NS3 Assignment

Shashwat Verma 4507746

April 30,2016

Contents

1 Introduction

NS3 (Network-Simulator 3) is latest part of software series which is used for computer simulation of discrete event networks. This software is primarily used in research and teaching. NS3 is a free software available under the GNU GPLv2 license for use.
This NS3 assignment simulates the situation of number of changing clients and their random placement under a single access point. It resembles the real-life scenarios of people sitting in public place and access internet by single access-point. I have considered varying nodes and a fixed access point.

2 Project Description

Objective of this project is to observe performance due to change in network parameters. In this assignment following network parameters were changed, number of nodes, size of packets, datarate model, node orientation in given space. By change in these parameters, throughput and delay were calculated for many scenarios.

3 Implementation

Assignment is implemented using `C++` language for NS3. The network topology of this simulation uses Wi-Fi 802.11n standard (5GHz frequency). Using this protocol one access point and N-number of nodes are initialised in the system. Each node have UDP protocol installed for communication between them.Performance of implementation is measured in terms of throughput and delay while varying network parameters like datarate model, size of packets, number of nodes and node orientation in given space.

Based on these criteria outcome of the system can be hypothesized as:

3.1 Hypothesis

- With increase in number of nodes throughput will decrease.

- With increase in number of nodes delay will increase.
- As packet size increases throughput will increase.

3.2 Code construction

In this scenario, one access point is defined which have a fixed and number of nodes and their placing is constantly changing. This scenario is created by installing `ConstantPositionMobilityModel` with fixed position on Access point and `ConstantPositionMobilityModel` with different placement models like `GridPositionAllocator`, `UniformDiscPositionAllocator`, `RandomRectanglePositionAllocator` and `RandomBoxPositionAllocator`.
Flow of code construction is

1. Initialisation of changeable attribute values like number of nodes, datarate model, frequency, packet size, UDP/TCP.
2. Initialisation of user variables like simulation time, mobility scenario, verbose.
3. Installing number of nodes as station nodes.
4. Installing MAC layer on nodes.
5. Installing mobility methods as per node type.
6. Initialising IP address to nodes.
7. Installing UDP server and client protocols.
8. Declaring output files for data analysis.
9. Running the simulation and storing values.

To calculate throughput and delay helper classes are used in code. These classes monitor the packets sent and received from different nodes by attaching headers to it. To achieve this monitoring `FlowMonitorHelper` class is used along with `Ipv4FlowClassifier` and `FlowMonitor class`.

After receiving data from these variable throughput and delay is calculated by these formulas:

```
throughput=rxBytes*8.0/(timeLstpacketreceived - timeLstpacketreceived)/1024/nStanodes;
delay=delaySum/ rxPackets;
```

4 Results & Analysis

In this section, detailed behavior of the system under different network parameters are discussed. Number of nodes, datarate model, packet size and node orientation are varied in order to calculate delay and throughput.

4.1 Throughput for different DataRate model

It is clear that average throughput is decreasing with increase in number of nodes. This is because as number of nodes increases, more traffic is generated.
But as observed in the figure 1 changing datarate model by keeping number of nodes same don't have much impact on overall threshold value.

This simulation is run for constant frequency=5MHz, Orientation model= random box, packet size=1472bytes.

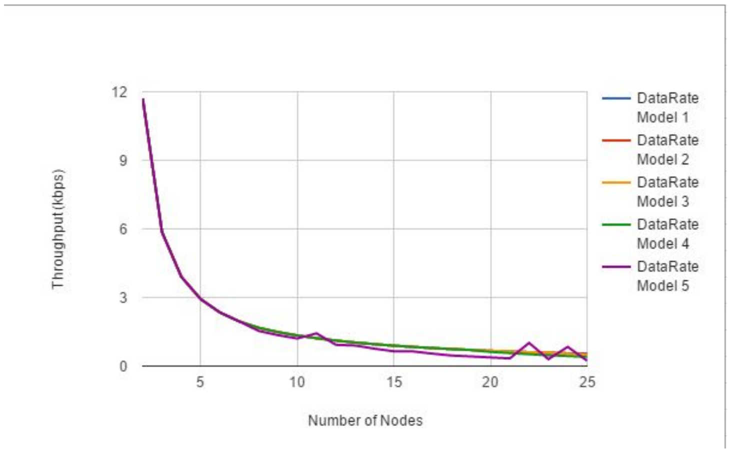


Figure 1: Throughput vs number of nodes under different datarate model

As shown in fig 1, throughput decreases as number of nodes increases. Also we can also conclude that throughput decreases as payload size decreases.

