

# **Approaches to Improve the Wi-Fi Range**

ET4394 Wireless Networking

by

**Kangqi Li**

4518942



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Project Description</b>	<b>3</b>
2.1	Objective . . . . .	3
2.2	Hypothesis . . . . .	3
<b>3</b>	<b>Results and Analysis</b>	<b>5</b>
3.1	Simulation scenario . . . . .	5
3.2	Performance at different distances . . . . .	6
3.3	Range vs. Transmit power . . . . .	7
3.4	Range vs. Antenna gain . . . . .	8
3.5	Range vs. Protocols . . . . .	9
	<b>Conclusion</b>	<b>11</b>
	<b>References</b>	<b>13</b>



# 1

## Introduction

IEEE 802.11 is a set of media access control (MAC) and physical layer (PHY) specifications for implementing wireless local area network (WLAN) computer communication.[1] Wireless signals propagating through the air lose strength while encountering natural and manmade obstacles hence Wi-Fi signals cannot travel to an infinite distance. The farther a WLAN signal goes, the weaker it gets. This is known as attenuation. When signal power decreases to relatively low values, the receiving 802.11 radio will likely encounter bit errors when decoding the signal. Therefore, only in the covering area can the WLAN achieve the expected performance.

There are many factors affecting the covering area. The material and size of obstacles, the environment, the sensitivity and direction of antennas, the transmit power and frequency are all related to the transmission and detection of signals. Also, the ability of convey signal to long distance differs with the kind of used protocols with different MAC and PHY.



# 2

## Project Description

### 2.1. Objective

In this project, network simulation will be conducted to find the relation between the data rate, the transmit power, the antenna gain and the protocol types with the covering area. The average throughput, the delay and the packet loss will be measured to represent the network performance.

### 2.2. Hypothesis

Weak transmit power will definitely limit the covering range because the signal will be lower than the energy threshold then will not be captured after attenuation. Antenna gain represents the sensitivity of antennas hence larger antenna gain will make the weak the signal be transmitted and captured. Different 802.11 layers use different frequencies and data rate. In the same condition, high-frequency radio signals suffer high power attenuation. Thus there is a trade-off between the speed and the covering range of wireless networks.

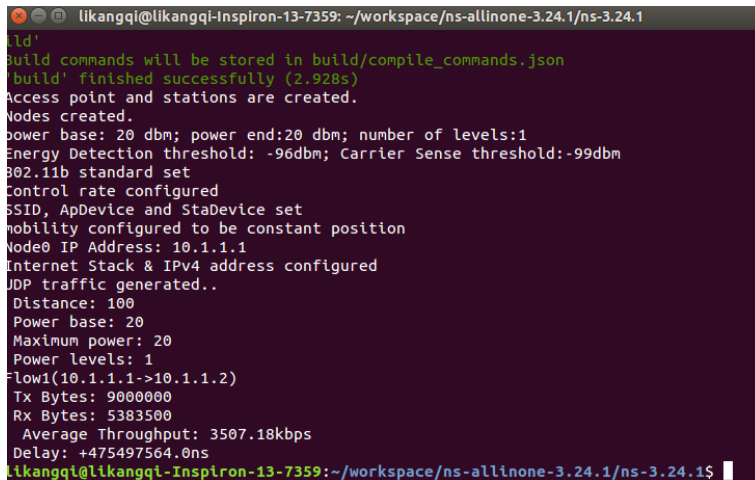




# 3

## Results and Analysis

### 3.1. Simulation scenario

A terminal window with a dark purple background and green text. The window title is 'likangqi@likangqi-Inspiron-13-7359: ~/workspace/ns-allinone-3.24.1/ns-3.24.1'. The output shows the successful completion of a build, the creation of an access point and station, and various configuration parameters. At the end, it displays performance metrics for a flow between two nodes.

```
likangqi@likangqi-Inspiron-13-7359: ~/workspace/ns-allinone-3.24.1/ns-3.24.1$ ./build
'build' finished successfully (2.928s)
Access point and stations are created.
Nodes created.
power base: 20 dbm; power end:20 dbm; number of levels:1
Energy Detection threshold: -96dbm; Carrier Sense threshold:-99dbm
802.11b standard set
Control rate configured
SSID, ApDevice and StaDevice set
mobility configured to be constant position
Node0 IP Address: 10.1.1.1
Internet Stack & IPv4 address configured
UDP traffic generated..
Distance: 100
Power base: 20
Maximum power: 20
Power levels: 1
Flow1(10.1.1.1->10.1.1.2)
Tx Bytes: 9000000
Rx Bytes: 5383500
Average Throughput: 3507.18kbps
Delay: +475497564.0ns
likangqi@likangqi-Inspiron-13-7359:~/workspace/ns-allinone-3.24.1/ns-3.24.1$
```

