Getting Started with DragonRadio in the Colosseum

Geoffrey Mainland
mainland@drexel.edu

November 2, 2021

The goal of this lab exercise is to get up-and-running with DragonRadio on the Colosseum and explore how different PHY and MAC parameters affect radio performance.

- Use the `dragonradio-main-20211102` image.
- Use the `dragonradio.py` radio script provided with this document.
- You may want to use one or more of the following options when invoking `dragonradio` to help you see what it is doing:
  - `-v` (verbose) Display information about what the radio is doing.
  - `-d` (debug) Display even more information about what the radio is doing.
  - `--verbose-packet-trace` Display each packet that is read/written from/to the network.

  If you use the `-d` option, the radio will report `Installing MAC schedule` when the MAC schedule has been installed, after which it will be ready to pass traffic.
- Some tools in this lab require X Windows, so you should either use the `-X` option to `ssh` or enable X forwarding in your `.ssh/config`.
- The `drgui` tool used in this lab has the ability to export what it displays as a PDF or PNG file (other export formats are supported too)—just look for the icon on the toolbar that looks like a disk.

1 Run DragonRadio on the Colosseum

1. Log in to the Colosseum website at https://experiments.colosseum.net.
2. Make a reservation with the `dragonradio-interactive` image for two SRNs. For the rest of the lab, we will refer to one of the nodes as the "receiver" and the other as the "transmitter"—assign these roles however you like.
3. When your reservation becomes active, use `ssh` to connect to your two nodes as the `root` user. The `root` password on the image is `dragonradio`. See Logging into an SRN for details on how to connect to a node with `ssh`.
4. Stop the instance of `dragonradio` that is running the the background on both nodes:

   ```
   service dragonradio stop
   ```

5. Copy the `dragonradio.py` radio script to the `/root` directory on both nodes.
6. While logged in to one of the containers, start the 0 dB path loss scenario by running the following command:

   ```
   colosseumcli rf start 1009 --cycle
   ```

   This will run the Test Scenario All Paths 0 db (1009), cycling it when it ends for continuous 0 dB connectivity between the nodes. You can check if the scenario is active with the following command:

   ```
   colosseumcli rf info
   ```

7. Change to the `/root/dragonradio` directory on both nodes.
8. Run the radio as follows on the receiving node:

   ```
   VIRTUAL_ENV=venv ./dragonradio ~/dragonradio.py -d -i 1
   ```

   On the **transmitting** node, run the radio as follows:

   ```
   VIRTUAL_ENV=venv ./dragonradio ~/dragonradio.py -d -i 2
   ```

   The `VIRTUAL_ENV` environment variable specifies the Python virtualenv that contains the Python packages needed by the radio. Instead of explicitly specifying `VIRTUAL_ENV`, you can instead activate the virtualenv before running the `dragonradio` binary.

   The `-i` options sets the node's identifier. We will always transmit from node ID 2 to node ID 1. The radio will not display a prompt—don't worry, it is waiting for traffic. The `-d` option turns on debug messages.

   Checkpoint When the radio starts and the `-d` flag is given, it will dump its configuration. What bandwidth does the radio use by default? Hint: look for the `bandwidth` configuration setting.
9. Make sure your radios can communicate with each other. When the radio starts up, it creates a tun interface with the IP address `10.10.10.N`, where `N` is the radio's ID. Any traffic sent to this interface will be transmitted by the radio. In a *separate terminal session* connwcted to the transmitter, use `ping` to make sure you can reach the receiver as follows:

   ```
   ping 10.10.10.1
   ```

   Checkpoint Confirm that `ping` works on the transmitter. Once you have done this, why do you not need to use `ping` on the receiver to make sure it can reach the transmitter? If you don't know what `ping` does, read the `man` page.
10. For the rest of the lab, we will use the `iperf` tool to benchmark the radio. On the receiver, run an `iperf` server as follows:

    ```
    iperf -s -u -i 1
    ```

    The `-s` option tells `iperf` to run in server mode, the `-u` options tells it to use UDP, and the `-i 1` option tells it to report bandwidth statistics every second.

    Run an `iperf` client on the transmitter as follows:

    ```
    iperf -c 10.10.10.1 -u -i 1 -b 200k -t 10
    ```

    The `-c 10.10.10.1` options tells `iperf` to run in client mode and connect to the server at IP address `10.10.10.1`, the `-b 200k` option sets the target bandwidth to 200Kbps, and the `-t 10` option specifies that data should be transmitted for 10 seconds.

    Checkpoint Make sure that `iperf` reports that it is receiving data on the receiving node. What data rate does it report? Why is this rate less than 200kps?

2 DragonRadio PHY

2.1 Viewing Constellations

DragonRadio can log both transmitted and received IQ data using the flags `-l logs --log-iq`. The first flag specifies that data should be logged in the `logs` directory, and the second flag tells DragonRadio to log IQ data. All data is logged to an HDF5 file, although you don't need to worry about this detail for the lab. After you shut down the radio (use `Ctrl-C` to shut it down gracefully), its logs will be in the file `logs/node-N/radio.h5`, where `N` is the radio ID, i.e., node ID 1 will write its logs to the file `logs/node-001/radio.h5`.

We will use the `drgui` tool included with DragonRadio to view logged IQ data.