4.2 Throughput for different Packet size

Packet size will have more impact on threshold as the number of bytes transferred are increased.

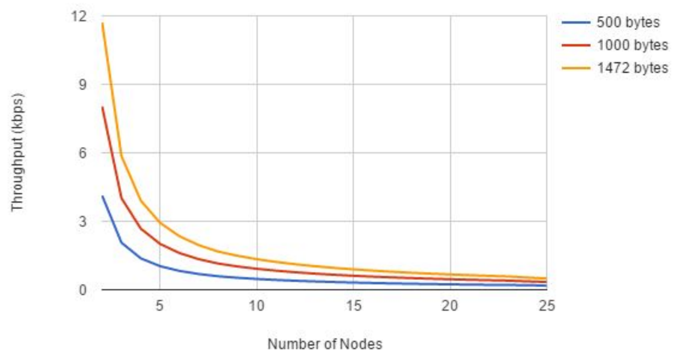
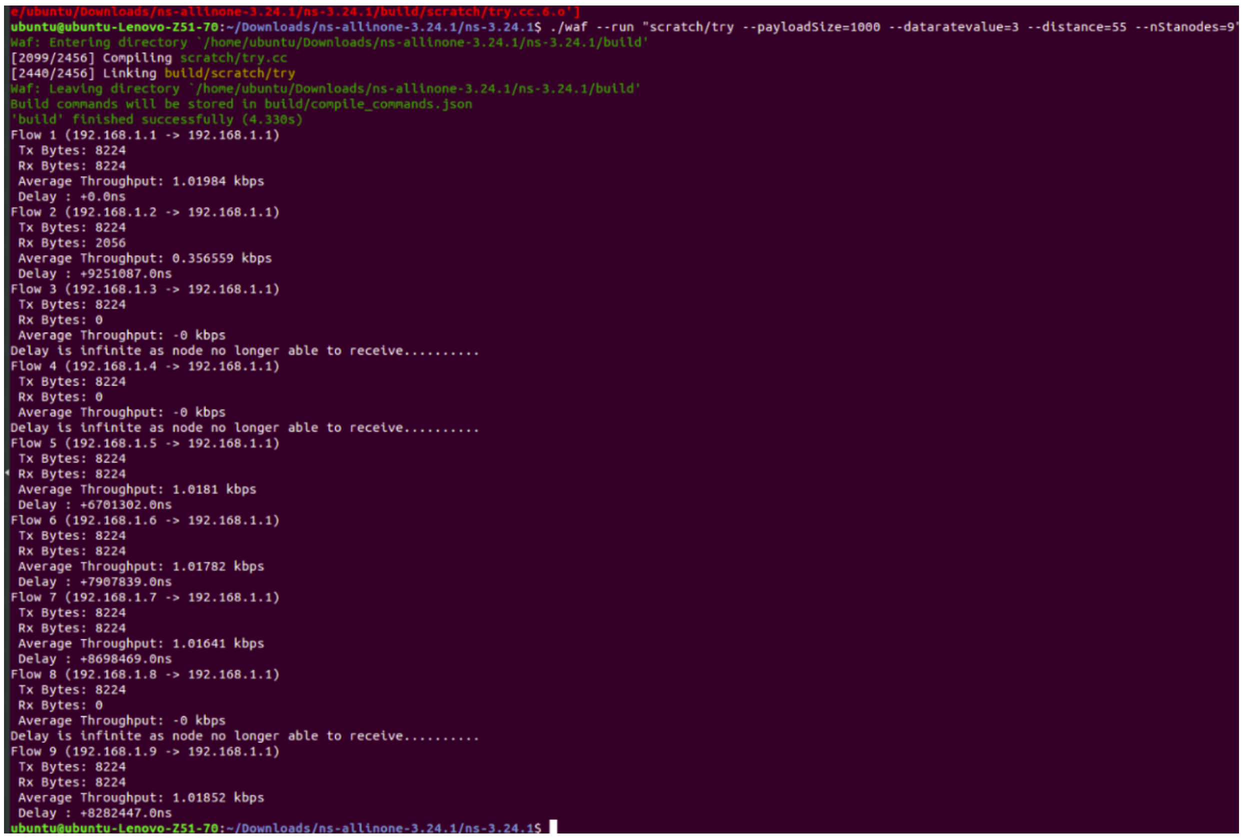


