DUBLIN CITY UNIVERSITY

ELECTRONIC AND COMPUTER ENGINEERING

EE500 Network Performance

Assignment 1

Author

Michael Lenehan	michael.lenehan4@mail.dcu.ie
Student Number:	15410402

08/12/2019

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

I have read and understood the DCU Academic Integrity and Plagiarism at `https://www4.dcu.ie/sites/default/files/policy/1%20-%20integrity_and_plagiarism_ovpaa_v3.pdf` and IEEE referencing guidelines found at `https://loop.dcu.ie/mod/url/view.php?id=448779`.

Signed: ____________________ Date: 08/12/2019

Michael Lenehan

Contents

1 Part 1: Data Transmission over the WiFi Network

Part 1 of this assignment involves modifying the provided “wifi-example-sim.cc” file in order to execute simulations with multiple input parameters, such as different bitrates, distances between nodes, and number of nodes.

1.1 Question A:

This section assess the initial simulation implementation, with a single user, a data rate of approximately 1.6Kbps, and a distance of 50 meters between the user and the access point. The bitrate is then modified to assess its affect on the system.

1.1.1 Part 1:

Within the wifi-example-sim.cc file, the simulation is given a run time of 20 seconds, a packet size of 1000 bytes and a 0.05 second delay between the transmission of packets. This corresponds to a total of 400 packets sent over the space of 20 seconds.

$$
\begin{aligned}
R &= \frac{rxPackets * packetSize * 8}{delay} \\
&\frac{400 * 1000 * 8}{4.9 \times 10^{-4}} \\
&6.53 \times 10^{9} bits
\end{aligned}
$$

Equation 1: Bitrate of Data Traffic (Kbps)

$$
\begin{aligned}
Throughput &= \frac{rxPackets * packetSize * 8}{txTime} \\
&\frac{400 * 1000 * 8}{20} \\
&\frac{3,200,000}{20} \\
&160,000 \\
&160 Kbps
\end{aligned}
$$

Equation 2: Average Throughput (Kbps)

The throughput value from the equation above can be seen plot against the users distance from the access point in Figure 1 (plot in Mbps).

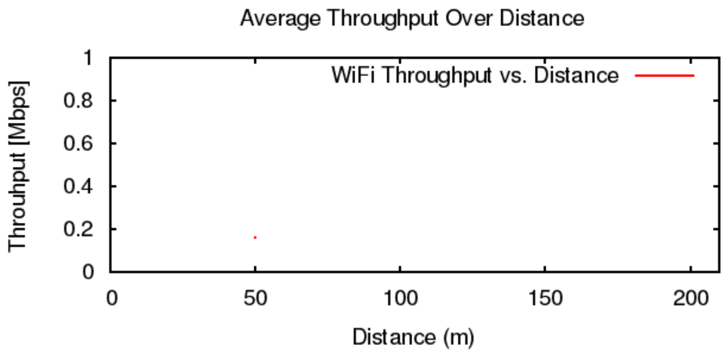

Figure 1: Throughput for a single user at a distance of 50 meters

Delay time is given within the wifi-example-sim.cc file as measured in nanoseconds. Form the output, this value is 490381ns. This can be calculated as follows:

$$
\begin{aligned}
\overline{delay} &= \frac{delaySum}{rxPackets} \\
&\frac{196,152,732}{400} \\
&= 490,381.83ns \\
&= 4.9 \times 10^{-4}s
\end{aligned}
$$

Equation 3: Average Delay (s)

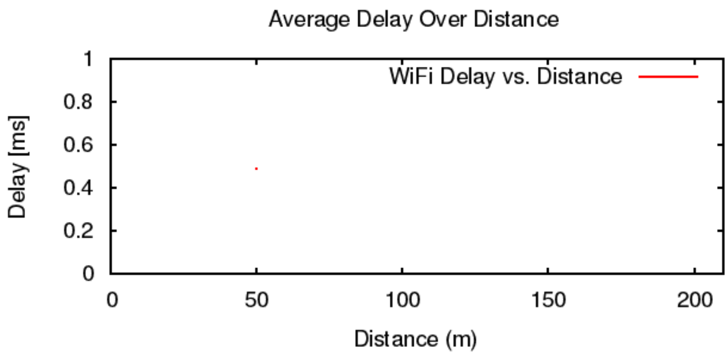

Figure 2: Delay for a single user at a distance of 50 meters

Packet loss ratio is given by the following formula:

$$PLR = \frac{lostPackets}{rxPackets + lostPackets0}$$
$$\frac{0}{400+0}$$
$$= \quad 0$$

Equation 4: Average Packet Loss Ratio

As seen in both the equation above, and in Figure 3, there is no packet loss within this simulation.

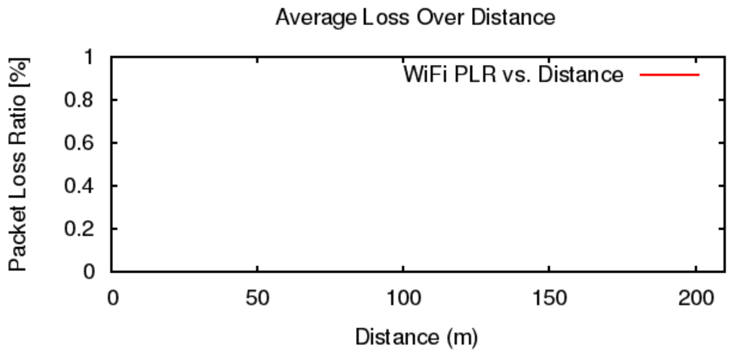

Figure 3: Packet Loss Ratio for a single user at a distance of 50 meters

1.1.2 Part 2:

Within the wifi-example-sim.cc file, the transmission bitrate can be modified in one of two ways. By modifying the size of the packets being transmit, or by modifying the delay between packet transmissions. This is shown in the following calculations:

$$R = \frac{rxPackets \times packetSize \times 8}{txTime}$$
$$\frac{R \times txTime}{packetSize \times 8} = rxPackets \qquad or \qquad \frac{R \times txTime}{rxPackets \times 8} = packetSize$$

$$rxPackets = \frac{txTime}{delay}$$

Equation 5: Calculation for Bitrate Modification

The following table shows the number of packets to be sent (modified by the delay between packet transmissions), or the size of packet to be sent in order to meet the required data rates.

Table 1: Bitrate Calculation Results

Bitrate	rxPackets	Delay	packetSize
1Mbps	2,500	8×10^{-3}	625
1.5Mbps	3,750	5.3×10^{-3}	937.5
5Mbps	12,500	1.6×10^{-3}	3125
10Mbps	25,000	8×10^{-4}	6250
20Mbps	50,000	4×10^{-4}	12,500

For the purposes of this section, the delay between the packet transmissions will be modified. Modifying packet sizes requires using half bytes for packet sizes. The modification within the code can be seen within the following snippet of the wifi-example-sim.cc file found in the Appendices.

Listing 1: Transmission Interval Code Snippet

```
sender[i]->SetAttribute("Interval", StringValue
("ns3::ConstantRandomVariable[Constant=" +
    txInterval + "]"));
```

The following equations show the throughput, delay, and packet loss ratio for the systems with the above bitrates.

$$
\begin{aligned}
Throughput &= \frac{rxPackets * 1000 * 8}{txTime} \\
TP_{8\times10^{-3}} &= \frac{2500 * 1000 * 8}{20} \\
&= 1000Kbps \\
TP_{5.3\times10^{-3}} &= \frac{3774 * 1000 * 8}{20} \\
&= 1509.6Kbps \\
TP_{1.6\times10^{-3}} &= \frac{12500 * 1000 * 8}{20} \\
&= 5000Kbps \\
TP_{8\times10^{-4}} &= \frac{24997 * 1000 * 8}{20} \\
&= 9998.8Kbps \\
TP_{4\times10^{-4}} &= \frac{34368 * 1000 * 8}{20} \\
&= 13747.2Kbps
\end{aligned}
$$

Equation 6: Average Throughput (Kbps)

From these calculations, it can be seen that a maximum throughput of 13.7Mbps was achieved. Loss begins to occur at approximately 10Mbps, limiting the throughput. The throughput can be seen plotted against the increasing transmission intervals in Figure 4.

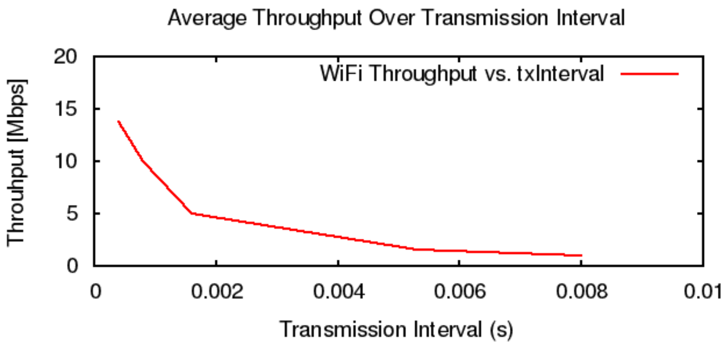

Figure 4: Throughput for transmission intervals as outlined in Table 1

$$
\begin{aligned}
\overline{delay} &= \frac{delaySum}{rxPackets} \\
\overline{delay}_{8\times10^{-3}} &= \frac{1102533332}{2500} \\
&= 441013.3328ns \\
\overline{delay}_{5.3\times10^{-3}} &= \frac{1649679480}{3774} \\
&= 437116.9793ns \\
\overline{delay}_{1.6\times10^{-3}} &= \frac{5484597352}{12500} \\
&= 438767.7882ns \\
\overline{delay}_{8\times10^{-4}} &= \frac{12434305114}{24497} \\
&= 497431.8964ns \\
\overline{delay}_{4\times10^{-4}} &= \frac{7873637069728}{34368} \\
&= 229097912.9ns
\end{aligned}
$$

Equation 7: Average Delay (s)

These calculations show that there is an increase in delay as the bitrate increases (i.e. the transmission interval decreases). This is further demonstrated in Figure 5.

