Yangyong Zhang
Satya Aditya Praneeth Emani
Aditya Sharma

PROJECT REPORT 2

This project report compiles the work done in simulating downlink transmission using the baseline policy and details the shortcomings of the policy used.

INTRODUCTION

Each wireless communication system is divided into an Access Point, or AP, through which all the packets for that autonomous system are routed. There are multiple clients in the system which send and receive packets. An uplink transmission is one wherein the client in the communication system sends a packet to the Access Point. A downlink transmission is one wherein the Access Point forwards the data packets it receives for a client to that particular client. There are different policies that are used for uplink and downlink transmissions. Let us consider Downlink transmissions. In downlink transmissions, the AP transmits to the clients based on a policy that is created. The AP has a computational part that decides upon the packet to be sent based on the policy and a hardware part that sends the actual packet. Once a packet has been transmitted to the hardware, the AP no longer has any control over the packet. It is not able to recall the packet or drop the packet. The hardware does not have any power to make decisions on the packet. Once the policy forwards the packets to the hardware, it will transmit packets according to the order that you forward them. The hardware will keep transmitting packets until they are all delivered, and it will not delete a packet even when the packet expires. Finally, upon the delivery of a packet, the hardware will be able to send a message to the policy to inform it that a packet has been successfully delivered. We can then define a baseline policy that can be used by the AP to forward packets to the hardware. The baseline policy explains how to forward packets to the hardware. In our case, the baseline policy is:

- At the beginning of an interval, the policy forwards all the packets it receives for every client to the hardware
- The policy forwards the packets to the hardware based on the debts of each client. The debt is calculated as

$$r_n(k+1) = (r_n(k) - u_n(k))^+ + w_n$$

where $r_n(k)$ is the debt for client n at the kth time slot,

$u_n(k) = 1$, if client n transmits in kth time slot

0, otherwise

W_n = number of time slots reserved for client n in the time interval

- The policy does not allow us the opportunity to drop packets and not send them to hardware. It also does not allow us the luxury of delaying packets or changing order of packets apart from debt.

OBJECTIVE

Our objective in this report is to show that the baseline policy described above does not function properly in all cases and will completely fail in some cases.

SIMULATION SETUP

We consider the following two scenarios and simulate them as follows:

SCENARIO 1

Consider the following scenario:

1. The AP has 10 clients
2. The channel has 100% reliability. In order to have 100% reliability, we place the nodes extremely close together and we follow the Two Ray Ground Model. The layout and the setup are as follows:

```
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
set val(nn)             11                          ;# number of mobilenodes
```