Figure 3.1: Simulation results by using NS3

Figure 3.1 shows the simulation results with parameters and automatically calculated average throughput, delay and packet loss. In this project, only one access point and one station are created and both of them are using static mobility model. Constant Speed Propagation Delay Model and Friis Propagation Loss Model are used to simulate the signal attenuation.

Table 1 lists the default parameters of the scenario. In following sections, single parameter will be adjusted to evaluate the influences from various aspects.

Table 1. Default Scenario

Protocol	802.11b
Packet Size	1472 bits
Data Rate	11 Mbps
Reception Gain	0
Transission Gain	0
Transmit Power	16.0206 dBm (40 mW)
ED Threshold	-96 dB
Cca Threshold	-99 dB
Simulation Time	10 s
Total Transmit Data	9000000 bytes

### 3.2. Performance at different distances

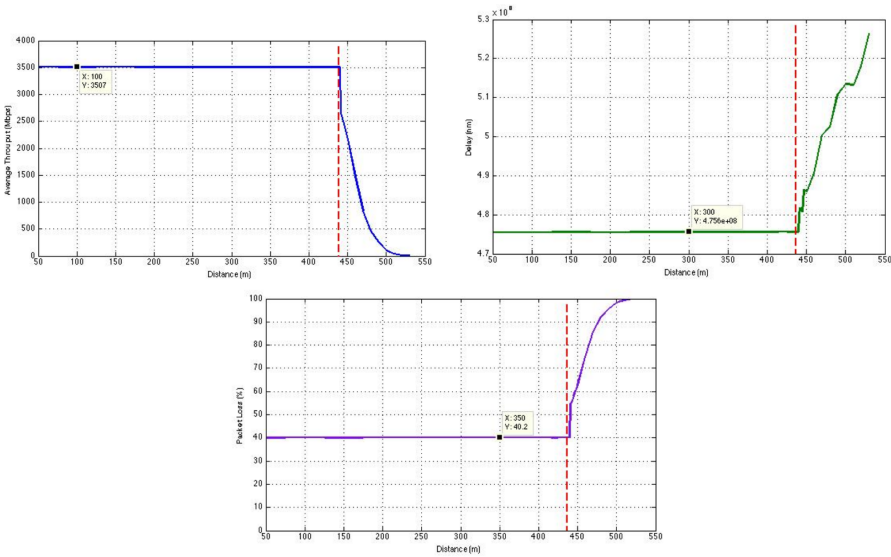


Figure 3.2: Average throughput, delay and packet loss at different distances

Figure 3.2 shows the change of performance with different distances between the access point and the station when the transmit power is 16.0206 dBm (40 mW), antenna gain is 0, protocol is 802.11b, channel width is 22 MHz, payload is 1472 bytes, frequency is 2.407 GHz, ED threshold is -96 dBm, Cca threshold is -99 dBm and data rate is 11 Mbps. The average throughput is the average speed of all bits conveyed successfully, the delay is the average time delay for every packet. Here

we define the working range is defined as the maximum distance that can keep the best performance of the network. From Figure 3.2, all the average throughput, delay and packet loss decreases suddenly after the red line. Therefore, the covering range at this condition is 440 m.

### 3.3. Range vs. Transmit power

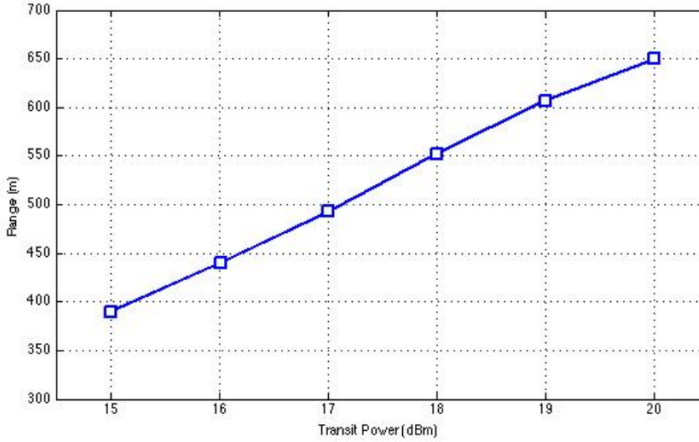


Figure 3.3: Covering range vs transmit power of 802.11b network when reception gain is 0, transmission gain is 0, data rate is 11 Mbps