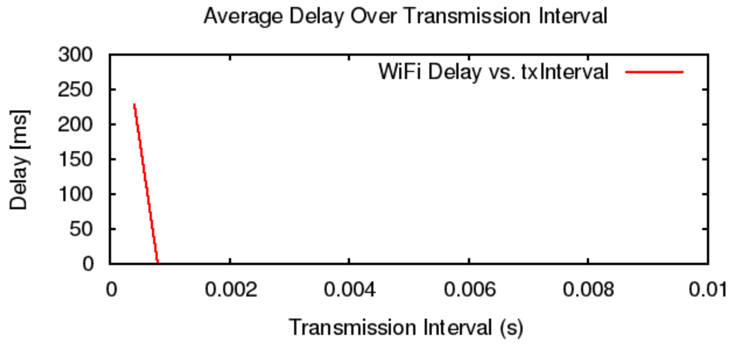

Figure 5: Delay for transmission intervals as outlined in Table 1

$$
\begin{aligned}
PLR &= \frac{lostPackets}{rxPackets + lostPackets} \\
PLR_{8\times10^{-3}} &= \frac{0}{2500+0} \\
&= 0 \\
PLR_{5.3\times10^{-3}} &= \frac{0}{3774+0} \\
&= 0 \\
PLR_{1.6\times10^{-3}} &= \frac{0}{12500+0} \\
&= 0 \\
PLR_{8\times10^{-4}} &= \frac{3}{24997+3} \\
&= 1.2\times10^{-4} \\
PLR_{4\times10^{-4}} &= \frac{15632}{34368+15632} \\
&= 0.30724
\end{aligned}
$$

Equation 8: Average Packet Loss Ratio

Again, as there is an increase in the bitrate, there is an increase in the loss within the system, with a maximum loss of 30%, which can be clearly seen within Figure 6.

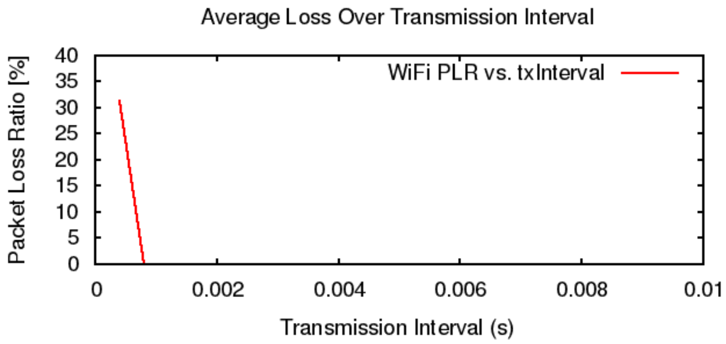

Figure 6: Loss for transmission intervals as outlined in Table 1

This section displays clearly that there is a direct relation between the bitrate and the throughput, delay, and loss. As the bitrate was increased in each test case, the throughput increased, until the point at which the loss and delay began to affect the linearity of the increase.

1.2 Question B:

This section investigates the effect of distance on throughput, delay and loss within a single user, single access point system.
The throughput, shown below, is constant for the user from distances between 0 and 110 meters from the access point. Packet loss occurs at approximately 120 meters, giving greatly decreased throughput. Between 130 and 150 meters, there are no packets received, giving a throughput of 0Kbps.

$$
\begin{aligned}
Throughput &= \frac{rxPackets * 1000 * 8}{txTime} \\
TP_{0-110} &= \frac{400 * 1000 * 8}{20} \\
&= 160Kbps \\
TP_{120} &= \frac{13 * 1000 * 8}{20} \\
&= 5.2Kbps \\
TP_{130-150} &= \frac{0 * 1000 * 8}{20} \\
&= 0Kbps
\end{aligned}
$$

Equation 9: Average Throughput (Kbps) over distances of 0-150m

The drop of in the throughput is clearly visible within the plot of Figure 7.

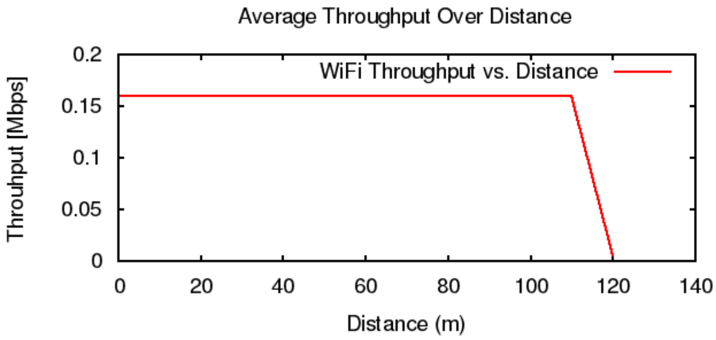

Figure 7: Throughput as measured at distances between 0 and 150 meters

The delay, whose values are shown below as read from the database file, begins to increase at a faster rate between 80 and 110 meters, with much greater increases between 110 and 120 meters.

$$\overline{delay}_0 = 271582ns$$
$$\overline{delay}_{10} = 271615ns$$
$$\overline{delay}_{20} = 271648ns$$
$$\overline{delay}_{30} = 397353ns$$
$$\overline{delay}_{40} = 381876ns$$
$$\overline{delay}_{50} = 490381ns$$
$$\overline{delay}_{60} = 605123ns$$
$$\overline{delay}_{70} = 649954ns$$
$$\overline{delay}_{80} = 835599ns$$
$$\overline{delay}_{90} = 1279691ns$$
$$\overline{delay}_{100} = 1574269ns$$
$$\overline{delay}_{110} = 1723305ns$$
$$\overline{delay}_{120} = 9362323ns$$

Equation 10: Average Delay (ns) over distances of 0-150m

Beyond a distance of 120m, due to there being a throughput of 0Kbps, there is no delay value. These values can be seen in Figure 8

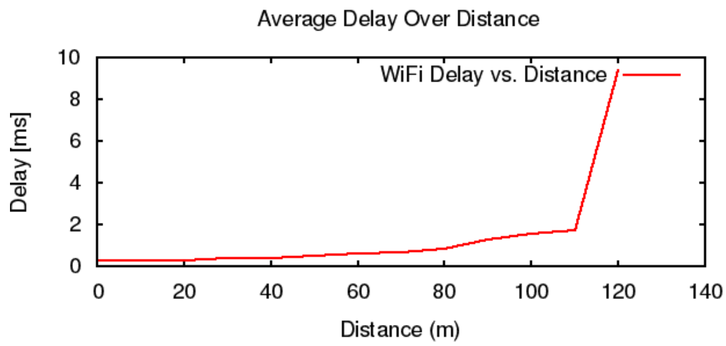

Figure 8: Delay as measured at distances between 0 and 150 meters

For distances between 0 and 110 meters, there is no packet loss due to, as discussed earlier, all of the packets transmit by the access point are received by the user.
For a distance of 120 meters, the user experiences loss of 96.75%, and for distances greater than this, experiences loss of 100%.
The calculations for the packet loss ratio can be seen below:

$$\begin{aligned}
PLR &= \frac{lostPackets}{rxPackets + lostPackets} \\
TP_{0-110} &= \frac{0}{400+0} \\
&= 0 \\
TP_{120} &= \frac{387}{13+387} \\
&= 0.9675 \\
TP_{130-150} &= \frac{400}{0+400} \\
&= 1
\end{aligned}$$

Equation 11: Packet Loss Ratio over distances of 0-150m

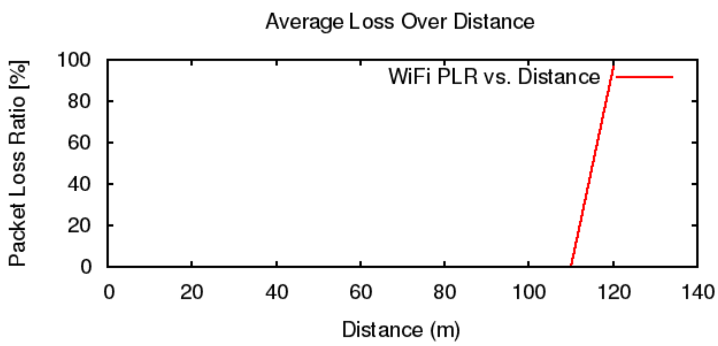

Figure 9: Packet Loss Ratio as measured at distances between 0 and 150 meters

1.3 Question C:

This section focuses on the effects of multiple users connected to the same access point. Combined with different distances between users and the access point, or larger numbers of users at the same distance from the access point, the throughput, delay, and PLR are calculated.

1.3.1 Part 1:

This section introduces a second user to the initial system, at a distance of 100 meters from the access point, i.e. twice the distance of the initial user. This second user was added by creating an input value which could be used to iterate through a loop, creating new sink nodes. For each of these sink nodes, a new source, with the corresponding listening port number was required to be created. This can be seen within the wifi-example-sim.cc code within the Appendices.
The throughput for both user 1 and user 2 can be seen below. The throughput of user 2 is lower than that of user 1, due to one lost packet. This packet is presumably lost due to the added distance from this user to the access point.