Figure 1 – Setup for the Channel

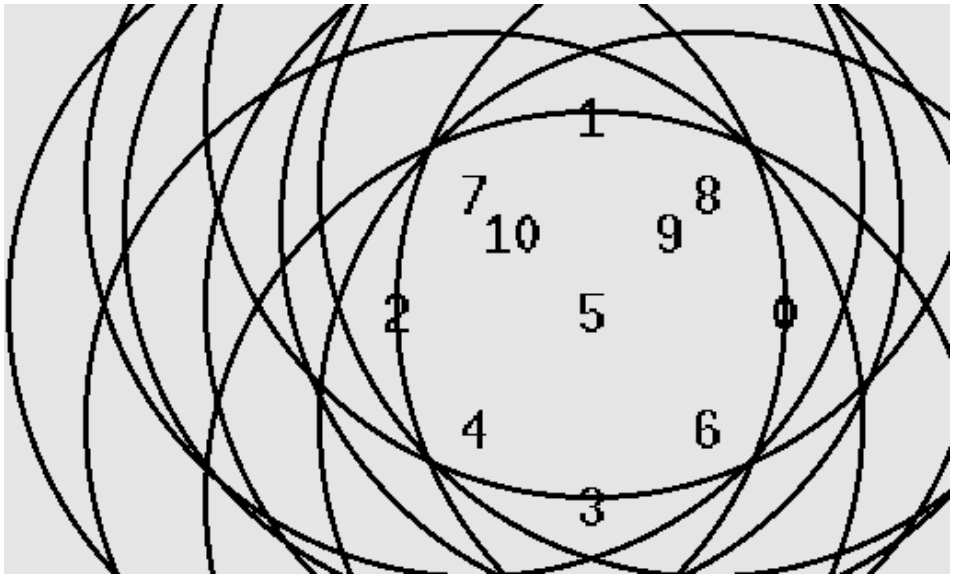

Figure 2 – Layout of the Nodes for Scenario 1

3. The time Interval T = 100ms and each time slot is around 10ms. Thus, each time interval is divided into 10 time slots.

```
set intvl [ expr { 10000 * 10 }]
set totalt [ expr { $intvl * $val(rept) }]
```

Figure 3 – Interval Time and Total Time

4. The AP gets 2 packets for each client in the time interval. That is, the AP gets a total of 20 packets each time interval.

The simulation is run for a total time of 2 seconds that is, the simulation is run for 20 time intervals, which consists of 200 time slots. We send the packets out to the hardware according to the debts of the clients. Since the reliability is 100%, the debts will be updated such that each client will get to transmit once in the time interval in a round robin fashion. The simulation is conducted without dropping of any packets before sending to the hardware.

SCENARIO 2

Consider the following scenario:

1. The AP has 2 clients
2. The channel has 50% reliability. In order to have 50% reliability, we place the nodes at a distance of 83 apart depending upon the graphs as shown below for the parameters as shown and we follow the Shadowing Model. The layout and the setup are as follows:

```
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/Shadowing   ;# radio-propagation model
set val(netif)          Phy/WirelessPhy            ;# network interface type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
```

Figure 4 – Channel Setup for Scenario 2

```
$val(prop) set pathlossExp_ 3.0  ;# path loss exponent
$val(prop) set std_db_ 6.0       ;# shadowing deviation (dB)
$val(prop) set dist0_ 1.0        ;# reference distance (m)
$val(prop) set seed_ 0           ;# seed for RNG
```

Figure 5 – Shadowing Channel Configuration for Scenario 2

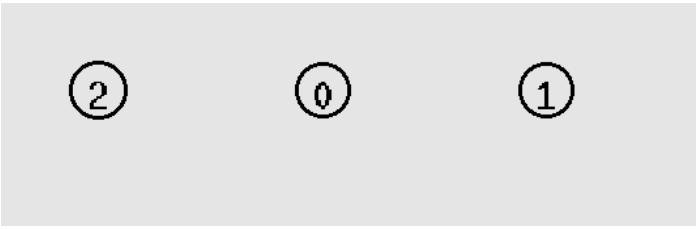

Figure 7 – Layout for Scenario 2

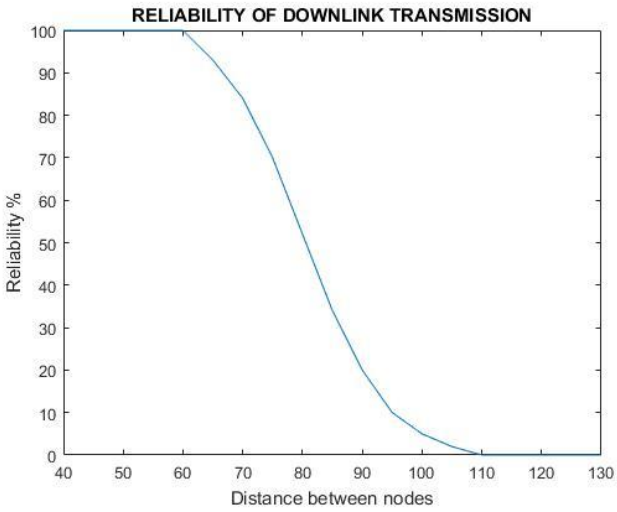

Figure 6 – Reliability plot for Downlink Transmission

3. The time Interval T = 400ms and each time slot is around 100ms. Thus, each time interval is divided into 4 time slots.

```
set intvl [ expr { 100000 * 4 }]
set totalt [ expr { $intvl * $val(rept) }]
```

Figure 8 – Interval Time and Total Time

4. The AP gets 1 packet for each client in the time interval. That is, the AP gets a total of 2 packets each time interval.