Figure 2: Throughput vs number of nodes under different Packet size

When number of nodes are increased above 20, then throughput for some nodes became zero. This is due to constant collision of packets, it causes packet to get lost in transmission and resulting in undefined value to delay.

This can be concluded from code out from fig 3. In this output node 2,3 and 7 corresponding to flow 3,4 and 8. And they are not able to receive any packets as RX bytes are zero. and delay can not be calculated. If delay is calculated by this method then **signal SIGSEGV** error is received. So it is needed to use debugger for such case. Hence there is a **if condition** in performance calculation block of script.



```
e/ubuntu/Downloads/ns-allinone-3.24.1/ns-3.24.1/build/scratch/try.cc.6.o']
ubuntu@ubuntu-Lenovo-Z51-70:~/Downloads/ns-allinone-3.24.1/ns-3.24.1$ ./waf --run "scratch/try --payloadSize=1000 --dataratevalue=3 --distance=55 --nStanodes=9"
Waf: Entering directory `/home/ubuntu/Downloads/ns-allinone-3.24.1/ns-3.24.1/build'
[2099/2456] Compiling scratch/try.cc
[2440/2456] Linking build/scratch/try
Waf: Leaving directory `/home/ubuntu/Downloads/ns-allinone-3.24.1/ns-3.24.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.330s)
Flow 1 (192.168.1.1 -> 192.168.1.1)
 Tx Bytes: 8224
 Rx Bytes: 8224
 Average Throughput: 1.01984 kbps
 Delay : +0.0ns
Flow 2 (192.168.1.2 -> 192.168.1.1)
 Tx Bytes: 8224
 Rx Bytes: 2056
 Average Throughput: 0.356559 kbps
 Delay : +9251087.0ns
Flow 3 (192.168.1.3 -> 192.168.1.1)
 Tx Bytes: 8224
 Rx Bytes: 0
 Average Throughput: -0 kbps
Delay is infinite as node no longer able to receive..........
Flow 4 (192.168.1.4 -> 192.168.1.1)
 Tx Bytes: 8224
 Rx Bytes: 0
 Average Throughput: -0 kbps
Delay is infinite as node no longer able to receive..........
Flow 5 (192.168.1.5 -> 192.168.1.1)
 Tx Bytes: 8224
 Rx Bytes: 8224
 Average Throughput: 1.0181 kbps
 Delay : +6701302.0ns
Flow 6 (192.168.1.6 -> 192.168.1.1)
 Tx Bytes: 8224
 Rx Bytes: 8224
 Average Throughput: 1.01782 kbps
 Delay : +7907839.0ns
Flow 7 (192.168.1.7 -> 192.168.1.1)
 Tx Bytes: 8224
 Rx Bytes: 8224
 Average Throughput: 1.01641 kbps
 Delay : +8698469.0ns
Flow 8 (192.168.1.8 -> 192.168.1.1)
 Tx Bytes: 8224
 Rx Bytes: 0
 Average Throughput: -0 kbps
Delay is infinite as node no longer able to receive..........
Flow 9 (192.168.1.9 -> 192.168.1.1)
 Tx Bytes: 8224
 Rx Bytes: 8224
 Average Throughput: 1.01852 kbps
 Delay : +8282447.0ns
ubuntu@ubuntu-Lenovo-Z51-70:~/Downloads/ns-allinone-3.24.1/ns-3.24.1$
```

Figure 3: Zero throughput scenario

4.3 Delay for different DataRate model

As per the hypothesis, increase in delay is observed when increasing number of nodes. It may be caused by more traffic and collision on the network.
This analysis is done keeping other parameters same, hence we can conclude that changing datarate model causes huge impact on delay value (it increases exponentially).
In scenarios for more then 20 nodes some nodes became unreachable. It can be refereed from figure 3.