Listing 2: Loop Snippet for adding a source and sink to the provided code.

```
cmd.AddValue ("users", "Number of Users", users);

Ptr<Sender> sender[users];
Ptr<Node> appSource = NodeList::GetNode (0);
for(int i=0; i<users; i++){
  sender[i] = CreateObject<Sender>();
  sender[i]->SetAttribute("Port", UintegerValue(1000+i)
      );//Lisening Port of the first WiFi user
  //Code removed for the purposes of snippet
  sender[i]->SetStartTime (Seconds (0));
}

Ptr<Receiver> receiver[users];
for(int i=0; i<users; i++){
  Ptr<Node> appSink = NodeList::GetNode (i+1);
  receiver[i] = CreateObject<Receiver>();
  receiver[i]->SetAttribute("Port", UintegerValue(1000+
      i));//Lisening Port
  appSink->AddApplication (receiver[i]);
  receiver[i]->SetStartTime (Seconds (0));

  //Set IP address of the first User to AP (source)
  string ipString = "192.168.0."+std::to_string(i+2);
  const char* ip = ipString.c_str();
  //cout << ip;
  Config::Set ("/NodeList/*/ApplicationList/"+std::
      to_string(i)+"/$Sender/Destination",
               Ipv4AddressValue (ip));
}
```

$$
\begin{aligned}
Throughput &= \frac{rxPackets * 1000 * 8}{txTime} \\
TP_{user_1} &= \frac{5000 * 1000 * 8}{20} \\
&= 2000Kbps \\
TP_{user_2} &= \frac{4999 * 1000 * 8}{20} \\
&= 1999.6Kbps
\end{aligned}
$$

Equation 12: Average Throughput (Kbps)

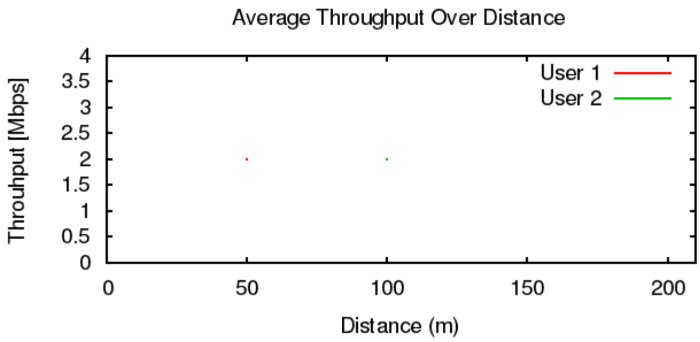

Figure 10: Throughput for 2 users, one at a distance of 50m, one at a distance of 100m

The calculations below show the difference in delay as shown in the database (".db") file. There is greater delay between user 2 and the access point, again, presumably due to the increased distance. This difference in delay however is not shown in the output plot, possibly due to an issue in wifi-p1qcp1.sh, which extracts the required information from the .db file using the "awk" command line tool.

$$
\begin{aligned}
\overline{delay} &= \frac{delaySum}{rxPackets} \\
\overline{delay}_{user1} &= \frac{2224452124}{5000} \\
&= 444890.4248ns \\
\overline{delay}_{user2} &= \frac{10823682785}{4999} \\
&= 2165169.591ns
\end{aligned}
$$

Equation 13: Average Delay (s)

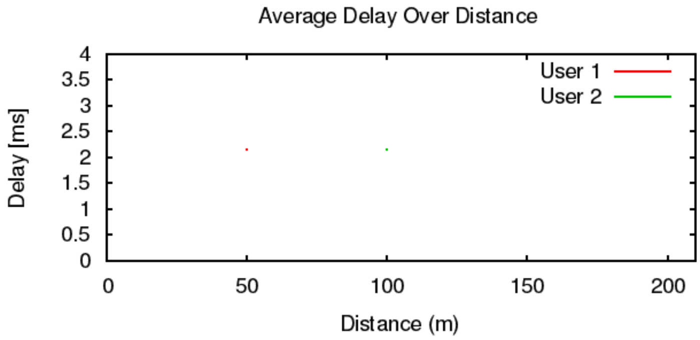

Figure 11: Delay for 2 users, one at a distance of 50m, one at a distance of 100m

While there is a minor difference in packet loss ratio between user 1 and user 2, this difference is so small it cannot be shown on the output graph. However, the difference shown in the equations below is based off the database file, which shows that there is one packet lost in the transmission between the access point and the second user.

$$
\begin{aligned}
PLR &= \frac{lostPackets}{rxPackets + lostPackets} \\
PLR_{user1} &= \frac{0}{5000+0} \\
&= 0 \\
PLR_{user2} &= 14999 + 1 \\
&= 0.0002
\end{aligned}
$$

Equation 14: Average Packet Loss Ratio

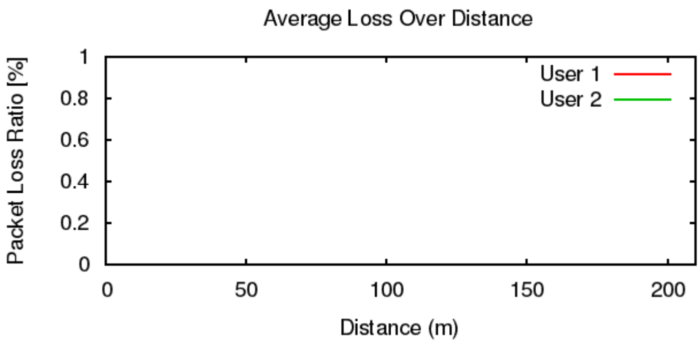

Figure 12: Loss for 2 users, one at a distance of 50m, one at a distance of 100m

1.3.2 Part 2:

This section introduces larger numbers of users, testing with 1, 5, 10, 20, and 50 users. This testing was implemented, as with every section, using a bash script, which can be found in the Appendices. As the loop implemented in the previous section allows for any amount of users to be input, the bash script simply needs to execute the simulation for each of the desired number of users.

Listing 3: wifi-p1qcp2.sh File Snippet

```
USERS="1 5 10 20 50"

#Code Removed for purposes of snippet

  for user in $USERS
   do
    echo Trial $trial , Users $user
      ../../waf --run "WiFi --users=$user --dbPrefix=
          $DBPREFIX --txInterval=$TXINTERVAL --run=run-
          $trial-$DBPREFIX-$user"
   done
```

The throughput can be seen to be relatively robust for each number of users, with each case having approximately 1Mbps throughput. The 20 and 50 user systems have slightly less than 1Mbps, however, as the difference is minor, it is not easily visible in the plot.

$$Throughput = \frac{rxPackets * 1000 * 8}{txTime}$$
$$TP_{1-10Users} = 1000Kbps$$
$$TP_{20Users} = \frac{1686.45 * 1000 * 8}{20}$$
$$= 647.58Kbps(\text{using average number of rxPackets})$$
$$TP_{20Users} = 989.92Kbps(\text{using Bash Script})$$
$$TP_{50User} = 989.92Kbps(\text{using Bash Script})$$

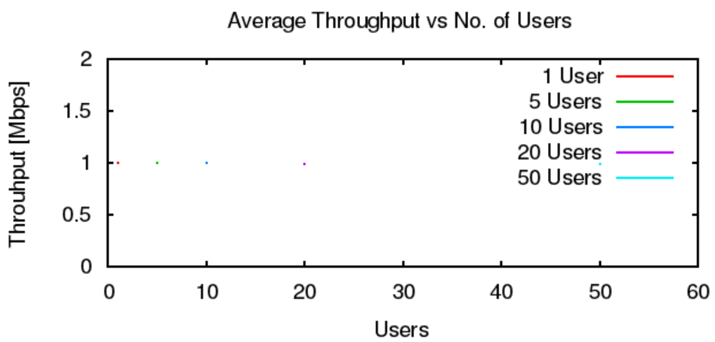

Figure 13: Throughput for systems with 0 to 50 users

The delay increases from a value of 0.44ms in a single user system to approximately 1530ms for a 50 user system. This can be seen in Figure 14. The equations below show the values as extracted from the .db file using the bash script.

$$\overline{delay} = \frac{delaySum}{rxPackets}$$
$$\overline{delay}_{1User} = 441013ns$$
$$\overline{delay}_{5User} = 2807059ns$$
$$\overline{delay}_{10User} = 8420910ns$$
$$\overline{delay}_{20User} = 277970000ns$$
$$\overline{delay}_{50User} = 1530570000ns$$

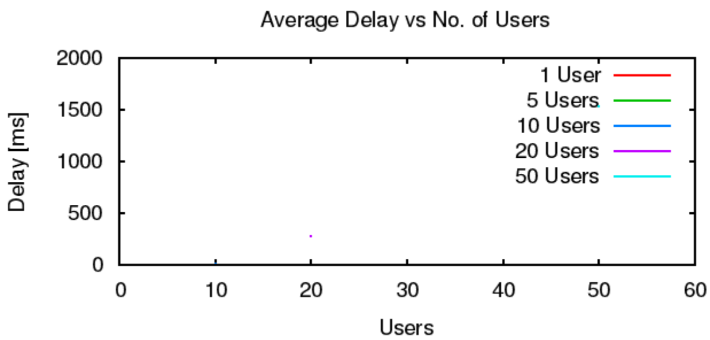

Figure 14: Delay for systems with 0 to 50 users

The systems relative robustness can also be shown through its packet loss ratio values for all numbers of users. For 1 to 10 users, there are no lost packets. For the 20 and 50 user systems, there is a PLR of only 1.08%, as extracted from the .db file.

$$PLR = \frac{lostPackets}{rxPackets + lostPackets}$$
$$PLR_{1User} = 0$$
$$PLR_{5User} = 0$$
$$PLR_{10User} = 0$$
$$PLR_{20User} = 1.08\%$$
$$PLR_{50User} = 1.08\%$$

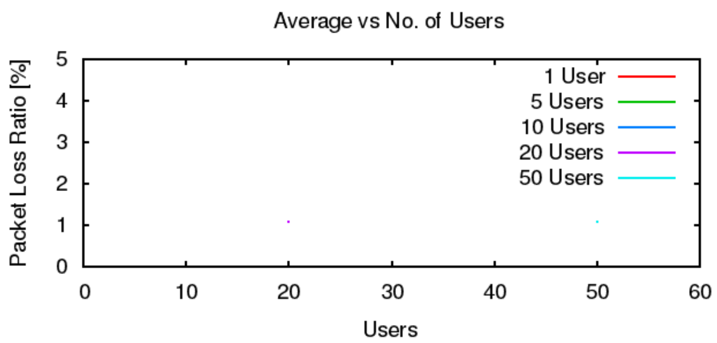

Figure 15: Loss for systems with 0 to 50 users

1.3.3 Part 3:

This section again uses larger numbers of users, from 1 to 20, and distances the users from 0 to 150 meters from the access point.
Due to the large number of outputs from this section, tables, such as that of Table 4 are used to show the extracted values from the .db file.