The simulation is run for a total time of 20 seconds that is, the simulation is run for 50 time intervals, which consists of 200 time slots. We send the packets out to the hardware according to the debts of the clients. Since the reliability is 50%, the debts will be updated such that each client will get to transmit twice, each one alternating in the time interval. The simulation is conducted without dropping of any packets before sending to the hardware.

OBSERVATIONS

Based on the simulations that were conducted, we have the observations as follows.

SCENARIO 1

For a feasible system, we have the condition for Debt First policy as follows:

$$\sum_{n \epsilon S} Wn \leq T - I_s$$

Since the number of clients is 10 and the time slots per time interval is 10, we cannot have an I_s. Since p = 1, and T = 10, we must have $\sum_{n \epsilon S} Wn = q_n \leq 10$. Thus, we cannot send more than 10 packets to the hardware for all the clients combined. The output of the NS-2 simulation is as follows:

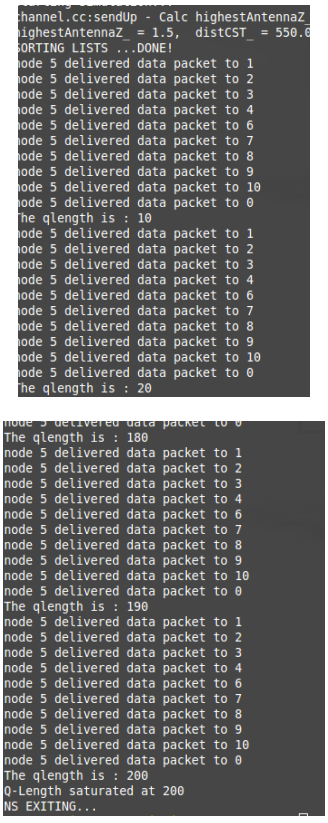

Figure 9 – Output of NS -2 for Scenario 1

However, our baseline policy does not account for this. At the beginning of each time interval, all the packets are forwarded to the hardware without dropping and thus all 20 packets will be forwarded. Thus, the number of packets in the hardware queue will keep increasing. Also, since the clients have similar chances at the output and the reliability is 100%, the timely throughput for each client will be the same. The plots for these are as follows:

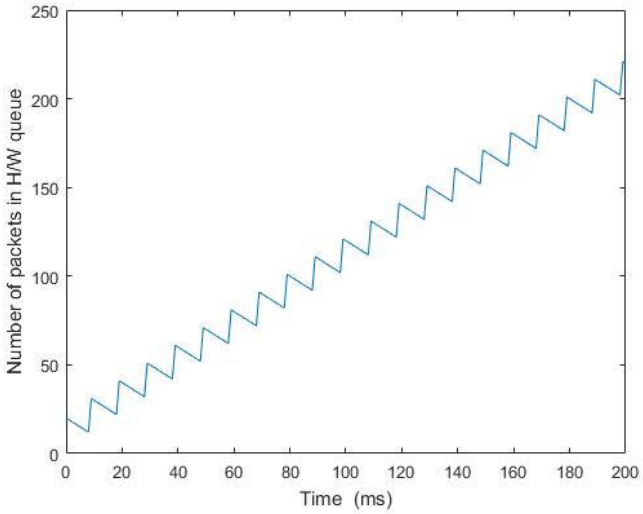

Figure 10 – Plot of number of packets in hardware queue

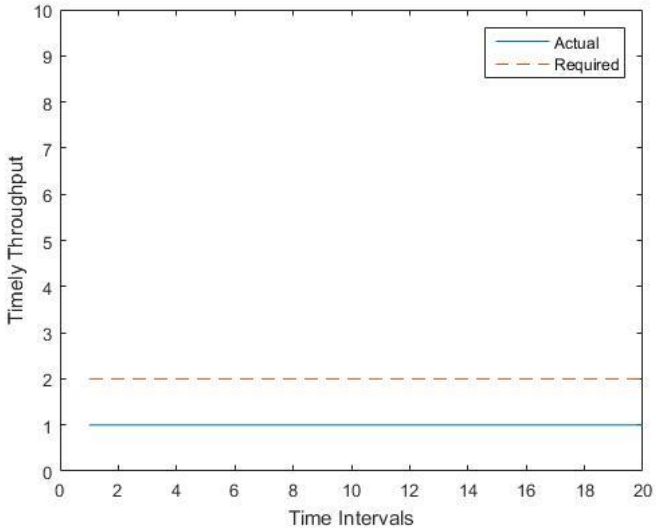

Figure 11 – Plot of Timely Throughput for client n

From the above data, we can say that the baseline policy fails for the first scenario.

SCENARIO 2

For a feasible system, we have the condition for Debt First policy as follows:

$$\sum_{n \epsilon S} Wn \leq T - I_s$$

Since the number of clients is 2 and the time slots per time interval is 4, we will have an I_s. Since p = 0.5, and T = 4, we must have $q_n \leq 0.8125$. Thus, while it is possible to send 2 packets to 2 clients in our scenario, we might not be able to send all of them. The output of the NS-2 simulation is as follows:

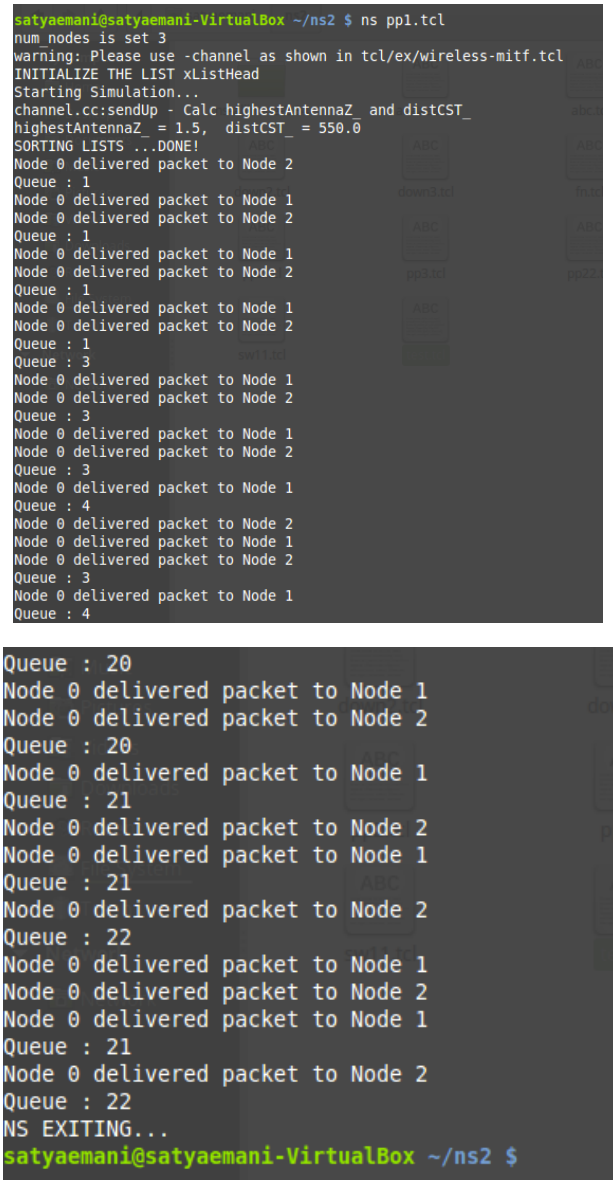

Figure 12 – Output from NS-2 for Scenario 2

However, our baseline policy does not account for this. Our baseline policy does not allow for us to drop packets. Thus, we see that packets are carried over from the previous interval if they are not sent. Thus, the number of packets in the hardware queue will keep increasing. The plots for the hardware queue length as well as the timely throughput for the clients are as follows:

