Lab4
Igor Povarich
10/08/2021
Tasks and Deliverables

1. Examples
    a. The topology of wifi-simple-adhoc-grid is a 5x5 grid made to represent an adhoc network of 25 nodes. By default, node 24 is the source and 0 is the sink and the distance between each node is 500m (note this needs to be reduced to ensure successful packet transfer).
    b. The FlowMonitor outputs two experiments from the simulation. The first one where the RTS/CTS is enabled and one where it is disabled. The system appears to contain 3 nodes with node 1 transmitting to node 2 and node 3 also transmitting to node 2. The two experiments clearly show the difference between using RTS/CTS. In the disabled one, Flow 1 delivers a lot of data, but Flow 2 delivers none. In the enabled case, Flow 1 doesn't deliver as many packets as in the disabled case, but both Flows deliver the same amount of data. FlowMonitor is used to track certain metrics such as the number of bytes transmitted vs. number of bytes received and the respective rates of each.
2. FlowMonitor and Data
    a. See Figure 1 in appendix
    b. Ten default metrics
        i. timesForwarded: number of forwards can dictate the reliability of the path and how long it will take
        ii. lostPackets: any packets that are allocated incorrectly can skew your numbers
        iii. rxPackets: tells how much data (in packets) was successfully received
        iv. txPackets: tells how much data (in packets) was transferred - combined with rxPackets tells you how well that flow path worked
        v. rxBytes: tells how much data (in bytes) was successfully received
        vi. txBytes: tells how much data (in packets) was transferred combined with rxBytes tells you how well that flow path worked
        vii. jitterSum: variation in delay tells you how consistently the signals were transferring data
        viii. delaySum: this tells you how efficiently the packets were sent
        ix. timeLastRxPacket: essentially the time when the data transfer ended (minus any lost packets)
        x. timeFirstTxPacket: when the data transfer started
    c. Ten derived metrics
        i. TxOffered1 = txBytes * 8 / duration / 1000 / 1000
            - Tells rate of data sent in Mbps
        ii. TxOffered2 = txBytes * 8 / duration / 1000
            - Tells rate of data sent in kbps
        iii. TxOffered3 = txBytes * 8 / duration
            - Tells rate of data sent in bps
        iv. Throughput1 = rxBytes * 8 / duration / 1000 / 1000
            - Tells rate of data received in Mbps

v. Throughput2 = rxBytes * 8 / duration / 1000
   - Tells rate of data sent in kbps

vi. Throughput3 = rxBytes * 8 / duration / 1000
   - Tells rate of data sent in bps

vii. txSymbolRate = txPackets / duration
   - Symbol transfer rate - average number of symbols sent per second

viii. rxSymbolRate = rxPackets / duration
   - Symbol receive rate - average number of symbols successfully received

ix. lostPacketrate = lostPackets / duration
   - Number of packets lost per second of simulation time

x. delaySumSec = delaySum / 1e+6
   - Total delay in seconds

d. See b. and c.

e. Command Line:
   i. ../waf --run "wifi-simple-adhoc-grid.cc --numPackets=1 --interval=1.0"
   ii. ../waf --run "wifi-simple-adhoc-grid.cc --numPackets=30 --interval=0.1"

f. The case with 1 packet showed a timesForwarded of 7, which makes sense as it's the fewest number of steps (after the first) from one corner of a 5x5 grid. The case where 30 packets (max allowed) were sent showed a timesForwarded of 210, which is 7*30, as each packet is routed with the fewest number of steps (though they have multiple options). With the increased number of packets and 0.1 sec interval, a nonzero jitterSum and larger delaySum were introduced, as the transfers both can't be as consistent and have more opportunities to vary. Of course, with the number of packets increased, the total data transfer rate was increased significantly as well.

APPENDIX

```
// Give OLSR time to converge-- 30 seconds perhaps
Simulator::Schedule (Seconds (30.0), &GenerateTraffic,
                     source, packetSize, numPackets, interPacketInterval);

FlowMonitorHelper flowmon;
Ptr<FlowMonitor> monitor = flowmon.InstallAll ();

// Output what we are doing
//NS_LOG_UNCOND ("Testing from node " << sourceNode << " to " << sinkNode << " with grid distance

Simulator::Stop (Seconds (33.0));
Simulator::Run ();

monitor->SerializeToXmlFile("wifi-simple-adhoc-grid2.xml", true, true);
Simulator::Destroy ();
```

Figure 1: Lab4 - 2a, Implement basic FlowMonitor in "wifi-simple-adhoc-grid.cc"