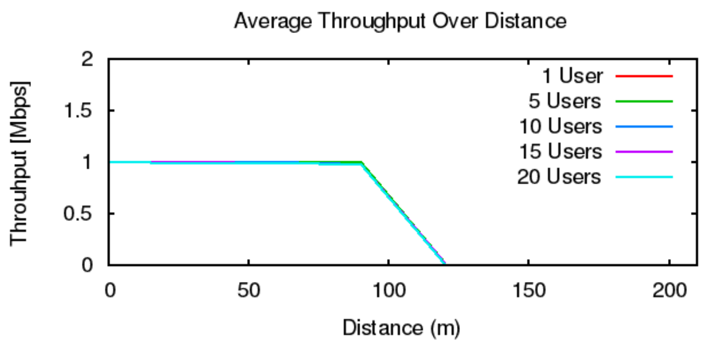

Figure 16: Throughput for systems with 0 to 20 users, over distances from 0 to 150 meters

The table below shows that for a single user, there is no change in throughput from 0 to 90 meters, with a large drop at 120 meters. Similarly, for 5 users, there is only a drop in throughput at a distance of 120 meters.
For 10 users, there is a small decrease in throughput at 90 meters, with a similar large decrease as before at 120 meters. The 15 user system has a drop in throughput at 60 meters, at 90 meters, and a larger drop again at 120 meters.
For 20 users there is an immediate drop at 30 meters, with a slow decrease in throughput between 30 and 90 meters. Again there is a more dramatic decrease in throughput at 120 meters.

Table 2: Throughput Values (Kbps) for 1-20 User Systems at Distances of 0-150m

Distance	1 User	5 Users	10 Users	15 Users	20 Users
0	1000	1000	1000	1000	1000
30	1000	1000	1000	1000	990.8
60	1000	1000	1000	986	986
90	1000	1000	974.8	975.6	975.6
120	17.6	18.4	18.4	17.6	15.2

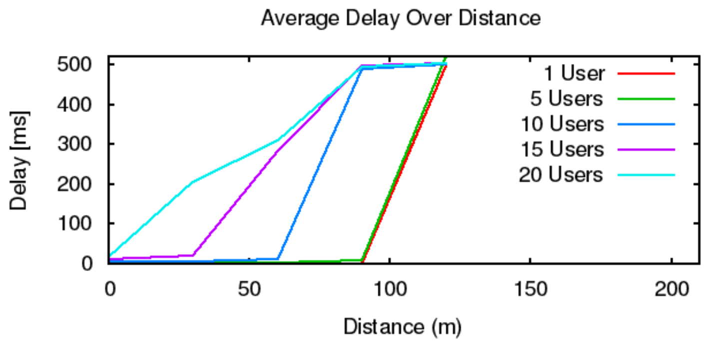

Figure 17: Delay for systems with 0 to 20 users, over distances from 0 to 150 meters

For delay, the values have a sharp increase in most cases, for example the single and five user systems see a large increase at a distance of 120 meters. The 20 user system however has an initial large increase between 0 and 30 meters, followed by a smaller increase between 30 and 60 meters, followed again by a large increase between 60 and 90 meters. The final delay value in all cases is quite similar.

Table 3: Delay Values (ms) for 1-20 User Systems at Distances of 0-150m

Distance	1 User	5 Users	10 Users	15 Users	20 Users
0	0.194653	1.57338	5.6031	11.2192	18.5429
30	0.347009	2.32888	7.16022	19.1675	203.886
60	0.566218	3.44795	12.5859	283.532	310.421
90	1.27689	8.00458	488.852	496.113	494.484
120	497.965	523.288	499.474	504.065	504.108

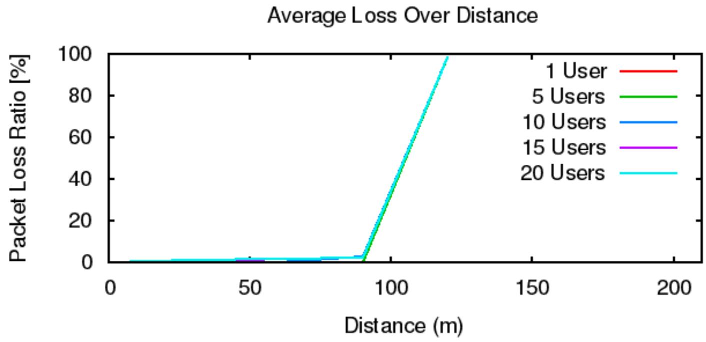

Figure 18: Loss for systems with 0 to 20 users, over distances from 0 to 150 meters

Loss can be seen to be similar to the throughput results (due to their dependence on the number of received packets). All systems have a PLR of approximately 98% at a distance of 120 meters. As the number of users increases, the distance at which loss begins to occur decreases, for example with 10 users, loss occurs at 90 meters, whereas with 20 users loss occurs at 30 meters.

Table 4: Packet Loss Ratio for 1-20 User Systems at Distances of 0-150m

Distance	1 User	5 Users	10 Users	15 Users	20 Users
0	0	0	0	0	0
30	0	0	0	0	0.92
60	0	0	0	1.4	1.4
90	0	0	2.52	2.44	2.44
120	98.24	98.16	98.16	98.24	98.48

From this section, it can be seen that the distance has a greater affect on the performance characteristics of the system than the number of users, however the affect of the number of users cannot be discounted

2 Part 2: Results and Comparison and Analysis

Each section of this assignment deals with three of the main performance characteristics of a network, namely throughput, delay, and loss. These values are affected by the number of users in the network, the distance between the nodes in the network, and the bitrate. Question A deals with the differences in bitrates, Question B with the differences in distances, and Question C Part 2 with the number of users in the network.

2.1 Throughput

For Question A Part 1, there is a throughput value of 1.6Kbps. This network has a single user (sink) node, and a single access point (source) node. The bitrate is determined in this case by the 0.05 second transmission interval.
Question A Part 2 investigates the throughput due to different bitrates. With a bitrate of 1Mbps, there is a throughput of 1Mbps. A 1.5Mbps bitrate corresponds to a 1.5Mbps

throughput, a 5Mbps bitrate to a 5Mbps throughput, and a 10Mbps bitrate to an approximately 10Mbps throughput. At a bitrate of 20Mbps however, there is a throughput of only 13.7Mbps, due to the loss of packets.
Question B investigates the throughput due to different distances, using the initial setup from Question A Part 1. At distances between 0 and 110 meters, there is a throughput of 160Kbps. At 120 meters, due to lost packets, there is a throughput of 5.2Kbps, and between 130 and 150 meters, no packets are received at the sink node, giving a throughput of 0Kbps.
Question C Part 2 investigates the throughput with an increased number of users within the network. For 1 to 10 sink nodes, there is no effect on the throughput, giving a value of 1Mbps. With 20 and 50 sink nodes, there is a small drop in throughput, with a value of 0.98992Mbps.

Table 5: Throughput Values for QA Part 1 and 2, QB, and QC Part 2

Throughput (Kbps)	
QA Part 1	160
QA $P2_{1Mbps}$	1000
QA $P2_{1.5Mbps}$	1500
QA $P2_{5Mbps}$	5000
QA $P2_{10Mbps}$	9999
QA $P2_{20Mbps}$	13747
QB_{0-110m}	160
QB_{120}	5.2
$QB_{130-150}$	0
$QC_{1-10Users}$	1000
$QC_{20-50Users}$	989.92

This table shows that attempting to increase the throughput by increasing the bitrate will only work to a certain point, at which packet loss will begin to occur, causing the throughput to drop below the defined bitrate.
The table also shows that distance between nodes has a much greater affect on the throughput than the number of users within the network.

2.2 Delay

For Question A Part 1, the single source, single sink network has a delay value of 0.49ms.
Question A Part 2 investigates the delay due to different bitrates. With a bitrate of 1Mbps, there is a delay of 0.44ms. A 1.5Mbps bitrate corresponds to a delay of 0.43ms, a 5Mbps bitrate to a delay of 0.438ms, and a 10Mbps bitrate to a delay of 0.497ms. At a bitrate of 20Mbps however, due to packet loss, there is a delay of 229ms.
Question B investigates the delay due to different distances, using the initial setup from Question A Part 1. The delay steadily increases for distances between 0 and 70 meters, from 0.27ms to 0.65ms. At 80 meters there is a steeper increase, to a delay of 0.84ms. Following this there is again a steady increase to a delay of 1.72ms at 110 meters, with a final spike to 9.3ms visible at 120 meters. Following 120 meters there is no delay, as no packets are being received.
Question C Part 2 investigates the delay with an increased number of users within the

network. For a single user, the delay is 0.44ms. With 5 users, this delay increases to 2.8ms. With 10 users, the delay increases to 8.4ms. With 20 users there is a delay of 227.9ms, and finally, with 50 users there is a delay of 1530.5ms.

Table 6: Delay Values for QA Part 1 and 2, QB, and QC Part 2

Delay (ms)	
QA Part 1	0.44
QA $P2_{1Mbps}$	0.44
QA $P2_{1.5Mbps}$	0.43
QA $P2_{5Mbps}$	0.43
QA $P2_{10Mbps}$	0.49
QA $P2_{20Mbps}$	229
QB	Refer to Equation 10
QB	MaxDelay = 9.36
QC_{1User}	0.44
QC_{5User}	2.8
QC_{10User}	8.4
QC_{20User}	227.9
QC_{50User}	1530.5

This table shows that distance between nodes has a much less of an affect on the delay than the number of users within the network. This is converse to the throughput, which was affected more by the distance.
With 50 users, the delay is approximately 160 times greater than the maximum delay within Question B in which the distance was being increased.

2.3 Loss

For Question A Part 1, the single source, single sink network has a Packet Loss Ratio (PLR) of 0% as no packets are lost.
Question A Part 2 investigates the PLR due to different bitrates. No packet loss occurs for bitrates between 1 and 5Mbps, giving a PLR of 0%. At 10Mbps there are 3 lost packets, giving a PLR of 1.2^{-5}%. A bitrate of 20Mbps corresponds to a 30% PLR.
Question B investigates the loss due to different distances, using the initial setup from Question A Part 1. For distances between 0 and 110 meters no packet loss occurs giving a PLR of 0%. At 120 meters, there is a PLR of 96.75%, due to 387 of 400 packets being lost. Between 130 and 150 meters, no packets were received, giving a PLR of 100%.
Question C Part 2 investigates the loss with an increased number of users within the network. For 1, 5, and 10 Sinks, there are no lost packets, giving a PLR of 0%. For 20 and 50 sinks, there is a packet loss ratio of 1.08%.