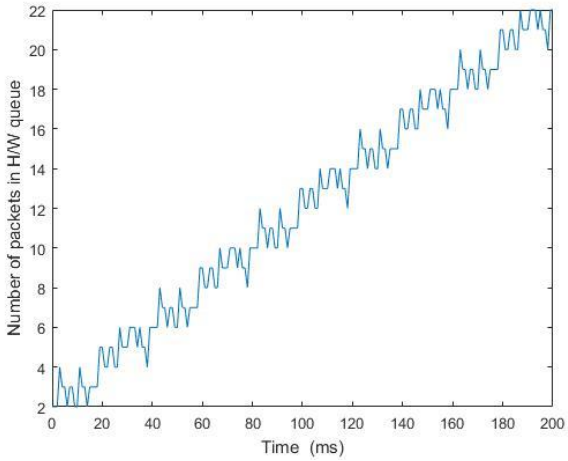

Figure 13 - Plot of number of packets in hardware queue

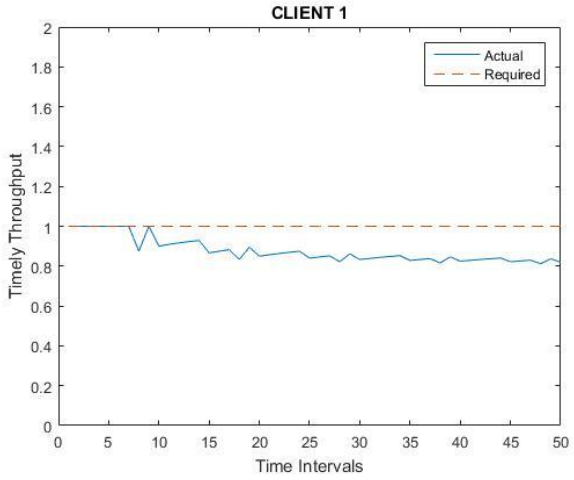

Figure 14 – Timely Throughput for Client 1

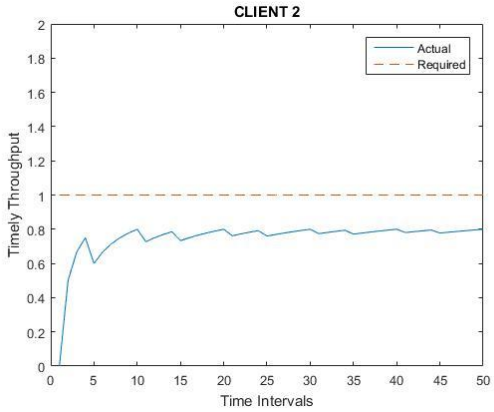

Figure 15 – Timely Throughput for Client 2

From the above data, we can say that the baseline policy fails for the second scenario.

CONCLUSION AND FUTURE WORK

As we can see from the observations, we have a baseline policy that completely fails in the two scenarios as described above. Thus, this would not be a good policy to implement in the AP as it could lead to a saturation of the queue or the buffer in the hardware of the AP and thus cause a failure in the system. Thus, it is evident that we need to make changes in the policy to remove these deficiencies in the system. The main problems in the system are the inability to delay packets and the inability to drop packets based on the simulations seen.

In future work, we intend to develop a working policy that takes care of the problems introduced into the system because of the deficiencies that are present in the baseline policy. By adding the ability to make decisions on the packet and dropping them or delaying them when necessary, we can make a beter policy that will function better.

ANSWER TO COMMENTS IN REPORT ONE

In our code, we have the receive node send an acknowledgement packet to the send node at the beginning of the next available time slot after which it receives a packet. In the previous simulation, the time slots at the receiver were not set at the minimum value so as to enable an immediate response from the receiver. Thus, we would get variable RTTs when the simulation was run depending on the amount of time that was left in the time slot at the receiver. Furthermore, during the first transmission, there is an additional delay due to initialization of the receive node. This error occurs consistently but is only present at the first transmission. Thus, the two changes that need to be made in the code are as follows:

1. Ignore the first packet transmission in the calculations in the maximum and minimum calculations.
2. Reduce the slot time so that the receiver immediately sends a ACK packet as soon as it receives a packet.

```
Mac/802_11 set SlotTime_ 1;


set ns_          [new Simulator]
set tracefd      [open simple.tr w]
$ns_ trace-all $tracefd
```

```
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
node 0 delivered data packet to  1 with 1184.4 ms
node 0 delivered data packet to  1 with 682.7 ms
node 0 delivered data packet to  1 with 782.7 ms
node 0 delivered data packet to  1 with 642.7 ms
node 0 delivered data packet to  1 with 762.7 ms
node 0 delivered data packet to  1 with 722.7 ms
node 0 delivered data packet to  1 with 202.7 ms
node 0 delivered data packet to  1 with 582.7 ms
node 0 delivered data packet to  1 with 622.7 ms
node 0 delivered data packet to  1 with 962.7 ms
node 0 delivered data packet to  1 with 1142.7 ms
node 0 delivered data packet to  1 with 642.7 ms
node 0 delivered data packet to  1 with 762.7 ms
node 0 delivered data packet to  1 with 642.7 ms
node 0 delivered data packet to  1 with 1062.7 ms
node 0 delivered data packet to  1 with 942.7 ms
node 0 delivered data packet to  1 with 942.7 ms
node 0 delivered data packet to  1 with 802.7 ms
node 0 delivered data packet to  1 with 282.7 ms
node 0 delivered data packet to  1 with 222.7 ms
node 0 delivered data packet to  1 with 1043.5 ms
node 0 delivered data packet to  1 with 982.7 ms
node 0 delivered data packet to  1 with 582.7 ms
node 0 delivered data packet to  1 with 602.7 ms
node 0 delivered data packet to  1 with 1042.7 ms
node 0 delivered data packet to  1 with 1102.7 ms
node 0 delivered data packet to  1 with 1282.7 ms
node 0 delivered data packet to  1 with 562.7 ms
node 0 delivered data packet to  1 with 634.9 ms
NS EXITING...
```

Figure 1 – RTTs for a value of Time Slot of 1

```
Mac/802_11 set SlotTime_ 0.05;


set ns_          [new Simulator]
set tracefd      [open simple.tr w]
$ns_ trace-all $tracefd
```

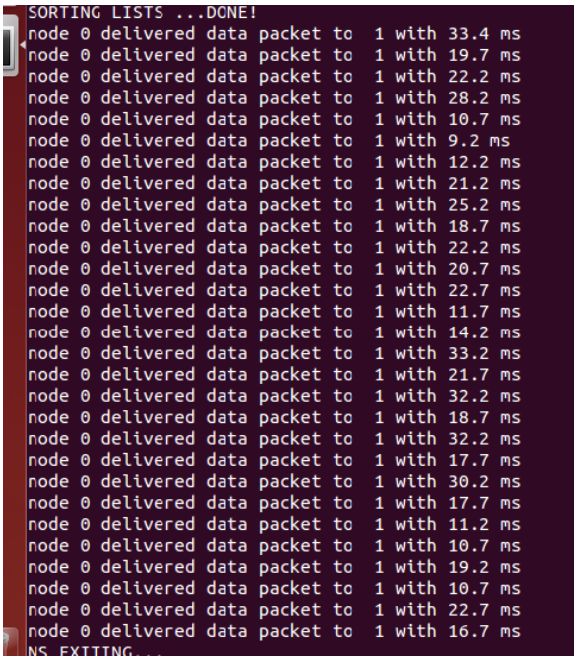

Figure 2 – RTT values for Time Slot of 0.05

```
Mac/802_11 set SlotTime_ 0.0000001;


set ns_          [new Simulator]
set tracefd      [open simple.tr w]
$ns_ trace-all $tracefd
```

```
INITIALIZE THE LIST xListHead
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
node 0 delivered data packet to  1 with 4.4 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
node 0 delivered data packet to  1 with 2.7 ms
NS EXITING...
```

Figure 3 – RTT values for Time Slot of 0.0000001

Thus, as can be seen from the above screenshots, we have a steady RTT as we lower the Time Slot. Thus, we can resolve the problem encountered in the previous report.