1. Run DragonRadio on the receiver, enabling logging:

```
VIRTUAL_ENV=venv ./dragonradio ~/dragonradio.py -l ~/logs --log-iq -d -i 1
```

2. Run DragonRadio on the transmitter as before.
3. Run `iperf` on both nodes as before.
4. Gracefully shut down the radio on both nodes using `Ctrl-C`. It may take a few seconds for the receiving radio to shut down because it needs to ensure all logs are written to disk.
5. Open a new terminal and connect to the receiving node. Change to the `/root/dragonradio/tools` directory (using `cd`) and activate the Python virtualenv there as follows:

   ```
   source venv/bin/activate
   ```

6. Use `drgui` to view the packets received by the receiving node:

   ```
   drgui ~/logs/node-*/radio.h5 --rx 1
   ```

   If you get and error of the form `ImportError: Argument 0 does not allow None as a value`, then you have not properly set up X forwarding.

   If you get an error of the form `OSError: Unable to open file`, then you either have left the radio running or you have a corrupt HDF5 file. This probably means you did not gracefully shut down the radio, so it did not have a chance to close and flush the log file. Make sure you gracefully terminate the radio using `Ctrl-C` and wait for it to completely shut down.

   NOTE From now on, it is easiest to `rm -rf` the `logs` directory before you run the radio. The radio will never overwrite an existing log, so if you don't remove all logs, you will end up with log files `logs/node-001/radio.h5`, `logs/node-001/radio-01.h5`, `logs/node-001/radio-02.h5`, etc. `drgui` will load these logs, but loading them all can take a lot of memory, and it can be a hassle to figure out which log is the one you want—starting from a clean slate is easiest.

7. `drgui` allows you to cycle through each received packet and display several types of information, including the spectrum used by the packet, its constellation, and its power spectral density.

   Checkpoint Look at the signal spectrum and constellation corresponding to a received packet. What modulation scheme does DragonRadio use by default? Hint: look at the constellation. What is the default bandwidth of DragonRadio's transmission? Hint: look at the power spectral density plot, labeled "PSD."

2.2 Experimenting with Radio Parameters

Try running the radio with different parameters and see how high you can get the data rate to go. You will hopefully need to use a larger value for the `-b` parameter on the `iperf` client!

The two DragonRadio parameters that you should experiment with are `-m`, which sets the modulation scheme, and `-b`, which sets the bandwidth. Invoke DragoRadio with the `-h` or `--help` argument to see what the valid option for the `-m` parameters are.

Checkpoint Without modifying the bandwidth (that is, without using the `-b` option), how fast can you get the radio to go? Now experiment with the bandwidth using the `-b` option. Note that not all bandwidths are supported! For a fixed modulation, what effect does increasing the bandwidth have on throughput? Does it depend on the modulation scheme? Explain why this might be.

3 MAC

3.1 ALOHA MAC

The default MAC for DragonRadio is a TDMA MAC, but it also supports the ALOHA MAC. We will use DragonRadio's snapshot capability to look at the behavior of these two MACs.

Snapshots capture received IQ data over a user-defined window that defaults to 0.5 seconds. You can capture a 1 second snapshot every 1 seconds (continuous snapshots) with the following command:

```
-l ~/logs --log-iq --log-snapshot --snapshot-duration 1 --snapshot-frequency 1
```

1. Run DragonRadio using the modulation you found gave the best throughput, but add the `--aloha` flag ob both the receiver and transmitter. On the receiver, add the flags above to enable snapshot collection. Remember to remove the `~/logs` directory before you start the receiver.
2. Use `drgui` to look at the snapshots collected by node 1:

   ```
   drgui ~/logs/node-*/radio.h5 --snapshot 1
   ```

   You should see both transmitted and received packets. Spectrum containing transmitted packets will be marked by a transparent blue box, and spectrum containing received packets will be marked by a transparent red box. Since snapshots are collected as soon as the radio starts, you may need to skip several snapshots before you see anything by clicking the "Next" button.

   Checkpoint What do the round-trip times for your pings look like when using the ALOHA MAC? Is this what you expect?
3. Run the experiment with the standard TDMA MAC, i.e., run the same experiment again, but without the `--aloha` flag.

   Checkpoint What do the round-trip times for your pings look like when using the TDMA MAC? Is this what you expect?

3.2 Modifying the TDMA Schedule

The TDMA schedule is constructed by the `pureTDMASchedule` function in `~/dragonradio.py`. If you run `dragonradio` with the debug flag, `-d`, it will dump diagnostic information, including the TDMA schedule.

1. Experiment with different TDMA schedule. For example, try a schedule with 10 slots where each node gets every other slots vs. a 10 slot schedule where node 1 gets the first 5 slots and node 2 gets the second 5 slots. Don't worry about the general case of N nodes: assume you only have two nodes. It is easiest to use `np.array` to construct a schedule. You will need to modify the schedule in the same way on *both* nodes.
2. Also experiment with the `--slot-size` argument to `dragonradio`, which specifies the TDMA slot size. Note that you must use the same slot size on all nodes!

Checkpoint Use `drgui` to visualize snapshots of `ping` and `iperf` traffic for different TDMA schedules and different slot sizes. We recommend that you vary only one parameter at a time: either the schedule or the slot size. It is perfectly fine to use the default slot size and experiment only with different schedules, and then use the standard "fair" TDMA schedule for all your slot size experiments. How do different schedules affect ping time What would you expect the *average* ping time to be for different schedules? What do you find experimentally? How do different schedules and slot sizes affect `iperf` throughput.

4 Clean Up

When you're finished, clean up as follows:

1. Save any code or data you want to keep by copying from your containers to a safe place.
2. Stop the RF scenario by executing the following command on either node:

   ```
   colosseumcli rf stop
   ```
3. Stop the reservation on the Colosseum web site.