Table 7: Packet Loss Ratio Values for QA Part 1 and 2, QB, and QC Part 2

Packet Loss (%)	
QA Part 1	0
QA $P2_{1Mbps}$	0
QA $P2_{1.5Mbps}$	0
QA $P2_{5Mbps}$	0
QA $P2_{10Mbps}$	1.2×10^{-5}
QA $P2_{20Mbps}$	30.7
QB_{0-110m}	0
QB_{120}	96.75
$QB_{130-150}$	100
$QC_{1-10Users}$	0
$QC_{20-50Users}$	1.08

Similarly to Table 5, the loss is affected much more by the distance between the nodes than by the number of nodes. Much higher loss is experienced at larger distances than with larger numbers of nodes in the network.

2.4 Conclusion

From the previous comparisons, it can be seen that both throughput and loss are affected more by distance than by the numbers of users in the system, whereas delay is affected much more by the number of users than by the distance to the source node.
As such, if throughput and loss mitigation are more important characteristics in the network, then users (sink nodes) should be kept within close proximity of the source node, at a maximum distance of 120 meters. If delay is the more important characteristic, then the number of sink nodes should be limited, as over 10 users tends to dramatically increase the delay within the system.

3 Appendix

For access to the .db files go to `https://github.com/mLenehan1/EE500-NetworkPerformance`

3.1 Bash Scripts

The following bash scripts were used to execute the simulations in ns-3, to save the outputs to .db files, and to plot the outputs from the simulation.

3.1.1 Question A Part 1 Bash Script

```
#!/bin/sh

TRIALS="1"
DBPREFIX="P1QAP1"
SIMTIME=20 #Default in wifi-example-sim.cc: The
    transmission Time approximately equals to the
    Simulation Running Time

echo WiFi Experiment Example

pCheck=`which sqlite3`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sqlite3 (wifi-example
     -sim does not)."
  exit 255
fi

pCheck=`which gnuplot`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires gnuplot (wifi-example
     -sim does not)."
  exit 255
fi

pCheck=`which sed`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sed (wifi-example-sim
       does not)."
  exit 255
fi

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:bin/

if [ -e ../../P1QAP1.db ]
then
```

```
  echo "Kill data.db? (y/n)"
  read ANS
  if [ "$ANS" = "yes" -o "$ANS" = "y" ]
  then
    echo Deleting database
    rm ../../P1QAP1.db
  fi
fi

for trial in $TRIALS
do
  echo Trial $trial
  ../../waf --run "WiFi --dbPrefix=$DBPREFIX --run=run-
     $DBPREFIX-$trial"
done

#
#Another SQL command which just collects raw numbers of
   frames receved.
#
#CMD="select Experiments.input,avg(Singletons.value) \
#     from Singletons,Experiments \
#     where Singletons.run = Experiments.run AND \
#           Singletons.name='wifi-rx-frames' \
#     group by Experiments.input \
#     order by abs(Experiments.input) ASC;"

mv ../../P1QAP1.db .

CMD="select exp.input, tx.value, rx.value, delay.value \
     from Singletons rx, Singletons tx, Experiments exp,
        Singletons delay \
     where rx.run = tx.run AND \
           rx.run = exp.run AND \
           delay.run = exp.run AND \
           rx.variable='receiver-rx-packets' AND \
           tx.variable='sender-tx-packets' AND \
           delay.variable='delay-average' \
     group by exp.input \
     order by abs(exp.input) ASC;"

sqlite3 -noheader P1QAP1.db "$CMD" > wifi-default.temp
sed -i "s/|/ /g" wifi-default.temp
awk '{print $1 " " $4*8*1000/20/1000/1000}' wifi-default.
   temp >throughput.data
awk '{print $1 " " $5/1000000}' wifi-default.temp >delay.
   data
awk '{print $1 " " ($3-$4)/$3*100}' wifi-default.temp >
   loss.data
```

```
gnuplot wifi-p1qap1.gnuplot

#Clean Directory
mv *.data *.db *.temp QA/P1/Data
mv *.png QA/P1/Images
echo "Done; data in wifi-default.data, plot in wifi-
    default.eps"
```

3.1.2 Question A Part 2 Bash Script

```
#!/bin/sh

TRIALS="1"
DBPREFIX="P1QAP2"
TXINTERVAL="0.008 0.0053 0.0016 0.0008 0.0004"

SIMTIME=20 #Default in wifi-example-sim.cc: The
    transmission Time approximately equals to the
    Simulation Running Time

echo WiFi Experiment Example

pCheck=`which sqlite3`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sqlite3 (wifi-example
     -sim does not)."
  exit 255
fi

pCheck=`which gnuplot`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires gnuplot (wifi-example
     -sim does not)."
  exit 255
fi

pCheck=`which sed`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sed (wifi-example-sim
      does not)."
  exit 255
fi

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:bin/

if [ -e ../../P1QAP2.db ]
then
```

```
  echo "Kill data.db? (y/n)"
  read ANS
  if [ "$ANS" = "yes" -o "$ANS" = "y" ]
  then
    echo Deleting database
    rm ../../P1QAP2.db
  fi
fi

for trial in $TRIALS
do
  for txinterval in $TXINTERVAL
  do
    echo Trial $trial, txInterval $txinterval
    ../../waf --run "WiFi --dbPrefix=$DBPREFIX --
        txInterval=$txinterval --run=run-$trial-$DBPREFIX-
        $txinterval"
  done
done

#
#Another SQL command which just collects raw numbers of
    frames receved.
#
#CMD="select Experiments.input,avg(Singletons.value) \
#     from Singletons,Experiments \
#     where Singletons.run = Experiments.run AND \
#           Singletons.name='wifi-rx-frames' \
#     group by Experiments.input \
#     order by abs(Experiments.input) ASC;"

mv ../../P1QAP2.db .

CMD="select exp.input, tx.value, rx.value, delay.value \
     from Singletons rx, Singletons tx, Experiments exp,
         Singletons delay \
     where rx.run = tx.run AND \
           rx.run = exp.run AND \
           delay.run = exp.run AND \
           rx.variable='receiver-rx-packets' AND \
           tx.variable='sender-tx-packets' AND \
           delay.variable='delay-average' \
     group by exp.input \
     order by abs(exp.input) ASC;"



sqlite3 -noheader P1QAP2.db "$CMD" > wifi-default.temp
sed -i "s/|/ /g" wifi-default.temp
awk '{print $1 " " $4*8*1000/20/1000/1000}' wifi-default.
```

```
    temp >throughput.data
awk '{print $1 " " $5/1000000}' wifi-default.temp >delay.
    data
awk '{print $1 " " ($3-$4)/$3*100}' wifi-default.temp >
    loss.data

gnuplot wifi-p1qap2.gnuplot

#Clean Directory
mv *.data *.db *.temp QA/P2/Data
mv *.png QA/P2/Images
echo "Done; data in wifi-default.data, plot in wifi-
    default.eps"
```

3.1.3 Question B Bash Script

```
#!/bin/sh

DISTANCES="0 10 20 30 40 50 60 70 80 90 100 110 120 130
    140 150"
DBPREFIX="P1QB"
TRIALS="1"
SIMTIME=20 #Default in wifi-example-sim.cc: The
    transmission Time approximately equals to the
    Simulation Running Time

echo WiFi Experiment Example

pCheck=`which sqlite3`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sqlite3 (wifi-example
     -sim does not)."
  exit 255
fi

pCheck=`which gnuplot`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires gnuplot (wifi-example
     -sim does not)."
  exit 255
fi

pCheck=`which sed`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sed (wifi-example-sim
       does not)."
  exit 255
```

```
fi

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:bin/

if [ -e ../../P1QB.db ]
then
  echo "Kill data.db? (y/n)"
  read ANS
  if [ "$ANS" = "yes" -o "$ANS" = "y" ]
  then
    echo Deleting database
    rm ../../P1QB.db
  fi
fi

for trial in $TRIALS
do
  for distance in $DISTANCES
  do
    echo Trial $trial, distance $distance
    ../../waf --run "WiFi --distance=$distance --dbPrefix
        =$DBPREFIX --run=run-$trial-$DBPREFIX-$distance"
  done
done

#
#Another SQL command which just collects raw numbers of
    frames receved.
#
#CMD="select Experiments.input,avg(Singletons.value) \
#     from Singletons,Experiments \
#     where Singletons.run = Experiments.run AND \
#           Singletons.name='wifi-rx-frames' \
#     group by Experiments.input \
#     order by abs(Experiments.input) ASC;"

mv ../../P1QB.db .

CMD="select exp.input, tx.value, rx.value, delay.value \
     from Singletons rx, Singletons tx, Experiments exp,
         Singletons delay \
     where rx.run = tx.run AND \
           rx.run = exp.run AND \
           delay.run = exp.run AND \
           rx.variable='receiver-rx-packets' AND \
           tx.variable='sender-tx-packets' AND \
           delay.variable='delay-average' \
     group by exp.input \
     order by abs(exp.input) ASC;"
```

```
sqlite3 -noheader P1QB.db "$CMD" > wifi-default.temp
sed -i "s/|/ /g" wifi-default.temp
awk '{print $1 " " $4*8*1000/20/1000/1000}' wifi-default.
    temp >throughput.data
awk '{print $1 " " $5/1000000}' wifi-default.temp >delay.
    data
awk '{print $1 " " ($3-$4)/$3*100}' wifi-default.temp >
    loss.data

gnuplot wifi-p1qb.gnuplot

#Clean Directory
mv *.data *.db *.temp QB/Data
mv *.png QB/Images
echo "Done; data in wifi-default.data, plot in wifi-
    default.eps"
```