Figure 3.3 shows that the covering range increases with the increase of the transmit power as a linear relationship. This is because of the Friis Propagation Loss Model is used. In this model, the received power is calculated by:

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^2 L}$$

With:

$P_r$ : reception power (W)

$P_t$ : transmission power (W)

$G_t$ : transmission gain (unit-less)

$G_r$ : reception gain (unit-less)

$\lambda$ : wavelength (m)

$d$ : distance (m)

$L$ : system loss (unit-less)

When other parameters are the same, the reception power is linear to the transmission power. Hence the covering range of network can be enlarged by increasing the transmit power. But when the antenna gain is 0, the maximum transmit power of 802.11b protocol is 100 mW (20 dBm). This constrains the solution to increase covering range by using larger transmit power as one bottleneck of 802.11b.

### 3.4. Range vs. Antenna gain

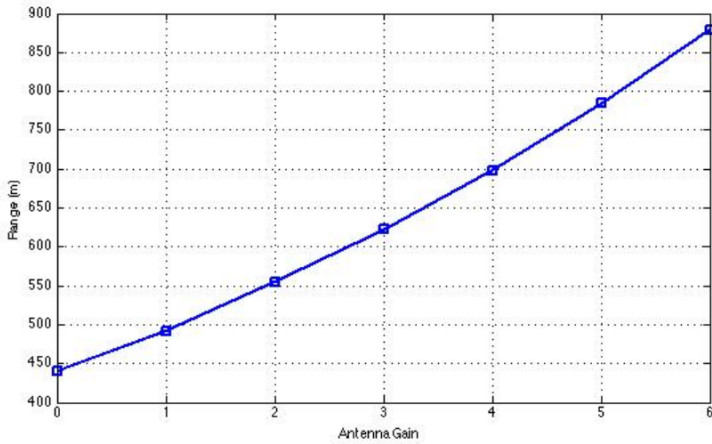


Figure 3.4: Covering range vs reception gain of 802.11b network when transmission gain is 0, transmit power is 16.0206 dBm, data rate is 11 Mbps

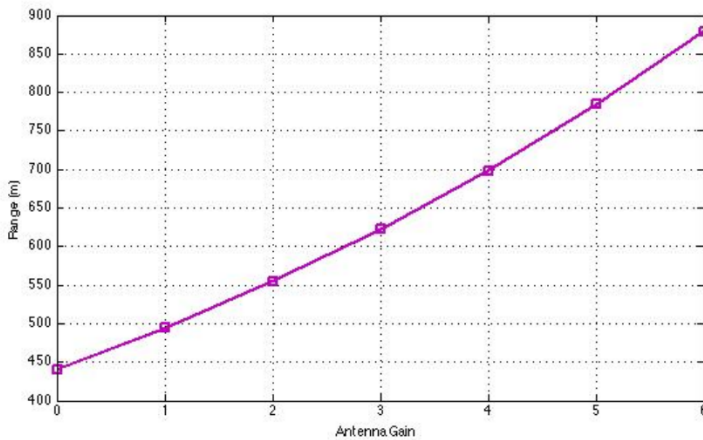


Figure 3.5: Covering range vs transmission gain of 802.11b network when transmission gain is 0, reception power is 16.0206 dBm, data rate is 11 Mbps

Figure 3.3 and Figure 3.4 show that both reception gain and transmission gain increases the covering range. Reception gain enhances the ability of receiver to detect weak signals while transmission gain makes the transmitter strengthen the signals before sending. Better antennas can amplified the weak signals larger than the energy detection threshold then captured as valid signals.