So our hypothesis is correct. In order to get best throughput and less delay, number of nodes should be less.

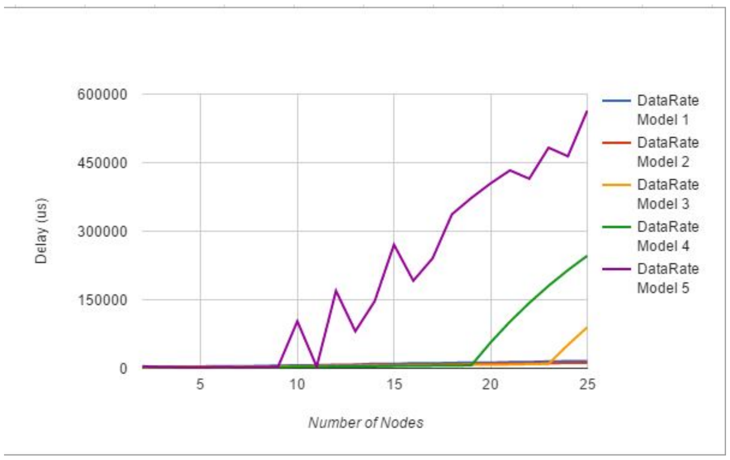


Figure 4: Delay vs number of nodes under different datarate model

4.4 Delay for different Packet Size

As packet data size increases delay increases as well. This can be concluded from fig 5. As packet size increases transmission time also increase for that packet resulting in increased delay.
Packet size is set to 1472, 1000 and 500 bytes.

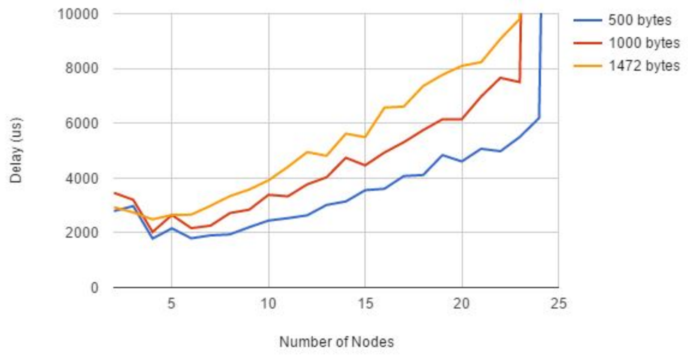


Figure 5: Delay vs number of nodes under different packet size

4.5 Effect of Node positioning

In this section different nodes are kept constant at 20 and their configuration is changed a described earlier. These configurations are `Grid positioning\, Rectangular positioning,Random rectangular positioning and Random disc positioning`.

These configurations don't have significant impact on throughput and delay 6 as nodes are always clustered and are in range of access-point.

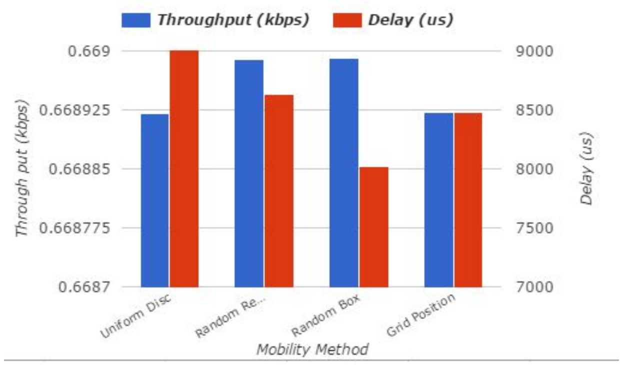


Figure 6: Delay and Throughput in different positioning scenario

Figure 7 shows one of these configuration in NetAnim simulator.

