Wireless Networking - NS3 Assignment

Pranav Sailesh Mani
Student No. 4508467
TU Delft

April 30, 2016

Contents

1 Introduction 3

2 Objective 3

3 Hypothesis 3

4 Implementation 4

5 Results and Analysis 5

6 Conclusion 6

7 References 6

8 Appendix 7

1 Introduction

Network Simulation is a common technique used in today's world to simulate the behavior of network and the devices connected to it before real world implementation. This would give a good idea on how the devices and the network would behave. As a part of the wireless networking course, the scenario I have decided to simulate is the effect of **Number of nodes on Packet Loss**. An attempt has been made to simulate a star connected WiFi network with one Access Point(AP) and varying number of nodes.

2 Objective

The objective of this assignment is to simulate a WiFi network consisting of a single Access Point and multiple nodes connected to it. I have chosen to implement the IEEE 802.11b WiFi standard. The objective is to monitor the packets lost as other parameters such as payload size and data rate vary. The network model to be implemented is similar to the one shown in Figure 1 except that we can vary the number of nodes.

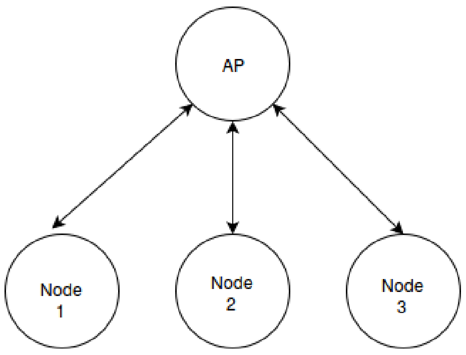

Figure 1: Network Topology to be simulated

3 Hypothesis

The following hypothesis is formulated:

- It is expected that as the number of nodes increase the packet loss also increases. This must be due to the fact that as the number of nodes increases, the collision of packets must increase thereby causing more packet loss.
- It is expected that as the data rate increases the packet loss also increases. Since more data is sent through in a smaller time frame, it is logical that the packet loss also would increase.
- It is expected that as the payload size increases the packet loss also increases. As the payload size increases, more packet loss is expected as this increases the stress on the nodes to deliver more amount of data in the same data rate.

On the whole, we expect packet loss to be minimum when the datarate, payload and number of nodes are minimal.

4 Implementation

The system mentioned above is simulated in ns3. The network topology to be simulated is IEEE 802.11b infrastructure based with one Access points and varying number of nodes as set by the user. The following steps are performed in ns3:

- NodeContainer class is used to define the Access point(only one) and the number of nodes as defined by the user.
- The channel is then created using YansWifi helper. The channel is set to IEEE 802.11b standard and constant speed propogation delay model and Friis propogation model is specified so as to simulate a real world like situation,
- After this the MAC parameters are defined. The data rate and control rate of the algorithm is specified. The data rate is controlled by a user defined variable. The control rate is kept at the default value.
- The devices to be attached to the nodes are then configured using NetDeviceContainer.
- Then the mobility model for the access point and the nodes have to be configured. We use static mobility models for both the devices. We use a grid like arrangement for arranging the nodes. The nodes are placed at a certain distances between them. This mobility model is then installed on to the nodes.
- Then, the Internet Stack is installed on to the devices using InternetStackHelper. The IP addresses to be assigned are set using Ipv4AddressHelper.
- The UDP server and the client application are then installed on to the nodes using UdpServerHelper and UdpClientHelper. One of the nodes is installed with the server while rest of the nodes are installed with the client application.
- The simulation and the flow monitor is then started. The simulation data is stored in a CSV file.

Some of the important things to note in our model are that we use nQoS MAC(i.e the network does not use QoS) and we generate UDP packets which does not need any handshake. The simulated network is shown in Figure 2. Note that the Access point is not visible since it overlaps with the position of one of the nodes. This does not effect the simulation result in anyway.

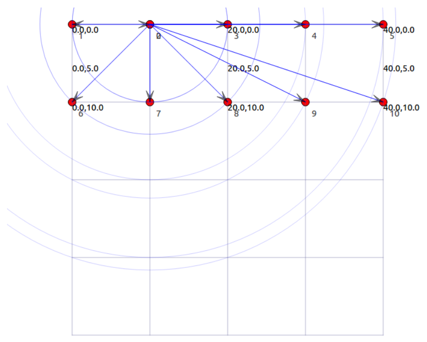

Figure 2: Graphical view of the simulated network in NetAnim for 10 nodes and 1 AP.