### 3.5. Range vs. Protocols

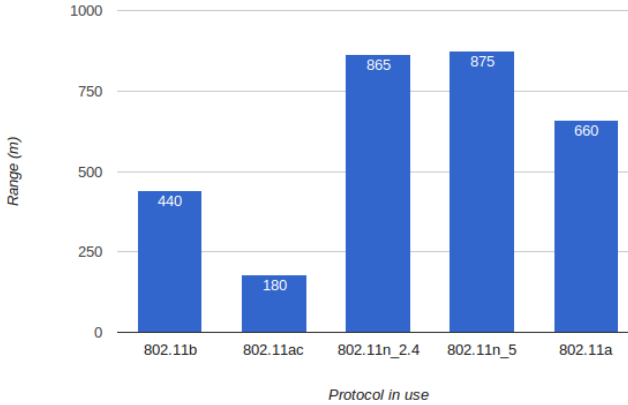


Figure 3.6: Covering range vs reception gain of 802.11b network when transmission gain is 0, transmit power is 16.0206 dBm, reception gain is 0

To verify the influences of protocols, various protocols of 802.11 standard are tested like Figure 3.6 shows above. The differences of these protocols mainly are the PHY layer, MAC layer and frequencies. 802.11a uses OFDM PHY for the 5 GHz band, 802.11b uses DSSS PHY for the 2.4 GHz band, 802.11n2.4 uses HT OFDM PHY for the 2.4 GHz band, 802.11n5 uses HT OFDM PHY for the 5 GHz band and 802.11ac uses VHT OFDM PHY for the 5 GHz band. Different PHY achieves different data rate. Since usually the data rate cannot be set randomly, it is impossible to make all networks in the same data rate. Hence in this section similar data rates are set to all protocols like the data rate of 802.11b is 11 Mbps and that of 802.11ac is 12 Mbps.

Simulation results of different protocols are shown in Figure 3.6. Usually, high band will contribute to a larger throughput. However, in section 3.2, the equation of Friis propagation loss model is discussed which shows that there is a negative relationship between the received power and the signal frequency. Therefore, theoretically wireless networks with 5 GHz bandwidth have smaller covering ranges when compared to 2.4 GHz wireless networks like the 802.11ac network performs. Thus, these is a trade-off when building a long-range Wi-Fi. Large range will sacrifice the throughput.



# Conclusion

Based on all the simulation results, there are several ways to increase the Wi-Fi range:

- Increase the transmit power.
- Use sensitive antennas.
- Use protocols with low frequency and PNY for long-range WLAN.

There are some work can be done to find more approaches:

- There are only one access point and one station node, which could not represent the advantage of 802.11ac and 802.11n network to support multi-user cases.
- All nodes are static now. Other mobilities can be introduced to test the performance.
- Friis Propagation Loss Model and Constant Speed Propagation Delay Model are used in this project. However, for long-distance, outdoor networks, other propagation model in NS3 like Okumura Hata Propagation Loss Model can also be considered since the environment also affect WLANs a lot.
- The antenna gain is affected not only the sensitivity of antennas but also the direction of antennas. In-depth analysis of the affect of antenna gain can be applied.





# References

1. [https://en.wikipedia.org/wiki/IEEE\\_802.11ac](https://en.wikipedia.org/wiki/IEEE_802.11ac)
2. <https://www.nsnam.org/doxygen-release/index.html>
3. [https://www.nsnam.org/doxygen/classns3\\_1\\_1\\_friis\\_propagation\\_loss\\_model.html#details](https://www.nsnam.org/doxygen/classns3_1_1_friis_propagation_loss_model.html#details)
4. [https://en.wikipedia.org/wiki/Network\\_delay](https://en.wikipedia.org/wiki/Network_delay)
5. [https://en.wikipedia.org/wiki/Packet\\_loss](https://en.wikipedia.org/wiki/Packet_loss)
6. [https://en.wikipedia.org/wiki/Long-range\\_Wi-Fi](https://en.wikipedia.org/wiki/Long-range_Wi-Fi)



