**Mininet-WiFi exercises (Part 2)**

**General tips:**

* Disable Linux's built-in network-manager (***should be done every time the VM starts***):
  + Linux CLI: **sudo systemctl stop NetworkManager.service**
* Saving Mininet-WiFi output (but unfortunately not your input) in a file (here **outputfile.txt**):
  + Linux CLI: **sudo python *myscript*.py 2>&1 | tee outputfile.txt**
* Resetting Mininet-WiFi (*good practice if the emulator terminated incorrectly*):
  + Linux CLI: **sudo mn –c**
* You can use your keyboard's up and down arrow keys to quickly scroll up and down in your command history in the Linux CLI and the Mininet-WiFi CLI.

For the steps in this part of the Mininet-WiFi exercise, you’ll need to base your custom network topologies on a different template file, so please start by downloading the file **skeleton\_part2.py** from DTU Learn and use this as template.

**Step 1:**

Create the network in Figure 1 in Mininet-WiFi by adding statements to a copy of the template file. The access point should be an 802.11a access point (i.e., in the 5 GHz band) configured to use channel 36. When you create the stations, you should include the argument **mode = 'a'** to indicate that they should also use 802.11a mode.



Figure 1

Use the iperf command in Mininet-WiFi to measure:

1. The throughput between the two wireless stations
2. The throughput between a wireless station (iperf client) and a wired host (iperf server)
3. The throughput between the two wired hosts

Comment on the results, especially the difference between the three different measurements. If you get "strange" results in c), you may need to specify the bandwidth on the wired links explicitly by adding the **bw=100** argument to the **net.addLink()** statements to specify that the links are 100 Mbit/s. You can ignore the warnings ("… quantum of class …").

**Step 2:**

Copy your \*.py file from step 1 to a new name and make the following changes in the copy:

1. In the first line of the topology() function, change **wmediumd\_mode=interference** to **wmediumd\_mode=error\_prob**
2. Add the following statements immediately after the line with **net.configureWifiNodes()**:  
   **net.addLink(sta1, ap1, error\_prob = 0.0)  
   net.addLink(sta2, ap1, error\_prob = 0.0)**

Start Mininet-WiFi with the new file and verify that all wired/wireless stations can communicate with the **pingall** command. Note that you might need to execute this command a couple of times since the wireless stations are not associated with the access point immediately after the start of Mininet – or you can wait about 15-20 seconds.

Use iperf to measure the throughput from sta1 (as iperf client) to h1. Repeat the throughput measurements when the **error\_prob** argument in point b) above is set to 0.001, 0.003, 0.01, 0.03, and 0.1, respectively. Unfortunately, you must exit Mininet-WiFi, edit the \*.py file with a new value, and restart Mininet-WiFI between every value of the **error\_prob** argument. Make five measurements (i.e., execute the iperf command five times on the client) for every value of **error\_prob** so that you can calculate an average value.

Make a plot of the average throughput as a function of the error probability and comment on the results.

*Note that the iperf command may fail and cause the emulation to abort if there are too many errors when you execute the iperf command. In this case, you should reset the emulator* (***sudo mn -c***)*, restart it, and attempt again.*

**Step 3:**

Create the network in Figure 2 in Mininet-WiFi. The access points should use the 802.11a variant on channel 36, as in the previous steps. If you use a copy of your topology from the previous step as the basis, you should change the **wmediumd\_mode** argument back to **wmediumd\_mode=interference** (as in step 1) and remove the **net.addLink()** commands for the wireless links.



Figure 2

The two access points' coverage areas are disjoint so that when Sta1 moves (as indicated by the dotted arrow) from the left BSS to the right BSS, it will be out of coverage for some time. You may need to experiment with the positions of the two access points so their coverage areas do not overlap and you have time to execute commands on Sta1 when it’s between the coverage areas.

Execute the network in Mininet-WiFi and document in your journal:

1. That Sta1 is initially associated[[1]](#footnote-1) with AP1 and that it can ping h1.
2. That Sta1 is not associated with any AP when it is between the BSSs.
3. That Sta1 is eventually associated with AP2 and can (again) ping h1. You will sometimes have to wait about 60 seconds before the flow through AP1 between Sta1 and h1 times out in the switch, before you can ping via AP2. If, after more than 60 seconds, you are still unable to ping between Sta1 and h1, use the **dpctl del-flows** command in the Mininet-WiFi CLI.

Tip: First, create the network without mobility and test that Sta1 can ping h1. *Then,* add Sta1's mobility.

Notes:

* Mininet-WiFi attempts to animate the movements of Sta1, but it is not always smooth, so you might need to repeat the experiment if you want to execute a command on Sta1 when it is, e.g., between the coverage areas.
* If you experimented with mobility, you may get Python-related error messages (which can be ignored) in the Linux console when you exit Mininet-WiFi.

1. See step 1 in last week’s exercises on how to check if a station is associated with an access point. [↑](#footnote-ref-1)