5 Results and Analysis

The first scenario that was simulated was the number of nodes versus the packet loss. The datarate was kept constant while varying the number of nodes and measuring the packet loss. It must be noted that the packet loss is found by taking the average of packets lost by each node. The result for this test is shown in Figure 3.

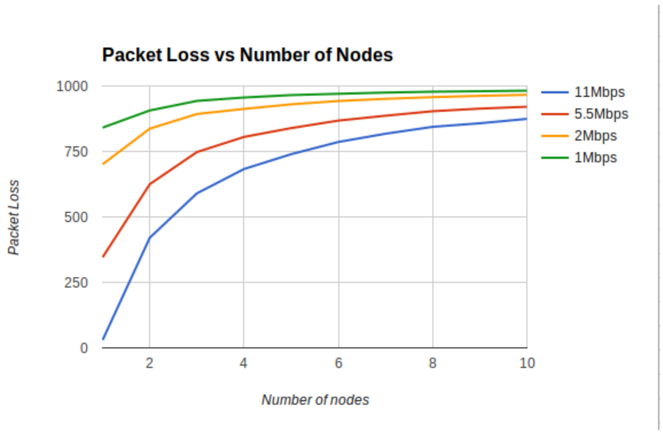

Figure 3: Graph of Packet loss vs Number of nodes for different data rates

It can be observed that as the number of nodes increase the packet loss also increases. This confirms our hypothesis that as the number of nodes increase, the packet loss also increases. Another interesting component of the above graph is that it can be seen that as the data rate decreases the packet loss increases. This is opposite to what was hypothesized. The next scenario to be simulated is the packet loss versus the number of nodes for different payloads. This is shown in Figure 4.

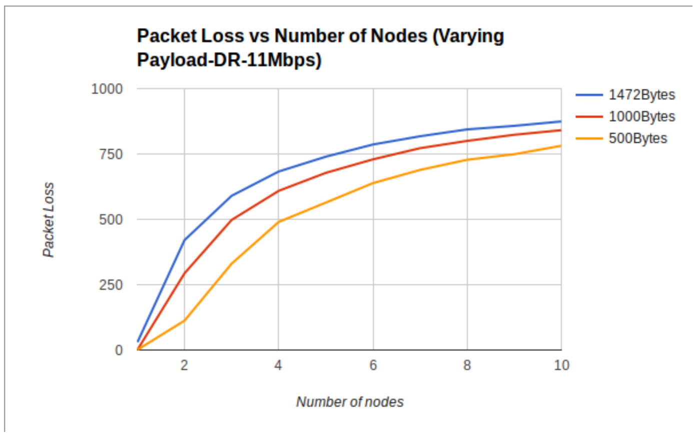

Figure 4: Graph of Packet loss vs Number of nodes for different data rates for different Payload sizes.

It can be seen in Figure 4 that as the payload increases the packet loss also increases thereby confirming our hypothesis.

6 Conclusion

In the beginning,we formulated three hypothesis out of which 2 have been confirmed. One, however has not been confirmed and that is as the data rate increases the packet loss would increase. However, this is not so and we observe the opposite. As the data rate increases the packet loss decreases. This could be due to the UDP Client Server application that is running on the nodes. This application could utilize the bandwidth that can cause the impact on the packet loss. The client server application would run better at higher data rate than at a lower data rate. This could lead to lesser packet loss at higher data rate. This causes our previously devised hypothesis hypothesis to fail. The following things could be concluded for IEEE 802.11b standard:

- As the number of nodes increase, the packet loss also increases.
- As the data rate increases, the packet loss decreases provided the nodes are running a UDP Client Server Application.
- As the payload size increases, the packet loss also increases.

7 References

1. https://www.nsnam.org/docs/tutorial/html/
2. Lecture Slides
3. Rizqi Hersyandika, NS3 Project Report

8 Appendix

The ns3 code used to simulate the environment is listed below. Please note that the data rate must be changed manually. The data generated is stored in a CSV file.

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/internet-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/random-variable-stream.h"
#include "ns3/netanim-module.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("WifiSimulate");
int main (int argc, char *argv[])
{
//Defining the inital parameters of the program such as payload,number of nodes
int numNodes = 10;
uint32_t payloadSize = 1472 ;

//Getting user variables
CommandLine cmd;
cmd.AddValue ("numNodes", "Number of Nodes", numNodes);
cmd.AddValue ("Payload", "Size of Payload", payloadSize);
cmd.Parse (argc,argv);

//The data will be stored in a file called data.csv
std::ofstream data_file("data.csv", std::ios::out | std::ios::app);
data_file << "Number_of_nodes,Payload Size,PacketLoss \n";