3.1.4 Question C Part 1 Bash Script

```
#!/bin/sh

DISTANCE=50
DBPREFIX="P1QCP1"
TXINTERVAL="0.004"
USERS="2"
TRIALS="1"
SIMTIME=20 #Default in wifi-example-sim.cc: The
    transmission Time approximately equals to the
    Simulation Running Time

echo WiFi Experiment Example

pCheck=`which sqlite3`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sqlite3 (wifi-example
      -sim does not)."
  exit 255
fi

pCheck=`which gnuplot`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires gnuplot (wifi-example
      -sim does not)."
  exit 255
fi
```

```
pCheck=`which sed`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sed (wifi-example-sim
       does not)."
  exit 255
fi

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:bin/

if [ -e ../../P1QCP1.db ]
then
  echo "Kill data.db? (y/n)"
  read ANS
  if [ "$ANS" = "yes" -o "$ANS" = "y" ]
  then
    echo Deleting database
    rm ../../P1QCP1.db
  fi
fi

for trial in $TRIALS
do
  echo Trial $trial, user 1 distance $DISTANCE
  ../../waf --run "WiFi --users=$USERS --distance=
      $DISTANCE --dbPrefix=$DBPREFIX --txInterval=
      $TXINTERVAL --run=run-$trial-$DBPREFIX-$USERS"
done

#
#Another SQL command which just collects raw numbers of
    frames receved.
#
#CMD="select Experiments.input,avg(Singletons.value) \
#     from Singletons,Experiments \
#     where Singletons.run = Experiments.run AND \
#           Singletons.name='wifi-rx-frames' \
#     group by Experiments.input \
#     order by abs(Experiments.input) ASC;"

mv ../../P1QCP1.db .

CMD="select exp.input, tx.value, rx.value, delay.value \
     from Singletons rx, Singletons tx, Experiments exp,
         Singletons delay \
     where rx.run = tx.run AND \
           rx.run = exp.run AND \
           delay.run = exp.run AND \
           rx.variable='receiver-rx-packets' AND \
           tx.variable='sender-tx-packets' AND \
```

```
        delay.variable='delay-average' \
    group by exp.input \
    order by abs(exp.input) ASC;"


sqlite3 -noheader P1QCP1.db "$CMD" > wifi-default.temp
sed -i "s/|/ /g" wifi-default.temp
awk '{if($2=="user1") {print $1 " " $4
    *8*1000/20/1000/1000 > "usr1TP.data"}
 else if($2=="user2") {print 100 " " $4
     *8*1000/20/1000/1000 > "usr2TP.data"}
 else {print $1 " " $4*8*1000/20/1000/1000 > "throughput.
     data"}
}' wifi-default.temp
awk '{if($2=="user1") {print $1 " " $5/1000000 > "
    usr1delay.data"}
 else if($2=="user2") {print 100 " " $5/1000000 > "
     usr2delay.data"}
 else {print $1 " " $5/1000000 > "delay.data"}
}' wifi-default.temp
awk '{if($2=="user1") {print $1 " " ($3-$4)/$3*100 > "
    usr1loss.data"}
 else if($2=="user2") {print 100 " " ($3-$4)/$3*100 > "
     usr2loss.data"}
 else {print $1 " " ($3-$4)/$3*100 > "loss.data"}
}' wifi-default.temp

gnuplot wifi-p1qcp1.gnuplot

#Clean Directory
mv *.data *.db *.temp QC/P1/Data
mv *.png QC/P1/Images
echo "Done; data in wifi-default.data, plot in wifi-
    default.eps"
```

3.1.5 Question C Part 2 Bash Script

```
#!/bin/sh

DISTANCES="50"
DBPREFIX="P1QCP2"
TXINTERVAL="0.008"
USERS="1 5 10 20 50"
TRIALS="1"
SIMTIME=20 #Default in wifi-example-sim.cc: The
    transmission Time approximately equals to the
    Simulation Running Time

echo WiFi Experiment Example
```

```
pCheck=`which sqlite3`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sqlite3 (wifi-example
      -sim does not)."
  exit 255
fi

pCheck=`which gnuplot`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires gnuplot (wifi-example
      -sim does not)."
  exit 255
fi

pCheck=`which sed`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sed (wifi-example-sim
       does not)."
  exit 255
fi

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:bin/

if [ -e ../../P1QCP2.db ]
then
  echo "Kill data.db? (y/n)"
  read ANS
  if [ "$ANS" = "yes" -o "$ANS" = "y" ]
  then
    echo Deleting database
    rm ../../P1QCP2.db
  fi
fi

for trial in $TRIALS
do
    for user in $USERS
    do
      echo Trial $trial, Users $user
      ../../waf --run "WiFi --users=$user --dbPrefix=
          $DBPREFIX --txInterval=$TXINTERVAL --run=run-
          $trial-$DBPREFIX-$user"
    done
done

#
```

```
#Another SQL command which just collects raw numbers of
    frames receved.
#
#CMD="select Experiments.input,avg(Singletons.value) \
#     from Singletons,Experiments \
#     where Singletons.run = Experiments.run AND \
#           Singletons.name='wifi-rx-frames' \
#     group by Experiments.input \
#     order by abs(Experiments.input) ASC;"

mv ../../P1QCP2.db .

CMD="select exp.input, tx.value, rx.value, delay.value \
     from Singletons rx, Singletons tx, Experiments exp,
         Singletons delay \
     where rx.run = tx.run AND \
           rx.run = exp.run AND \
           delay.run = exp.run AND \
           rx.variable='receiver-rx-packets' AND \
           tx.variable='sender-tx-packets' AND \
           delay.variable='delay-average' \
     group by exp.input \
     order by abs(exp.input) ASC;"



sqlite3 -noheader P1QCP2.db "$CMD" > wifi-default.temp
sed -i "s/|/ /g" wifi-default.temp
awk '{print $1 " " $4*8*1000/20/1000/1000 > $1"usrTP.data
    "}' wifi-default.temp
awk '{print $1 " " $5/1000000 > $1"usrdelay.data"}' wifi-
    default.temp
awk '{print $1 " " ($3-$4)/$3*100 > $1"usrloss.data"}'
    wifi-default.temp

gnuplot wifi-p1qcp2.gnuplot

#Clean Directory
mv *.data *.db *.temp QC/P2/Data
mv *.png QC/P2/Images
echo "Done; data in wifi-default.data, plot in wifi-
    default.eps"
```

3.1.6 Question C Part 3 Bash Script

```
#!/bin/sh

DISTANCES="0 30 60 90 120 150"
DBPREFIX="P1QCP3"
TXINTERVAL="0.008"
```

```
USERS="1 5 10 15 20"
TRIALS="1"
SIMTIME=20 #Default in wifi-example-sim.cc: The
    transmission Time approximately equals to the
    Simulation Running Time

echo WiFi Experiment Example

pCheck=`which sqlite3`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sqlite3 (wifi-example
     -sim does not)."
  exit 255
fi

pCheck=`which gnuplot`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires gnuplot (wifi-example
     -sim does not)."
  exit 255
fi

pCheck=`which sed`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sed (wifi-example-sim
      does not)."
  exit 255
fi

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:bin/

if [ -e ../../P1QCP3.db ]
then
  echo "Kill data.db? (y/n)"
  read ANS
  if [ "$ANS" = "yes" -o "$ANS" = "y" ]
  then
    echo Deleting database
    rm ../../P1QCP3.db
  fi
fi

for trial in $TRIALS
do
  for user in $USERS
  do
    for distance in $DISTANCES
```

```
      do
        echo Trial $trial , Users $user , distance $distance
        ../../waf --run "WiFi --distance=$distance --users=
            $user --dbPrefix=$DBPREFIX --txInterval=
            $TXINTERVAL --run=run-$trial-$DBPREFIX-$user-
            $distance"
      done
    done
done

#
#Another SQL command which just collects raw numbers of
    frames receved.
#
#CMD="select Experiments.input,avg(Singletons.value) \
#     from Singletons,Experiments \
#     where Singletons.run = Experiments.run AND \
#           Singletons.name='wifi-rx-frames' \
#     group by Experiments.input \
#     order by abs(Experiments.input) ASC;"

mv ../../P1QCP3.db .

CMD="select exp.input, tx.value, rx.value, delay.value \
     from Singletons rx, Singletons tx, Experiments exp,
        Singletons delay \
     where rx.run = tx.run AND \
           rx.run = exp.run AND \
           delay.run = exp.run AND \
           rx.variable='receiver-rx-packets' AND \
           tx.variable='sender-tx-packets' AND \
           delay.variable='delay-average' \
     group by exp.input \
     order by abs(exp.input) ASC;"



sqlite3 -noheader P1QCP3.db "$CMD" > wifi-default.temp
sed -i "s/|/ /g" wifi-default.temp
awk '{print $1 " " $5*8*1000/20/1000/1000 > $2"usrTP.data
    "}' wifi-default.temp
awk '{print $1 " " $6/1000000 > $2"usrdelay.data"}' wifi-
    default.temp
awk '{print $1 " " ($4-$5)/$4*100 > $2"usrloss.data"}'
    wifi-default.temp

gnuplot wifi-p1qcp3.gnuplot

#Clean Directory
mv *.data *.db *.temp QC/P3/Data
```

```
mv *.png QC/P3/Images
echo "Done; data in wifi-default.data, plot in wifi-
    default.eps"
```

3.2 C++ Files

The following script is a modified version of the provided code required to execute the simulation within ns-3.

3.2.1 C++ Simulation File

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil
    ; -*- */
/*
 * This program is free software; you can redistribute it
     and/or modify
 * it under the terms of the GNU General Public License
    version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will
    be useful,
 * but WITHOUT ANY WARRANTY; without even the implied
    warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General
    Public License
 * along with this program; if not, write to the Free
    Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston,
    MA  02111-1307  USA
 *
 * Authors: Joe Kopena <tjkopena@cs.drexel.edu>
 * Modified by: Longhao Zou, Oct 2016 for EE500 <longhao.
    zou@dcu.ie>
 *

              EE500 Assignment 2016-2017
              Default WiFi Network Topology

                 WiFi 192.168.0.0
              ------------------------
              |AP (node 0:192.168.0.1)|
              ------------------------
                *          *            *
               /           |             \
   Traffic 1/   Traffic 2|    ------     \ Traffic N
```

```
          /              |                 \
       user 1        user 2      -------    user N
    (node 1          (node 2      -------   (node N
    :192.168.0.2     :192.168.0.3 -------   :192.168.0.N+1
    :1000)           :1001)         -------:1000+(N-1))

    PS: In this example, I just implemented only 1 WiFi
        user.

