1. **Flow Monitor in “wifi-simple-adhoc-grid.cc” using “flowmon-parse-results.py” to evaluate results**

The “wifi-simple-adhoc-grid.cc” file was simply modified with the standard call for FlowMonitor. The difficult part was in developing the code to parse the XML output data. The example program “flowmon-parse-results.py” was modified to suit this purpose. Snippets of the modified code can be found in the appendix.

1. **Traffic trials with flow monitor**

All trials were done at distance of 100m and a simulation time of 33.0 seconds, which gave 3 seconds for data transfer after the OSLR was converged. The standard command line inputs were:

1. ./waf
2. ./waf --run "wifi-simple-adhoc-grid.cc --numPackets=<int> --interval=<float>"
3. python3 ~/repos/wireless\_comm\_lab/Lab5/flowmon-parse-results\_Lab5.py wifi-simple-adhoc-grid.xml

See below for a few example outputs.

**Text

Description automatically generated**

Figure : Nptkt=1, interval=1.0 (default)

**Text

Description automatically generated**

Figure : Npkt=10, interval=0.5

Table 1: "wifi-simple-adhoc-grid" Traffic Trials

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Number of Packets** | **1** | **10** | **10** | **10** | **50** | **50** | **100** | **100** | **100** |
| **Interval** | **1.0** | **1.0** | **0.5** | **0.1** | **0.1** | **0.05** | **0.05** | **0.01** | **0.04** |
| txPackets | 1 | 3 | 6 | 10 | 30 | 50 | 60 | 100 | 75 |
| rxPackets | 1 | 3 | 6 | 10 | 30 | 48 | 57 | 14 | 31 |
| txBytes | 1028 | 3084 | 6168 | 10280 | 30840 | 51400 | 61680 | 102800 | 77100 |
| rxBytes | 1028 | 3084 | 6168 | 10280 | 30840 | 49344 | 58596 | 14392 | 31868 |
| Tx Bitrate | None | 12.34 | 19.74 | 91.38 | 85.08 | 167.84 | 167.27 | 830.71 | 208.38 |
| Rx Bitrate | None | 12.69 | 20.19 | 97.41 | 86.74 | 169.01 | 168.28 | 68.53 | 140.29 |
| Mean Delay | 127.94 | 90.8 | 84.1 | 80.4 | 75.07 | 124.06 | 115.88 | 776.94 | 641.89 |

Table 1 shows the results of the trials when varying the number of packets and interval time. The optimal conditions appeared to occur when approximately 60 packets were transferred and an interval time around 0.05 sec was selected. Below that time, the number of packets received fell off drastically, the delay time increases significantly, and the resultant received bitrate falls off.

1. **Additional background flows**

Additional flows were added by including a 2nd sink/source and adding randomly spaced flows (see appendix). The sources/sinks were arranged, respectively:

1. sinkNode=0, sourceNode=24
2. sinkNode2=3, sourceNode2=20
3. sinkNode3=10, sourceNode3=5
4. sinkNode4=8, sourceNode4=18

The case of 10 packets and 0.1 interval time from Table 1 (above) was used as a benchmark. Table 2 shows the results of the trails by varying the number of flows when looking at the original flow (node 24 to node 0).

Table : Additional Flow model trials

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Packets** | 10 | 10 | 10 | 10 |
| **Interval** | 0.1 | 0.1 | 0.1 | 0.1 |
| **Flows** | **1** | **2** | **3** | **4** |
| **txPackets** | 10 | 10 | 10 | 10 |
| **rxPackets** | 10 | 7 | 7 | 9 |
| **txBytes** | 10280 | 10280 | 10280 | 10280 |
| **rxBytes** | 10280 | 7196 | 7196 | 9252 |
| **Tx Bitrate** | 91.38 | 91.38 | 91.38 | 91.38 |
| **Rx Bitrate** | 97.41 | 52.21 | 66.15 | 83.46 |
| **Mean Delay** | 80.4 | 187.15 | 37.35 | 49.27 |

Table 2 shows that increasing the congestion on the overall network didn’t necessarily have a detrimental effect the on the one flow of note. Going up to two flows increased the delay time and reduced the bitrate, but each additional flow seemed to increase the bitrate and decrease the delay time. Perhaps, as long as the network is not saturated, the number of hops that the packet needs to take by itself is decreased.

Text

Description automatically generated

Figure 3: Example output with 2 flows

1. **Takagami Loss Propagation Model**

Table : Packet/Interval Trials with Takagami Loss Propagation Model

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Number of Packets** | **10** | **10** | **10** | **50** | **50** | **100** | **100** |
| **Interval** | **1.0** | **0.5** | **0.1** | **0.1** | **0.05** | **0.05** | **0.01** |
| txPackets | 3 | 6 | 10 | 30 | 50 | 60 | 100 |
| rxPackets | 3 | 6 | 10 | 30 | 50 | 60 | 100 |
| txBytes | 3084 | 6168 | 10280 | 30840 | 51400 | 61680 | 102800 |
| rxBytes | 3084 | 6168 | 10280 | 30840 | 51400 | 61680 | 102800 |
| Tx Bitrate | 12.34 | 19.74 | 91.38 | 85.08 | 167.84 | 167.27 | 830.71 |
| Rx Bitrate | 12.41 | 19.83 | 92.53 | 85.41 | 168.61 | 167.91 | 824.79 |
| Mean Delay | 12.49 | 10.62 | 10.34 | 9.35 | 9.11 | 9.05 | 42.64 |

The implementation with the Takagami Loss Propagation model didn’t show much difference on the low end, when there were few packets and the interval times were large. However, the biggest difference was seen the Mean Delay time, which showed much lower values than the previous Friis Propagation Loss Model. This resulted in the simulation being able to take the transfers up to much higher numbers of packets and smaller interval times before the performance started falling off.

Appendix (code)

b)



A picture containing text

Description automatically generated

Text

Description automatically generated

c)

Text

Description automatically generated