

This assignment will familiarise you with various aspects of the physical layer in wireless communications.

1 Instructions

First, login to ops.cmcl.cs.cmu.edu and copy all the files in `./xiaohui/emuNode` into `/emuNode`. Remember to `chmod +x *`. Read the script `hocStart` carefully and learn how to use the various commands to configure your wireless card. See the recitation presentation for details.

Also, try out the example in:

<http://www.cs.cmu.edu/haowen/15496/Recitation02/code/>

You can modify those files to complete this assignment. Take special note of how the java code re-initializes the wireless module on the nodes at the start of each experiment to ensure that the nodes are loaded with known parameters (e.g. the channels are reset, the mode is set to 802.11b, etc etc). This step is very important and you **MUST** do this for every experiment in this assignment.

Finally, note that part 4 of this assignment requires 4 nodes. You should attempt this assignment as early as possible since there is no guarantee of these nodes being available close to the deadline. We will not grant extensions based on node unavailability.

You are allowed to perform external research (e.g. browse the web, refer to the text, library etc) to gain information to answer the discussion questions. Please make sure you use reliable sources: e.g. not everything posted online is accurate.

For ALL experiments in this assignment, use the 802.11b mode.

2 Part A - Batch Data Collection in Java (2 points)

In assignment 1, we had semi-automated experiments where the nodes were moved automatically after a fixed time interval, but the user had to be prompted to run the appropriate iperf diagnostics on the nodes at the appropriate times.

In assignment 2 and 3, we will remove the need for interactivity by using the remote daemon invocations to run the relevant iperf processes from the java interface. In this experiment you will implement an automated version of the experiment you performed in Assignment 1, part C.

2.1 Experiment Description

Write the java file `experiment2a.java` to perform a series of iperf data rate measurements between two nodes at distances 55m, 65m, 75m, 85m, 95m, 105m apart. Your implementation should generate the entire set of raw data results in a single run, i.e. it should move the nodes into each of the correct positions and then remotely execute the relevant iperf commands on the nodes, directing the results of the experiment into one or more raw data files. The raw data files do not need to be formatted in any particular way as long as you, the student, can easily parse them (by eye or by script, up to you). Simply capturing the iperf output will be fine. On the nodes, your implementation should not require the user to do anything other than start and stop the `nodeDaemon` processes (you are also allowed to manually load the `rmiregistry` daemon at the start of everything of course).

2.2 Deliverables

The xml file `experiment2a.xml` and the java file `experiment2a.java`. The xml file should be basically identical to the one from part C of assignment 1, except that it loads the class `experiment2a` (instead of `experiment1c`).

A set of sample raw data files generated by your run of the above experiment. Only include files that were used to generate your actual experiment data (i.e. ignore error files and data files that you did not use).

Data file `experiment2a.txt` formatted similarly to the file in Assignment 1 Part C (i.e., columns of distance then data rate). These results should be extracted from the same raw data files that you submitted. Please include a brief explanation of how you extracted these results if you used a different procedure from the tutorial.

A graph `experiment2a.eps` of (Y axis) data rate (MBps) vs (X Axis) node separation (m) based on the data in the file `experiment2a.txt`.

2.3 Hints

1. Remember to ssh to the nodes to start up the remote java demons manually. You have to first execute `rmiregistry &` and then `nodeDaemon`.
2. Alternatively you can use the script `startDaemon <node>` to start the daemons on the nodes.
3. It is more convenient to capture the output from the server iperf rather than the client (which sometimes may not receive the final report from the server especially for connections with a high loss rate). Make sure you turn on periodic reports on the server side, at least once per second (`-i 1`).
4. You may either terminate each instance of the iperf server per run of the client (like in the example), or just keep one copy of the server resident throughout to service all of

your tests; either way has its own advantages.

5. When links are tenuous, during a UDP test, the iperf server often does not receive the end-of-connection message from the iperf client. Then when the next test comes in, it may treat both tests as a single connection. Make sure you leave enough time between tests to be able to disambiguate (you will see a bunch of times with no packets between the two tests; see the tutorial). More importantly, do not simply use the server-side reported average data rate for weak connections, because the server has no idea how long the test(s) went for. Instead, figure out the total number of packets that got through in each test and divide by the duration of the test (20s).
6. You should stop and restart the nodeDaemon process on all nodes, and make sure no processes started by the previous experiment are still resident (use ps and kill any iperf, java, etc still running) between runs of emuRun.

3 Part B - Rate-selection Schemes (6 points)

In the remaining parts of the assignment, you will be expected to write your experiments to collect all your data in a single batch, driven from the java interface, like what you did in part A.

Most of the 802.11 devices support multiple bit-rates by using different rate schemes (1Mbps, 2Mbps, 5.5Mbps, and 11Mbps for 802.11b). Which bit-rate can provide better link throughput depends on the current link quality. Higher bit-rates allow high quality links to transmit more data, but provide low throughput on low quality links. In part B, you will conduct experiments to study how different bitrate selection schemes perform under various link loss settings.