  */

#include <ctime>

#include <sstream>

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"

#include "ns3/stats-module.h"

#include "wifi-example-apps.h"

using namespace ns3;
using namespace std;

NS_LOG_COMPONENT_DEFINE ("WiFiExampleSim");

void TxCallback (Ptr<CounterCalculator<uint32_t> > datac,
                 std::string path, Ptr<const Packet>
                    packet) {
  NS_LOG_INFO ("Sent frame counted in " <<
               datac->GetKey ());
  datac->Update ();
  // end TxCallback
}

//
   ----------------------------------------------------------------------

//-- main
//----------------------------------------------------------------
int main (int argc, char *argv[]) {
  //LogComponentEnable ("WiFiDistanceApps",
     LOG_LEVEL_INFO);
  double distance = 50.0;
  double distance2 = 100.0;
```

```
double simTime = 20; //Simulation Running Time (in
    seconds)
int users = 1;
string txInterval = "0.05";
string dbPrefix = "DBFile";
string format ("db"); //Default as .db format (Please
    refer to sqlite-data-output.cc and sqlite-data-
    output.h located in /src/stats/model)

string experiment ("wifi-example-sim"); //the name of
    your experiment
string strategy ("wifi-default");
string input;
string runID;

{
  stringstream sstr;
  sstr << "run-" << time (NULL);
  runID = sstr.str ();
}

// Set up command line parameters used to control the
    experiment.
CommandLine cmd;
cmd.AddValue ("distance", "Distance apart to place
    nodes (in meters).",
              distance);
cmd.AddValue ("txInterval", "Interval between packet
    trasmissions (in seconds).", txInterval);
cmd.AddValue ("users", "Number of Users", users);
cmd.AddValue ("dbPrefix", "Database File Name",
    dbPrefix);
//cmd.AddValue ("format", "Format to use for data
    output.",
  //            format);
cmd.AddValue ("simTime", "Simulation Running Time (in
    seconds)", simTime);
cmd.AddValue ("experiment", "Identifier for experiment.
    ",
              experiment);
cmd.AddValue ("strategy", "Identifier for strategy.",
              strategy);
cmd.AddValue ("run", "Identifier for run.",
              runID);
cmd.Parse (argc, argv);

if (format != "omnet" && format != "db") {
     NS_LOG_ERROR ("Unknown output format '" << format
         << "'");
     return -1;
```

```
  }

#ifndef STATS_HAS_SQLITE3
if (format == "db") {
    NS_LOG_ERROR ("sqlite support not compiled in.");
    return -1;
  }
#endif

{
  stringstream sstr ("");
  if(dbPrefix.compare("P1QAP2")==0){
      sstr << txInterval;
  } else if(dbPrefix.compare("P1QCP3")==0){
      sstr << distance;
      sstr << " " << users;
  } else if(dbPrefix.compare("P1QCP2")==0){
      sstr << users;
  } else{
      sstr << distance;
  }
  input = sstr.str ();
}

//
   ----------------------------------------------------------------

//-- Create nodes and network stacks
//--------------------------------------------------
NS_LOG_INFO ("Creating nodes.");
NodeContainer nodes;
nodes.Create (users+1);

NS_LOG_INFO ("Installing WiFi and Internet stack.");
WifiHelper wifi = WifiHelper::Default ();
NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default
    ();
wifiMac.SetType ("ns3::AdhocWifiMac");
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default
    ();
YansWifiChannelHelper wifiChannel =
    YansWifiChannelHelper::Default ();
wifiPhy.SetChannel (wifiChannel.Create ());
NetDeviceContainer nodeDevices = wifi.Install (wifiPhy,
     wifiMac, nodes);

InternetStackHelper internet;
internet.Install (nodes);
Ipv4AddressHelper ipAddrs;
ipAddrs.SetBase ("192.168.0.0", "255.255.255.0");
```

```
ipAddrs.Assign (nodeDevices);

//
    ------------------------------------------------------------------------

//-- Setup physical layout
//----------------------------------------------------
NS_LOG_INFO ("Installing static mobility; distance " <<
     distance << " .");
MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc =
  CreateObject<ListPositionAllocator>();
positionAlloc->Add (Vector (0.0, 0.0, 0.0));
if(dbPrefix.compare("P1QCP1")!=0){
  for(int i=0; i<users; i++){
    positionAlloc->Add (Vector (0.0, distance, 0.0));
  }
} else {
  positionAlloc->Add (Vector (0.0, distance, 0.0));
  positionAlloc->Add (Vector (0.0, distance2, 0.0));
}
mobility.SetPositionAllocator (positionAlloc);
mobility.Install (nodes);

//
    ------------------------------------------------------------------------

//-- Create the traffic between AP and WiFi Users
//
    ------------------------------------------------------------------------

//
    ------------------------------------------------------------------------

//-- 1. Create the first traffic for the first WiFi
    user on WiFi AP (source)
//----------------------------------------------------
NS_LOG_INFO ("Create traffic source & sink.");
Ptr<Sender> sender[users];
Ptr<Node> appSource = NodeList::GetNode (0);
for(int i=0; i<users; i++){
  sender[i] = CreateObject<Sender>();
  sender[i]->SetAttribute("Port", UintegerValue(1000+i)
      );//Lisening Port of the first WiFi user
  sender[i]->SetAttribute("PacketSize", UintegerValue
      (1000)); //bytes
  sender[i]->SetAttribute("Interval", StringValue ("ns3
      ::ConstantRandomVariable[Constant=" + txInterval +
       "]")); //seconds
```

```
    sender[i]->SetAttribute("NumPackets",UintegerValue
        (100000000));
    appSource->AddApplication (sender[i]);
    sender[i]->SetStartTime (Seconds (0));
}

//
    ------------------------------------------------------------

//-- 2. Create the first WiFi User (sink)
//----------------------------------------------
Ptr<Receiver> receiver[users];
for(int i=0; i<users; i++){
    Ptr<Node> appSink = NodeList::GetNode (i+1);
    receiver[i] = CreateObject<Receiver>();
    receiver[i]->SetAttribute("Port", UintegerValue(1000+
        i));//Lisening Port
    appSink->AddApplication (receiver[i]);
    receiver[i]->SetStartTime (Seconds (0));

    //Set IP address of the first User to AP (source)
    string ipString = "192.168.0."+std::to_string(i+2);
    const char* ip = ipString.c_str();
    //cout << ip;
    Config::Set ("/NodeList/*/ApplicationList/"+std::
        to_string(i)+"/$Sender/Destination",
                Ipv4AddressValue (ip));
}

//
    ------------------------------------------------------------

//-- Setup stats and data collection
//--  for the WiFi AP and the first WiFi User
//----------------------------------------------

// Create a DataCollector object to hold information
    about this run.
DataCollector dataofuser[users];
for(int i=0; i<users; i++){
    dataofuser[i].DescribeRun (experiment,
                    strategy,
                    input+" user"+std::to_string(i+1),
                    runID);

    // Add any information we wish to record about this
        run.
    dataofuser[i].AddMetadata ("author", "EE500-Michael")
        ; //Please replace XXX with your first name!
}
```

```
// Create a counter to track how many frames are generated.  Updates
// are triggered by the trace signal generated by the WiFi MAC model
// object.  Here we connect the counter to the signal via the simple
// TxCallback() glue function defined above.
Ptr<CounterCalculator<uint32_t> > totalTx[users];
Ptr<PacketCounterCalculator> totalRx[users];
Ptr<PacketCounterCalculator> appTx[users];
Ptr<CounterCalculator<> > appRx[users];
Ptr<PacketSizeMinMaxAvgTotalCalculator> appTxPkts[users];
Ptr<TimeMinMaxAvgTotalCalculator> delayStat[users];
for(int i=0; i<users; i++){
  totalTx[i] = CreateObject<CounterCalculator<uint32_t> >();
  totalRx[i] = CreateObject<PacketCounterCalculator>();
  appTx[i] = CreateObject<PacketCounterCalculator>();
  appRx[i] = CreateObject<CounterCalculator<> >();
  appTxPkts[i] = CreateObject<PacketSizeMinMaxAvgTotalCalculator>();
  delayStat[i] = CreateObject<TimeMinMaxAvgTotalCalculator>();


  totalTx[i]->SetKey ("wifi-tx-frames");
  totalTx[i]->SetContext ("node[0]");
  Config::Connect ("/NodeList/0/DeviceList/*/$ns3::WifiNetDevice/Mac/MacTx",
                   MakeBoundCallback (&TxCallback, totalTx[i]));
  dataofuser[i].AddDataCalculator (totalTx[i]);

// This is similar, but creates a counter to track how many frames
// are received.  Instead of our own glue function, this uses a
// method of an adapter class to connect a counter directly to the
// trace signal generated by the WiFi MAC.
  totalRx[i]->SetKey ("wifi-rx-frames");
  totalRx[i]->SetContext ("node["+std::to_string(i+1)+"]");
  Config::Connect ("/NodeList/"+std::to_string(i+1)+"/DeviceList/*/$ns3::WifiNetDevice/Mac/MacRx",
                   MakeCallback (&PacketCounterCalculator::PacketUpdate,
                                 totalRx[i]));
```

```
  dataofuser[i].AddDataCalculator (totalRx[i]);

// This counter tracks how many packets--as opposed to
    frames--are
// generated.  This is connected directly to a trace
    signal provided
// by our Sender class.
  appTx[i]->SetKey ("sender-tx-packets");
  appTx[i]->SetContext ("node[0]");
  Config::Connect ("/NodeList/0/ApplicationList/"+std::
      to_string(i)+"/$Sender/Tx",
                      MakeCallback (&
                          PacketCounterCalculator::
                          PacketUpdate,
                                      appTx[i]));
  dataofuser[i].AddDataCalculator (appTx[i]);

// Here a counter for received packets is directly
    manipulated by
// one of the custom objects in our simulation, the
    Receiver
// Application.  The Receiver object is given a pointer
    to the
// counter and calls its Update() method whenever a
    packet arrives.
  appRx[i]->SetKey ("receiver-rx-packets");
  appRx[i]->SetContext ("node["+std::to_string(i+1)+"]"
      );
  receiver[i]->SetCounter (appRx[i]);
  dataofuser[i].AddDataCalculator (appRx[i]);

/**
 * Just to show this is here...
 Ptr<MinMaxAvgTotalCalculator<uint32_t> > test =
 CreateObject<MinMaxAvgTotalCalculator<uint32_t> >();
 test->SetKey("test-dc");
 data.AddDataCalculator(test);

 test->Update(4);
 test->Update(8);
 test->Update(24);
 test->Update(12);
**/

// This DataCalculator connects directly to the
    transmit trace
// provided by our Sender Application.  It records some
    basic
// statistics about the sizes of the packets received (
    min, max,
```

