CNT5505 Programming Assignment No. 3: Simulation of WiFi Protocols

(This programming assignment can be done either individually, or by a group of two. Due November 20)

**Purpose**

- Experience with network simulation.
- Better understanding of WiFi protocols.

**Description**

In this assignment, you will write programs to compare the performance of two WiFi protocols. The first is the 802.11 DCF MAC protocol discussed in class, which includes using acknowledgement, random wait, virtual sensing, and binary exponential back-off. The second is the 802.11 MAC protocol with the RTS/CTS option. The goal is to compare the performance of the two protocols and draw conclusions on which offers higher performance (at what situations).

The input to the programs is the same traffic file used in assignment No. 2. You can reuse but generalize your traffic generation program to produce different types of traffic. The first line of the traffic file contains a number telling the total number of packets in the file. After that, the traffic file contains lines of the formats

*pkt_id src_node dst_node pkt_size time*

that are sorted based on the *time*. Each line represents a packet of size *pkt_size* in bits from *src_node* to *dst_node* that is ready to be sent at time *time* in microseconds. The *pkt_id* is unique for each packet in the file. We will assume a single WiFi network with a single AP. As such, it is assumed that all packets are sent to the AP (and the *dst_node* in the traffic file is ignored). The minimal generalization of the random traffic generation program required in this project is to produce packets of random packet sizes following at least two different probability distributions (e.g. exponential distribution, uniform distribution, etc).

We will assume that all nodes run at a fixed data rate of 6Mbps. The physical layer is assumed to be perfect, i.e., there is packet loss unless there is a collision and nodes can sense the medium perfectly.

Your program should simulate the operations of the protocols for each node. For example, when a node has a packet to send at a time (read from file, retransmission, etc), the node should check if the medium is busy or not, and if busy, it should pick a random number as the back off counter, and wait until the medium is free for DIFS before starting to decrement the counter. For the details of the MAC, please refer to the course slides. A retransmission may push other packets that are ready to be sent

at a later time: packets that are ready to be sent, but have not been sent must be queued in order to be sent in the future.

In simulating the RTS/CTS mechanism, the virtual sensing for RTS should be until the end of CTS while the virtual sensing for CTS should assume the channel will be busy until the end of the ACK.

We will make one simplification in the simulator about the ACK timeout mechanism. If there is a collision, all nodes in the network will wait until the longest transmission completes (because collision occurs when more than one node started transmission at the same time and some may have longer packets than others), then all nodes in the network will wait for DIFS, then start to decrement the backoff counter for nodes with packet to send. Normally, after a node transmits a packet, it will wait for the ACK; and if ACK is not received after a timeout, it will start to wait for DIFS. Our simplification basically assumes that a node can determine whether a packet transmissions is good or not immediately after the packet is sent, which should have similar effect as the ACK timeout, but will simplify the implementation.

Details of the simulation parameters follow:

- Network speed: 6Mbps.
- ACK time: The ACK is sent by the AP after a packet has been correctly received. It takes 44 us.
- RTS/CTS packet time: 30 us.
- Slot: 9 us
- DIFS: 28 us
- SIFS: 10 us
- Before collision, random wait time: DIFS+[0-15] slots
- Initial Contention Window Size: 32
- Maximum Contention Window Size: 1024

Your program should produce a log for each node about important events with a time stamp. The events include *packet ready to send*, the *random backoff decision* (both initial before collision and for binary exponential backoff), *packet start sending*, *packet end sending*, *collision*, etc. A sample log file will be provided, your output should more or less match the sample for testing cases. In addition, your program should also output the following statistics:

- network throughput
- total number of transmissions
- total number of collisions
- fraction of time the medium is free

- the number of packets send for each station
- average latency of packets in microseconds for each station

After you finish your program, you must report a throughput/latency study of the two protocols with your conclusion of the comparative study. Note that in order for the plots to be statistically sound, you must do experiments with a sufficiently large number of packets.

**Grading Policy**

You can write the program in any language that is available on linprog, where your program will be graded. You must supply a README file describing how to run your program.

- Program compiles with no compiler error (10 points) -- programs with compiling errors get 0 points.
- README file, make file, etc (5 points)
- Assignment demonstration (15 points)
- Generalized traffic generator with 2 different packet size distributions (10 points)
- 802.11 DCF simulator (25 points)
- 802.11 DCF with RTS/CTS (25 points)
- Project report (10 points)
- For the simulators, your program will get 60% of the points for passing a basic test (similar to the sample traffic file). The reminding 40% will be subject to more extensive testing.
- Your report should resemble any scientific report of a Computer Science experiment: it first describes how things are done in the experiments (including how you simulate the protocols) and the setting of the experiments before reporting results. **Readers of a good report should be able to reproduce the results in the report (from the information given in the report) if they choose to repeat the experiments.**

**Deadline and materials to be submitted**

- Submit all files including readme, makefile, and project report through campus.fsu.edu by November 20. You will sign up for a demo time in the next class.

**Miscellaneous**

All programs submitted will be checked by a plagiarism detection software. Honor code policy will be strictly enforced. Write the code by yourself and protect your code.

This program is reasonably large with a lot of logics. Start early.