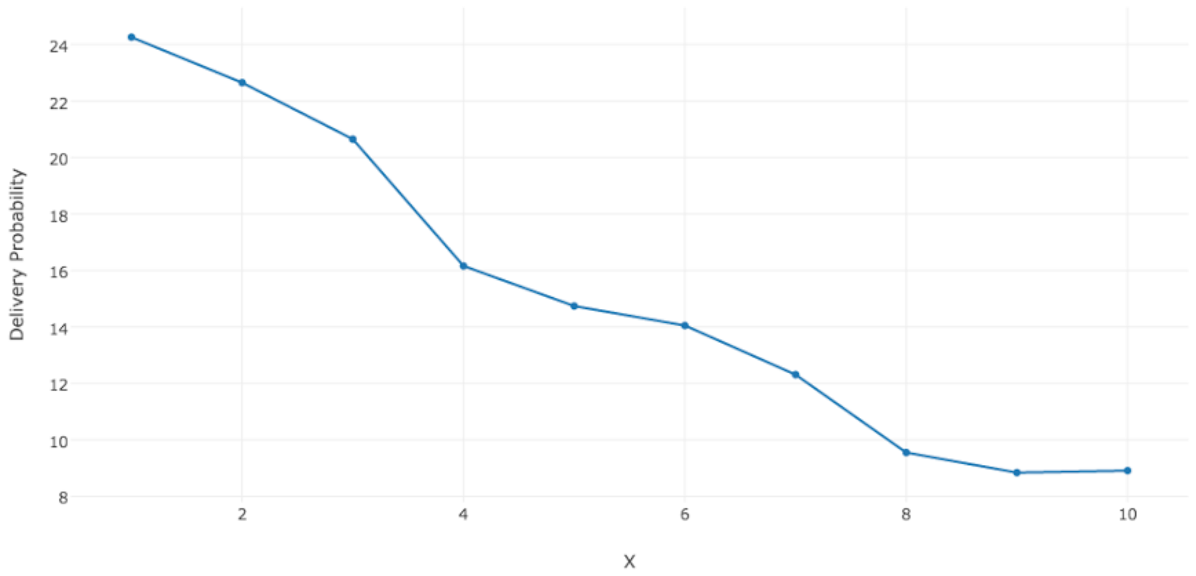
Results of Simulation

Initially we took the packet counts starting from 1 and we proceeded up to 10. The time span chosen was 50 seconds. By increasing the packet counts in successive intervals we saw that the probability of the packets reaching the sink kept on decreasing.

Please find below the results that we found as a part of the project. The table below shows the results when we increased the packet count from 1 to 10.



Simulation Time	Number Of Packets(X)	DeliveryProbability(P)
50	1	24.26%
50	2	22.65%
50	3	20.65%
50	4	16.16%
50	5	14.74%
50	6	14.05%
50	7	12.31%
50	8	9.55%
50	9	8.84%
50	10	8.91%



X = packet count

Y= Probability of delivery in %

Rationale behind this nature

When we increase the duplicity of packets by increasing the number of duplicate packets to assure receiving confirmation, instead of increased probability of receiving packets there is a decrease in the count of the packets. This is so because the overall network congestion becomes intense. This causes a lot of collision and in the absence of sensing device the medium becomes full of signals. Initially while starting the activity we transmitted only one packet, there were less collisions and hence the probability was high.

This brings us to a note that there may be a certain set value of the retransmissions to assure packet receiving at the sink. If we call his threshold then after this value collision will not assure delivery but will instead cause collisions and hence congestion overwhelming the receiver.