```
// avg, total # bytes), although in this scenaro they'
   re fixed.
  appTxPkts[i]->SetKey ("tx-pkt-size");
  appTxPkts[i]->SetContext ("node[0]");
  Config::Connect ("/NodeList/0/ApplicationList/"+std::
      to_string(i)+"/$Sender/Tx",
                  MakeCallback
                    (&
                        PacketSizeMinMaxAvgTotalCalculator
                        ::PacketUpdate,
                    appTxPkts[i]));
  dataofuser[i].AddDataCalculator (appTxPkts[i]);

// Here we directly manipulate another DataCollector
   tracking min,
// max, total, and average propagation delays.  Check
   out the Sender
// and Receiver classes to see how packets are tagged
   with
// timestamps to do this.
  delayStat[i]->SetKey ("delay");
  delayStat[i]->SetContext (".");
  receiver[i]->SetDelayTracker (delayStat[i]); //
      nanoseconds
  dataofuser[i].AddDataCalculator (delayStat[i]);
}

//
   ------------------------------------------------------------

//-- Run the simulation
//-------------------------------------------
NS_LOG_INFO ("Run Simulation.");
Simulator::Stop(Seconds(simTime));
Simulator::Run ();

//
   ------------------------------------------------------------

//-- Generate statistics output.
//-------------------------------------------

// Pick an output writer based in the requested format.
Ptr<DataOutputInterface> output = 0;
if (format == "omnet") {
    NS_LOG_INFO ("Creating omnet formatted data output.
        ");
    output = CreateObject<OmnetDataOutput>();
  } else if (format == "db") {
  #ifdef STATS_HAS_SQLITE3
```

```
      NS_LOG_INFO ("Creating sqlite formatted data output
          .");
      output = CreateObject<SqliteDataOutput>();
    #endif
    } else {
      NS_LOG_ERROR ("Unknown output format " << format);
    }

  // Finally, have that writer interrogate the
      DataCollector and save
  // the results.
  if (output != 0)
  {
    output->SetFilePrefix(dbPrefix);
    for(int i=0; i<users; i++){
      output->Output (dataofuser[i]);
    }
  }
  // Free any memory here at the end of this example.

  Simulator::Destroy ();

  // end main
}
```

3.3 Gnuplot Files

The following scripts are used to plot the values extracted from the .db files output from the simulations.

3.3.1 Question A Part 1 Bash Script

```
set terminal png  enhanced lw 2 font Helvetica 14

set size 1.0, 0.66

#-------------------------------------------------------
set out "wifi-throughput.png"
set title "Average Throughput Over Distance"
set xlabel "Distance (m)"
set xrange [0:210]
set ylabel "Throuhput [Mbps]"
set yrange [0:1]

plot "throughput.data" with lines title "WiFi Throughput
    vs. Distance"

#-------------------------------------------------------
set out "wifi-delay.png"
set title "Average Delay Over Distance"
```

```
set xlabel "Distance (m)"
set xrange [0:210]
set ylabel "Delay [ms]"
set yrange [0:1]

plot "delay.data" with lines title "WiFi Delay vs.
    Distance"

#----------------------------------------------------------------
set out "wifi-loss.png"
set title "Average Loss Over Distance"
set xlabel "Distance (m)"
set xrange [0:210]
set ylabel "Packet Loss Ratio [%]"
set yrange [0:1]

plot "loss.data" with lines title "WiFi PLR vs. Distance"
```

3.3.2 Question A Part 2 Bash Script

```
set terminal png  enhanced lw 2 font Helvetica 14

set size 1.0, 0.66

#----------------------------------------------------------------
set out "wifi-throughput.png"
set title "Average Throughput Over Transmission Interval"
set xlabel "Transmission Interval (s)"
set xrange [0:0.01]
set ylabel "Throuhput [Mbps]"
set yrange [0:20]

plot "throughput.data" with lines title "WiFi Throughput
    vs. txInterval"

#----------------------------------------------------------------
set out "wifi-delay.png"
set title "Average Delay Over Transmission Interval"
set xlabel "Transmission Interval (s)"
set xrange [0:0.01]
set ylabel "Delay [ms]"
set yrange [0:300]

plot "delay.data" with lines title "WiFi Delay vs.
    txInterval"

#----------------------------------------------------------------
set out "wifi-loss.png"
set title "Average Loss Over Transmission Interval"
set xlabel "Transmission Interval (s)"
```

```
set xrange [0:0.01]
set ylabel "Packet Loss Ratio [%]"
set yrange [0:40]

plot "loss.data" with lines title "WiFi PLR vs.
    txInterval"
```

3.3.3 Question B Bash Script

```
set terminal png  enhanced lw 2 font Helvetica 14

set size 1.0, 0.66

#---------------------------------------------------------------
set out "wifi-throughput.png"
set title "Average Throughput Over Distance"
set xlabel "Distance (m)"
set xrange [0:140]
set ylabel "Throuhput [Mbps]"
set yrange [0:0.2]

plot "throughput.data" with lines title "WiFi Throughput
    vs. Distance"

#---------------------------------------------------------------
set out "wifi-delay.png"
set title "Average Delay Over Distance"
set xlabel "Distance (m)"
set xrange [0:140]
set ylabel "Delay [ms]"
set yrange [0:10]

plot "delay.data" with lines title "WiFi Delay vs.
    Distance"

#---------------------------------------------------------------
set out "wifi-loss.png"
set title "Average Loss Over Distance"
set xlabel "Distance (m)"
set xrange [0:140]
set ylabel "Packet Loss Ratio [%]"
set yrange [0:100]

plot "loss.data" with lines title "WiFi PLR vs. Distance"
```

3.3.4 Question C Part 1 Bash Script

```
set terminal png  enhanced lw 2 font Helvetica 14

set size 1.0, 0.66
```

```
#---------------------------------------------------------------------------
set out "wifi-throughput.png"
set title "Average Throughput Over Distance"
set xlabel "Distance (m)"
set xrange [0:210]
set ylabel "Throuhput [Mbps]"
set yrange [0:4]

plot "usr1TP.data" with lines title "User 1", \
     "usr2TP.data" with lines title "User 2"
#---------------------------------------------------------------------------
set out "wifi-delay.png"
set title "Average Delay Over Distance"
set xlabel "Distance (m)"
set xrange [0:210]
set ylabel "Delay [ms]"
set yrange [0:4]

plot "usr1delay.data" with lines title "User 1", \
     "usr2delay.data" with lines title "User 2"

#---------------------------------------------------------------------------
set out "wifi-loss.png"
set title "Average Loss Over Distance"
set xlabel "Distance (m)"
set xrange [0:210]
set ylabel "Packet Loss Ratio [%]"
set yrange [0:1]

plot "usr1loss.data" with lines title "User 1", \
     "usr2loss.data" with lines title "User 2"
```

3.3.5 Question C Part 2 Bash Script

```
set terminal png  enhanced lw 2 font Helvetica 14

set size 1.0, 0.66

#---------------------------------------------------------------------------
set out "wifi-throughput.png"
set title "Average Throughput vs No. of Users"
set xlabel "Users"
set xrange [0:60]
set ylabel "Throuhput [Mbps]"
set yrange [0:2]

plot "1usrTP.data" with lines title "1 User", \
     "5usrTP.data" with lines title "5 Users", \
     "10usrTP.data" with lines title "10 Users", \
     "20usrTP.data" with lines title "20 Users", \
```

```
     "50usrTP.data" with lines title "50 Users"
#------------------------------------------------------------------
set out "wifi-delay.png"
set title "Average Delay vs No. of Users"
set xlabel "Users"
set xrange [0:60]
set ylabel "Delay [ms]"
set yrange [0:2000]

plot "1usrdelay.data" with lines title "1 User", \
     "5usrdelay.data" with lines title "5 Users", \
     "10usrdelay.data" with lines title "10 Users", \
     "20usrdelay.data" with lines title "20 Users", \
     "50usrdelay.data" with lines title "50 Users"

#------------------------------------------------------------------
set out "wifi-loss.png"
set title "Average vs No. of Users"
set xlabel "Users"
set xrange [0:60]
set ylabel "Packet Loss Ratio [%]"
set yrange [0:5]

plot "1usrloss.data" with lines title "1 User", \
     "5usrloss.data" with lines title "5 Users", \
     "10usrloss.data" with lines title "10 Users", \
     "20usrloss.data" with lines title "20 Users", \
     "50usrloss.data" with lines title "50 Users"
```

3.3.6 Question C Part 3 Bash Script

```
set terminal png  enhanced lw 2 font Helvetica 14

set size 1.0, 0.66

#------------------------------------------------------------------
set out "wifi-throughput.png"
set title "Average Throughput Over Distance"
set xlabel "Distance (m)"
set xrange [0:210]
set ylabel "Throuhput [Mbps]"
set yrange [0:2]

plot "1usrTP.data" with lines title "1 User", \
     "5usrTP.data" with lines title "5 Users", \
     "10usrTP.data" with lines title "10 Users", \
     "15usrTP.data" with lines title "15 Users", \
     "20usrTP.data" with lines title "20 Users"

#------------------------------------------------------------------
```

```
set out "wifi-delay.png"
set title "Average Delay Over Distance"
set xlabel "Distance (m)"
set xrange [0:210]
set ylabel "Delay [ms]"
set yrange [0:520]

plot "1usrdelay.data" with lines title "1 User", \
     "5usrdelay.data" with lines title "5 Users", \
     "10usrdelay.data" with lines title "10 Users", \
     "15usrdelay.data" with lines title "15 Users", \
     "20usrdelay.data" with lines title "20 Users"

#---------------------------------------------------------------
set out "wifi-loss.png"
set title "Average Loss Over Distance"
set xlabel "Distance (m)"
set xrange [0:210]
set ylabel "Packet Loss Ratio [%]"
set yrange [0:100]

plot "1usrloss.data" with lines title "1 User", \
     "5usrloss.data" with lines title "5 Users", \
     "10usrloss.data" with lines title "10 Users", \
     "15usrloss.data" with lines title "15 Users", \
     "20usrloss.data" with lines title "20 Users"
```