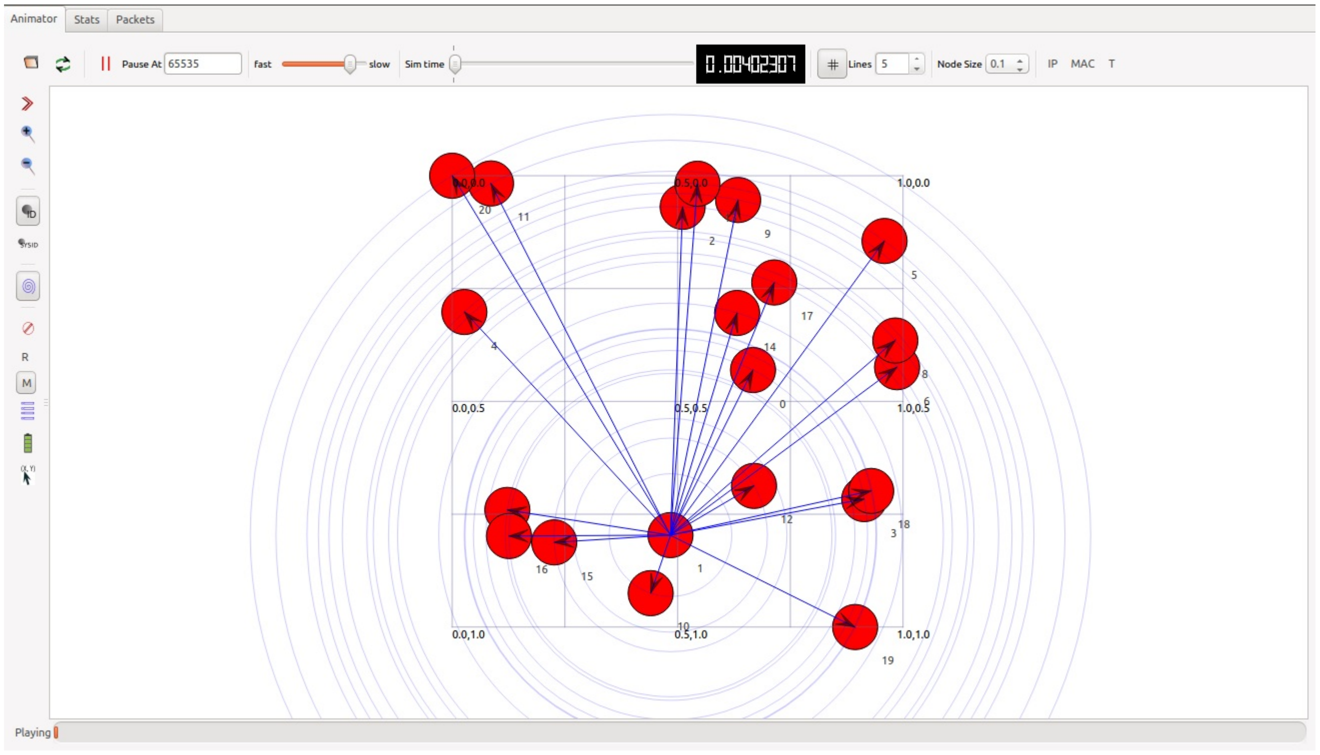


Figure 7: Random Box orientation

5 Conclusion

From results we can conclude that for WIFI 802.11n, as distance of AP increases from STA, throughput decreases. This means AP has be as close to all nodes in best case performance in addition with higher packet size. Number of nodes also plays a important part in traffic producing. To have optimal throughput and less delay, number of nodes must be as less as possible.
From analysis of simulation results we can conclude following things:

1. As node density increases in a keeping positions model and packet size same, it is observed that overall throughput decreases exponentially and delay for communication response increases linearly due to data collisions. This results in unreadability of few nodes.

2. Changing orientation of nodes in range of access-point hardly make any difference in overall throughput and delay.

3. To increase throughput of system packet size should be more. But this also increases overall delay of system due to more time to transfer and more frequent collisions. A trade off have to made in order to achieve maximum performance.

4. Changing the datarate model hardly have any difference on throughput, but it increases delay exponentially.

6 References

- https://www.nsnam.org/docs/tutorial/html/
- Lecture slides
- http://www.learnpython.org/

7 Appendix

7.1 run.py script

```
#!/usr/bin/env python2

import os

import subprocess


payload = [1472,1000,500]


#iteration for mobility method for constant values


 for t in range(1,4):

    z='./waf --run "scratch/main_code --nStanodes=20 --frequency=5 --simulationTime=10 --pa

    print z

    direct_output = subprocess.check_output(z, shell=True)
```

7.2 NS3 code

Source code can be found on GitHub link:

https : //github.com/shashwat91/Wireless_Networking−ET4394/blob/master/NS−3/Main_code.cc