In-class Laboratory Exercise 7 (L07)

Part I – Objectives and Laboratory Materials

Objective

The objective of this laboratory is to:

- Analyze the effect of the virtual sensing mechanism, i.e., using request-to-send (RTS) and clear-to-send (CTS) signaling, on IEEE 802.11b performance.

After completing the assignment, you should be able to:

- Explain the drawbacks of the carrier sense multiple access with collision avoidance (CSMA/CA) protocol and explain how the virtual sensing mechanism helps to avoid them; and
- List the trade-offs in enabling the exchange of RTS and CTS frames and discuss under which circumstances RTS/CTS signaling provides the greatest benefits.

Hardware to be used in this laboratory assignment

- Xircom IEEE 802.11b PC Card adapter
- Dell notebook computer, with a fully charged battery

Software to be used in this laboratory assignment

- *iperf*, a traffic generation and network performance measurement tool, running under Linux on the notebook computer

Part II – Pre-Laboratory Assignment

This laboratory assignment requires coordination with other groups. It is important that you complete the entire pre-lab assignment before you come to class. It is not fair to other students if you are not prepared for the in-class experiments.

Write two scripts to configure the *iperf* server and client.

- Write a server script (*iperf_RTS_server*) that configures the *iperf* server to receive UDP datagrams at ports 5000, 5001, and 5002 and at 2 Mbits/s. The server script should read the datagram length from the command prompt.
- Write a client script (*iperf_RTS_client*) that configures the client to transmit UDP datagrams to the server at a rate of 2 Mbits/s for 60 seconds. The client script should read the server's IP address, port number and the length of UDP datagrams from the command prompt. The *iperf* executable is located in /usr/local/bin. Store the scripts in /root/WNMS/lab_7.

Part III – In-class Laboratory Assignment

We will study the effect of the RTS/CTS virtual sensing mechanism on nodes that are closely spaced and are within each other's transmission range (i.e., that can "hear" each other). The experiment involves the measurement of throughput achieved by each transmitter, transferring UDP data to one receiver. We will vary the packet size and observe the effect of the virtual sensing mechanism on link throughput.

1. Teams consisting of four groups (nominally eight students) will perform this experiment. One group will serve as the receiver (and will run the *iperf* server). The other three groups will act as

transmitters (and will run the *iperf* client). All groups must ensure that their notebook computers are within radio range of the other three groups on their team. To the extent possible, each team should separate themselves from other teams.

2. Set up an ad hoc IEEE 802.11b network, as described below.

   a) Using *iwconfig*, set each of the four nodes to the ad-hoc mode of operation. Set the ESSID to "wnms*nn*" where *nn* is the two-digit group number corresponding to the <u>receiver</u> of the connection (the group running the *iperf* server). For example, if Group 04 is acting as the receiver, the team should use ESSID wnms04. Set the channel of operation to the one assigned to your team by the laboratory instructor and set the rate to 2 Mbits/s on each notebook. Set the IP address of your machine, using *ifconfig*, to 169.254.1.*x* (255.255.255.0), where *x* is the number corresponding to your group. Set the RTS threshold to 2300 bytes (which effectively turns off the virtual carrier sensing). Bring up the interfaces as described immediately below.

   b) To create the ad-hoc network, the <u>receiver</u> node (the group running the *iperf* server) should bring up its interface first while the other three nodes do <u>not</u> bring up their interfaces. (The laboratory instructor can verify that the channel and ESSID configuration is correct using Netstumbler or a similar tool.) Once the receiver node is up, then the other three groups may bring up their interfaces. Again, do not bring up the interfaces on the sender nodes until the receiver node is properly configured and active.

   c) After all four nodes are up, check whether all the notebooks are within radio ("hearing") range of each other. Each node should be able to ping the other three nodes in the team. Capture a screen shot of the ping output obtained at the receiver by pinging each of the other three nodes and include it in your report. Check your channel of operation by inspecting the file /proc/driver/aironet/eth1/BSSList.[1]

3. At the notebook computer serving as the <u>receiver</u>, run the server script developed in the pre-laboratory assignment, *iperf_RTS_server*, in a terminal window by typing:

   ./iperf_RTS_server *length_of_datagrams*

   On the notebook computers serving as <u>transmitters</u>, run the client script developed in the pre-laboratory assignment, *iperf_RTS_client*, by typing:

   ./iperf_RTS_client *server_IP_address port_number length_of_datagrams*

   Each of the three transmitters should transmit to a different port on the server. (Ports 5000, 5001, and 5002 should be available.) Start with a datagram length of 50 bytes.

4. At the *iperf* server, take note of the throughput corresponding to the data exchanges from each of the three clients. Capture a screen shot of the throughput returned by *iperf* at the server to include in your report. Note that you need to capture the screenshot for the experiment with a data length of 50 bytes with both RTS/CTS disabled and RTS/CTS enabled (which is done in step 6 below). You do not need to capture a screen shot for experiments with datagrams of other sizes, but you do need to record results for all datagram sizes.

5. Repeat the test using UDP datagram sizes of 200 bytes up to and including 1400 bytes, increasing the datagram size in steps of 200 bytes. (Do tests with datagram sizes of 200, 400, 600, 800, 1000, 1200, and 1400 bytes). Note the throughput for each case.

[1] It has been observed that sometimes the file /proc/driver/aironet/eth1/BSSList sometimes does not indicate the available channels. If the channel information is not provided in this file, then you can examine the file /proc/driver/aironet/eth1/Config to see the configuration parameters such as the channel, data rate, etc. It does not show the available channels, but you can verify that you have configured the interface to use the correct channel.

6. Now we will study the effect of RTS packets as overhead on the throughput. Enable the virtual sensing mechanism by setting the RTS threshold to 20 bytes using *iwconfig*.
7. Repeat steps 2 through 5 and make a note of the link throughput returned by *iperf* at the server for each of the three transmitters. Also, capture a screen shot showing the results with 50-byte datagrams with RTS/CTS signaling enabled.

Be sure that all groups in a team have the required screen shots and the throughput measured for all datagram sizes with RTS/CTS signaling disabled and with RTS/CTS signaling enabled. Each group will analyze these results in the associated at-home exercise.