3.1 Experiment Description

For each rate setting (1Mbps, 2Mbps, 5.5Mbps, and 11Mbps and auto), measure the maximum UDP data rate between two nodes for link loss settings between 90Db and 110Db (inclusive) in 2 Db increments. Use 20s UDP iperf tests. Plot these data rates on a single graph and observe how the higher rate modulations perform worse for poor-quality links, and how well the auto rate selection adapts to link quality.

3.2 Discussion Questions

1. Explain briefly (informally) why different modulation/encoding schemes yield different data rates for different link qualities.

3.3 Deliverables

Text file `experiment2b.txt` containing the collated data of the experiment (i.e. a single table containing the average throughput for each rate scheme at each loss setting).

Graph `experiment2b.eps` showing the data from the text file, with data rate (Mbps) on the y-axis and loss (Db) on the x-axis, and one line for each rate setting (total 5 lines).

The script and java files `experiment2b.xml`, `experiment2b.java` which generated these results. The entire set of results should be generated by a single execution of `emuRun` on these files.

A file `experiment2b-discuss.txt` containing answers to discussion questions.

3.4 Hints

1. Remember to reset the nodes (down, then `hocStart`: see the example) before starting the experiment. This is particularly important since you must operate in 802.11b mode for this assignment rather than the default 802.11g.
2. When first switching into the auto rate selection mode, send some dummy traffic (a short `iperf` test with the results discarded is fine) on the link so that the algorithm can settle into the right rate to start with.
3. Use `sudo iwconfig ath0 rate <modulation>` (where modulation = "auto", "11M", "5.5M", "2M", "1M") to change the modulation on both sender and receiver nodes.

4 Part C - Auto Modulation Weaknesses (6 points)

In part B you probably observed that auto modulation did pretty well compared with the various fixed rate modulations. Now consider a link which periodically drops out; specifically, the link is set to 0 loss for 1s, then infinite loss for 0.2s, and then 0 loss for 1s, then infinite loss for 0.2s, and repeat. Compare the performance of auto rate selection vs fixed 11Mbps for this unstable link.

4.1 Discussion Questions

1. Explain why auto rate selection performs worse than fixed-rate selection with the above link behavior.
2. Suggest another kind of link behavior for which auto rate selection may behave even worse than this periodically dropping one. Explain why the rate selection algorithm might have even more difficulty with your link than the one above.
3. Under what scenarios would such links be likely to occur in the field?

4.2 Deliverables

The script and java files `experiment2c.xml`, `experiment2c.java` for the experiment, implementing the above link and performing iperf measurements using auto rate selection and fixed 11Mbps.

Text file `experiment2c.txt` showing the results of the experiment, i.e. the average data rate of auto modulation as well as the average data rate of fixed 11Mbps modulation.

A file `experiment2c-discuss.txt` containing answers to discussion questions.

5 Part D - Inter Channel Interference (6 points)

In the 2.4 GHz band used by the 802.11b networks, multiple channels are supported. Frequency reuse becomes important when people deploy large-scale wireless systems. When wireless devices are far away to each other, they can operate in the same channel without interfering to each other. When they are close to each other, they will share the channel with lower performance. To achieve higher performance, adjacent wireless devices can operate on separate non-interfering channels and minimise the interference among them. In part D, you will study the effectiveness of frequency reuse in the 802.11 networks.

5.1 Experiment Description

A NS script file with four nodes and a default channel of 6 is provided at the Recitation code location:

`http://www.cs.cmu.edu/haowen/15496/Recitation02/code/4node.ns`

Deploy four nodes A,B,C,D all of which can communicate with each other with 0 loss. Node C and D are the interfering pair; for the duration of the entire experiment, Node C sends UDP data as fast as it can to node D on channel 6 (a long-running iperf test is fine for this). Now set the channel for communications between A and B to channels 1 through 11 consecutively and measure the maximum UDP data rate for each setting.

5.2 Discussion Questions

1. What is the minimum channel separation needed for two close-by communications to have absolutely no impact on each other?
2. Explain why adjacent channels cause more inter-channel interference than widely separated channels.
3. Briefly discuss the implications of this observation on channel-division multiple-access MAC schemes in dense networks.

5.3 Deliverables

Data file `experiment2d.txt` containing the experiment results (the data rate between A and B for each channel setting).

Graph `experiment2d.eps` showing the plot derived from the data file: Y-axis: data rate, X-axis: channel of AB.

The script and java files `experiment2d.xml`, `experiment2d.java` for this experiment.

Answers to discussion questions `experiment2d-discuss.txt`.

5.4 Hints

1. To change the channel to channel N, you can use: `sudo iwconfig ath0 channel N`
2. To send a lot of data between the interfering nodes, you can just set up a long iperf test. On the client, use the `-t XX` option to run a test for XX seconds.
3. Remember to modify your `experiment2d.xml` to define all four nodes in the experiment instead of just two for the previous experiments.

6 Submission Instructions

Same as assignment 1.

To submit the solution, use the CMU Blackboard (<http://www.cmu.edu/blackboard/>) and select this class (S08 Special Topic: Hands on Introduction to Wireless Networks). Select Assignments, and under Assignment 2 select "View/Complete Assignment". Upload your files there with any comments you feel are useful or necessary. The assignment is due at 11:59pm on the due date.