//Simulated for nNodes number of nodes
for(int nNodes=1; nNodes<=numNodes; nNodes++)
{
//Defining simulation starting and stop time and distance for mobility model.
double StartTime = 0.0;
double StopTime = 10.0;
int distance=10;
//Defining parameters to be used later in the program. Data Rate and maxPacket for client
StringValue DataRate;
DataRate = StringValue("DsssRate11Mbps"); //must be changed manually to change datarate
uint32_t maxPacket = 10000 ;
```

```
// Creating access point and nodes
NodeContainer wifiApNode;
wifiApNode.Create (1);
NodeContainer wifiStaNodes;
wifiStaNodes.Create (nNodes);

//Creating the physical channel
YansWifiChannelHelper channel;
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
channel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
channel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
phy.SetChannel (channel.Create ());

//Setting the Wifi Standards
WifiHelper wifi = WifiHelper::Default ();
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);


// configuring control parameters
        wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager","DataMode", DataRate,
NqosWifiMacHelper mac = NqosWifiMacHelper::Default ();

//Defining ssid
Ssid ssid = Ssid ("WifiSim");
mac.SetType ("ns3::StaWifiMac","Ssid", SsidValue (ssid),"ActiveProbing", BooleanValue (fal
NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);
mac.SetType ("ns3::ApWifiMac","Ssid", SsidValue (ssid));
NetDeviceContainer apDevice;
apDevice = wifi.Install (phy, mac, wifiApNode);

//Defining Mobility models
MobilityHelper mobility;
MobilityHelper mobility1;
        Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator> ();
        positionAlloc->Add (Vector (distance, 0, 0));
        mobility1.SetPositionAllocator (positionAlloc);
mobility1.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.SetPositionAllocator ("ns3::GridPositionAllocator","MinX", DoubleValue (0.0),"Min
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiStaNodes);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility1.Install (wifiApNode);

//Installing the Internet Stack over the devices
InternetStackHelper stack;
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);
Ipv4AddressHelper address;
```

```
Ipv4Address addr;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer staNodesInterface;
Ipv4InterfaceContainer apNodeInterface;
staNodesInterface = address.Assign (staDevices);
apNodeInterface = address.Assign (apDevice);
for(int i = 0 ; i < nNodes; i++)
{
addr = staNodesInterface.GetAddress(i);
std::cout << " Node " << i+1 << "\t "<< "IP Address "<<addr << std::endl;
}
addr = apNodeInterface.GetAddress(0);


//Installing the client and the server app on the nodes
ApplicationContainer serverApp;
UdpServerHelper myServer (4001);
serverApp = myServer.Install (wifiStaNodes.Get (0));
serverApp.Start (Seconds(StartTime));
serverApp.Stop (Seconds(StopTime));
UdpClientHelper myClient (apNodeInterface.GetAddress (0), 4001);
myClient.SetAttribute ("MaxPackets",UintegerValue (maxPacket));
myClient.SetAttribute ("Interval", TimeValue (Time ("0.002")));
myClient.SetAttribute ("PacketSize", UintegerValue (payloadSize));
ApplicationContainer clientApp = myClient.Install (wifiStaNodes);
clientApp.Start (Seconds(StartTime));
clientApp.Stop (Seconds(StopTime+5));
std::cout << "Simulation Model Generated and UDP communication has started" << '\n';

//Flow Monitor and netanim usage definition so
FlowMonitorHelper flowmon;
Ptr<FlowMonitor> monitor = flowmon.InstallAll();
Simulator::Stop (Seconds(StopTime+2));
AnimationInterface anim("Animation.xml");
//Start of simulation process and monitoring
Simulator::Run ();
monitor->CheckForLostPackets ();
Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifie
std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
double pac[100];
for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin (); i != sta
{
       pac[i->first]=i->second.lostPackets;
Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
std::cout << "Flow " << i->first  << " (" << t.sourceAddress << " -> " << t.destinationAdd
std::cout << "  Tx Bytes:   " << i->second.txBytes << "\n";
std::cout << "  Rx Bytes:   " << i->second.rxBytes << "\n";
std::cout << "  Packet Loss:   " << i->second.lostPackets << "\n";
}
//Calculation of average packet loss by taking average of packet loss in each node. Storin
double avgpac=0;
for (int i=1;i<=nNodes;++i)
{
avgpac = avgpac + pac[i];
}
avgpac=avgpac/nNodes;
std::cout << " Avg Packet Loss:   " << avgpac << "\n";
data_file << nNodes<<","<<payloadSize<<","<<avgpac<<","<<"\n";

Simulator::Destroy ();
}

return 0;
}
```