# Appendix

```
1 #include "ns3/core-module.h"
2 #include "ns3/network-module.h"
3 #include "ns3/applications-module.h"
4 #include "ns3/wifi-module.h"
5 #include "ns3/mobility-module.h"
6 #include "ns3/ipv4-global-routing-helper.h"
7 #include "ns3/internet-module.h"
8 #include "ns3/flow-monitor-module.h"
9 #include "ns3/random-variable-stream.h"
10 #include "ns3/netanim-module.h"
11 #include "ns3/propagation-module.h"
12 #include "ns3/propagation-loss-model.h"
13 #include <iostream>
14 #include <fstream>
15 #include <vector>
16 #include <string>
17
18 using namespace ns3;
19 NS_LOG_COMPONENT_DEFINE ("ProjectNS3");
20 int main (int argc, char *argv[])
21 {
22     int nNodes = 1;
23     double StartTime = 0.0;
24     double StopTime = 10.0;
25     uint32_t payloadSize = 1472;
26     uint32_t maxPacket = 10000;
27     double edThreshold = -96.0;
28     double ccaThreshold = -99.0;
29     double powerStart = 16.0206;
30     double distance = 100.0;
31     double rg = 0;
32     double tg = 0;
33
34     StringValue DataRate;
35     //DataRate = StringValue("DsssRate11Mbps"); //for 802.11b
36     DataRate = StringValue("OfdmRate12Mbps"); //for 802.11a
37     //DataRate = VhtWifiMacHelper::DataRateForMcs (0); //for 802.11ac
38     //DataRate = HtWifiMacHelper::DataRateForMcs (0); //for 802.11n
39
40     CommandLine cmd;
41     cmd.AddValue("distance", "Distance between AP and STA", distance);
42     cmd.AddValue("ps", "Base power of PHY", powerStart);
43     cmd.AddValue("t", "Simulation time", StopTime);
44     cmd.AddValue("rg", "Antenna gain of receiver", rg);
45     cmd.AddValue("tg", "Simulation time", tg);
46     cmd.Parse (argc, argv);
47
48 }
```

```

49 //Nodes generation
50 NodeContainer wifiApNode;
51 wifiApNode.Create(1);
52
53 std::cout<<"Access point and stations are created."<< '\n';
54
55 // PHY layer setup
56 NodeContainer wifiStaNodes;
57 wifiStaNodes.Create(nNodes);
58 std::cout<<"Nodes created."<< '\n';
59
60 YansWifiPhyHelper phy = YansWifiPhyHelper::Default();
61 phy.Set("RxGain", DoubleValue(rg));
62 phy.Set("TxGain", DoubleValue(tg));
63 phy.Set("TxPowerStart", DoubleValue(powerStart));
64 phy.Set("TxPowerEnd", DoubleValue(powerStart));
65 std::cout<<"power base: "<<powerStart<<" dbm; power end:"<<powerStart<<"
    dbm;"<<'\n';
66
67 phy.Set("EnergyDetectionThreshold", DoubleValue(edThreshold));
68 phy.Set("CcaMode1Threshold", DoubleValue(ccaThreshold));
69 std::cout<<"Energy Detection threshold: "<<edThreshold<<"dbm; Carrier Sense
    threshold:"<<ccaThreshold<<"dbm"<< '\n';
70
71 //Channel setup
72 YansWifiChannelHelper channel;
73 channel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
74 channel.AddPropagationLoss("ns3::FriisPropagationLossModel");
75 phy.SetChannel(channel.Create());
76
77 //wifi setup
78 WifiHelper wifi = WifiHelper::Default();
79 //wifi.SetStandard( WIFI_PHY_STANDARD_80211b); //for 802.11b
80 //std::cout<<"802.11b standard set"<<'\n';
81
82 //wifi.SetStandard( WIFI_PHY_STANDARD_80211ac); //for 802.11ac
83 //std::cout<<"802.11ac standard set"<<'\n';
84
85 //wifi.SetStandard( WIFI_PHY_STANDARD_80211n_5GHZ); //for 802.11n 5 GHz
86 //std::cout<<"802.11n 5 G standard set"<<'\n';
87
88 //wifi.SetStandard( WIFI_PHY_STANDARD_80211n_2_4GHZ); //for 802.11n 2.4 GHz
89 //std::cout<<"802.11n 2.4 G standard set"<<'\n';
90
91 wifi.SetStandard( WIFI_PHY_STANDARD_80211a);
92 std::cout<<"802.11b standard set"<<'\n';
93 wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager", "DataMode",
    DataRate, "ControlMode", DataRate);
94
95
96
97 NqosWifiMacHelper mac = NqosWifiMacHelper::Default(); //for 802.11b and
    802.11a
98 //VhtWifiMacHelper mac = VhtWifiMacHelper::Default (); //for 802.11ac
99 //HtWifiMacHelper mac = HtWifiMacHelper::Default (); //for 802.11n
100 std::cout<<"Control rate configured"<<'\n';
101

```

```

102 //MAC setup
103 Ssid ssid = Ssid("ProjectNS3");
104 mac.SetType("ns3::StaWifiMac", "Ssid", SsidValue(ssid), "ActiveProbing",
    BooleanValue(false));
105
106 NetDeviceContainer staDevices;
107 staDevices=wifi.Install(phy,mac,wifiStaNodes);
108
109 mac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssid));
110
111 NetDeviceContainer apDevice;
112 apDevice=wifi.Install(phy,mac,wifiApNode);
113 std::cout<<"SSID, ApDevice and StaDevice set"<<'\n';
114
115 //Mobility setup
116 MobilityHelper mobility;
117 mobility.SetPositionAllocator("ns3::GridPositionAllocator",
118     "MinX", DoubleValue(0.0),
119     "MinY", DoubleValue(0.0),
120     "DeltaX", DoubleValue(distance),
121     "DeltaY", DoubleValue(0.0),
122     "GridWidth", UIntegerValue(nNodes + 1),
123     "LayoutType", StringValue("RowFirst"));
124
125
126 mobility.Install(wifiStaNodes);
127
128
129 mobility.Install(wifiApNode);
130 mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
131 std::cout<<"mobility configured to be constant position"<<'\n';
132
133 //Stack & address setup
134 InternetStackHelper stack;
135 stack.Install(wifiApNode);
136 stack.Install(wifiStaNodes);
137
138 Ipv4AddressHelper address;
139 Ipv4Address addr;
140 address.SetBase("10.1.1.0", "255.255.255.0");
141 Ipv4InterfaceContainer staNodesInterface;
142 Ipv4InterfaceContainer apNodeInterface;
143 staNodesInterface = address.Assign(staDevices);
144 apNodeInterface = address.Assign(apDevice);
145
146 addr = apNodeInterface.GetAddress(0);
147 for(int i = 0; i < nNodes; i++)
148 {
149     addr = staNodesInterface.GetAddress(i);
150     std::cout << "Node" << i << " IP Address: " << addr << std::endl;
151 }
152 std::cout<<"Internet Stack & IPv4 address configured"<<'\n';
153
154 //Server & client setup
155 ApplicationContainer serverApp;
156 UdpServerHelper myServer(400);
157 serverApp = myServer.Install(wifiStaNodes.Get(0));

```

```

158 serverApp.Start(Seconds(StartTime));
159 serverApp.Stop(Seconds(StopTime));
160
161 UdpClientHelper myClient(apNodeInterface.GetAddress(0),400);
162 myClient.SetAttribute ("MaxPackets", UIntegerValue(maxPacket));
163 myClient.SetAttribute ("Interval", TimeValue(Time("0.002")));
164 myClient.SetAttribute ("PacketSize", UIntegerValue(payloadSize));
165 ApplicationContainer clientApp;
166 clientApp = myClient.Install (wifiStaNodes.Get(0));
167 clientApp.Start(Seconds(StartTime));
168 clientApp.Stop (Seconds(StopTime+5));
169 std::cout << "UDP traffic generated.." << '\n';
170
171 // Calculate Throughput & Delay using Flowmonitor
172 FlowMonitorHelper flowmon;
173 Ptr<FlowMonitor> monitor = flowmon.InstallAll();
174 AnimationInterface anim ("projectAnim.xml");
175 Simulator::Stop (Seconds(StopTime+2));
176 Simulator::Run();
177
178 monitor->CheckForLostPackets();
179 monitor->SerializeToXmlFile("projectFlow.xml", true, true);
180 Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>(
    flowmon.GetClassifier());
181 std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats();
182
183 std::cout<<" Distance: " << distance<< "\n";
184 std::cout<<" Maximum power: " << powerStart<< "\n";
185 std::cout<<" Receiver antenna gain: " <<rg << "\n";
186 std::cout<<" Transmitter antenna gain: " <<tg << "\n";
187 for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.
    begin (); i != stats.end () ; ++i)
188 {
189
190
191
192 Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
193 std::cout<<"Flow"<<i->first<<"("<<t.sourceAddress<<"->" << t.
    destinationAddress << ")\n";
194 std::cout<<" Tx Bytes: " << i->second.txBytes << "\n";
195 std::cout<<" Rx Bytes: " << i->second.rxBytes << "\n";
196 std::cout<<" Packet Loss: " << ( i->second.txBytes - i->second.rxBytes) *
    100 / i->second.txBytes << "% \n";
197 std::cout<<" Average Throughput: " << i->second.rxBytes * 8.0 / (i->
    second.timeLastRxPacket.GetSeconds() - i->second.timeFirstTxPacket.
    GetSeconds()) / 1024 / nNodes << "kbps\n";
198 std::cout << " Delay: " << i->second.delaySum / i->second.rxPackets << "\
    n";
199
200 }
201 Simulator::Destroy();
202 return 0;
203 }

```