Experimental evaluation of the performance of a 802.11 wireless network

1. Tutorial goals

After this tutorial students should have acquired enough skills to

1) configure a wireless network composed of Linux only devices powered by open-source software
2) assess the performance of the network

2. Tutorial steps

1) **Connect to Alix nodes.** All Alix nodes run a Linux kernel and a stripped down version of the GNU operating system called OpenWRT. As such the best way for accessing and managing them is through the console. You can either connect to the Alix box using a serial cable or log in through a Secure SHell (SSH) session. This second way is preferred: to this end run the following command from a Linux system connected to the same (wired) network providing connectivity to the Alix nodes:

```
$: ssh root@ALIX-WIRED-IP-ADDRESS
```

NOTE1: in the following use always the password "quaquaOCA2011" when requested.
NOTE2: in the following change ALIX-WIRED-IP-ADDRESS with the corresponding Wired IP reported on its top case.
NOTE3: you enter as root, having all privileges to remove whatever file on the filesystem. Pay attention to not destroy it.

Once inside you can navigate through folders like with any other Unix-based system. Please note that not all commands you might have used in the past are available and that some of those you already know can behave differently as they are not the original but they are emulated by a multi-binary software called "busybox".

2) **Bring up the Wireless Network Interface Card.** The Wireless Network Interface Card (NIC) is connected to the main system bus through a PCI bus. To check if the board is correctly seen by the system run the following command:

```
$: lspci | grep -i broadcom
```

You should read a few lines describing the hardware revision. Please note that the piece of kernel that drives the Wireless NIC is not loaded by default to avoid issues at boot if for some reason the firmware fails to load. For this reason the NIC is not active after power-up and simply it does not run any firmware. To load the module and check it was correctly attached to the Linux kernel run the following

```
$: insmod b43 qos=0
$: dmesg | tail -20
```

where the first command actually loads the module, the second shows the last twenty lines of the kernel log.

NOTE: always load the module with the switch "`qos=0`" otherwise it won't work.

At this point the Wireless NIC is still not working: bring it up by running

```
$: ifconfig wlan0 ALIX-WIRELESS-IP-ADDRESS up
```

where `ALIX-WIRELESS-IP-ADDRESS` is the wireless IP address also reported on the top case.

3) **Practice with traffic captures.** Though the interface is now running, it could not ping any other device. As told during the seminar, in fact, connectivity in an 802.11 network is active only when some nodes configured as "Stations" (STAs) are associated to a single node configured as "Access Point" (AP), so that they form the Service Set which can be thought as a "virtual" datalink segment. In this case the NICs of the corresponding STAs are said to be in "managed" mode while the NIC of the AP is said to be in "master" mode. Before learning how to configure nodes, we will first study how to capture traffic using the wireless interface. To this end we have to first create a "virtual" interface on top of the physical one using the following command:

```
$: iw dev wlan0 interface add fish0 type monitor
$: ifconfig fish0 up
```

Then you can capture the traffic and show it on the console using the tcpdump software

```
$: tcpdump -i fish0 -nn
```

To display the content of each frame add to the command the switch "`-XXX`". To increase the number of captured bytes to N, use switch "`-s N`": setting N to zero displays the whole frame.

Questions:

a. Which kinds of frame are being displayed by tcpdump?
b. Try to figure out how to selectively display only Beacons, based on the fact the for such frames the first byte is always 0x80: dig on Google how to specify filters with tcpdump (suggestion: look for BPF expressions).
c. Are you capturing beacons transmitted by AP nodes configured on specific channels? On which frequency? Try to change the frequency of the NIC by using the command "`iwconfig`" as the tcpdump command is running by connecting to the node using a second SSH session: check with Google how to do it (suggestion: "`iwconfig wlan0 channel …`").
    i. Try different channels, e.g., start with 6, then 11, 12, 13 and 14: do you feel traffic is decreasing as you get closer to channel 14?
d. Focus now on data packets (first byte either 0x08 or 0x88): is it possible to understand what payload they transport above MAC layer?
e. Save now the capture to a file using switch "`-w /tmp/filename.pcap`" but *pay attention* to save it inside the `/tmp` folder as suggested otherwise you risk to damage the filesystem: use a filter to capture only data frames and capture at most 1000 frames (use switch "`-c 1000`"). Once the file is captured, use an external laptop and copy the file to it using Secure Copy (if you don't know how to do, please ask the lab assistant for help).
f. Run Wireshark on the laptop and open the copied file, explore each data frame exploding the various section displayed by Wireshark
    i. Do you find any evidence of packets that might have been retransmitted?
    ii. Is there any mechanism that can be used by receivers to avoid receiving the same frame multiple times? Look for a sequence counter inside each frame: study the evolution of the sequence counters for consecutive frames transmitted by the same source.

4) **Create your first BSS.** Choose one node as AP: connect to it, load the b43 driver and bring up the wireless interface, assign a free IP address (if you have doubts ask the lab assistant for help). Start editing the file named "hostapd.conf" that is a simple example for configuring a Basic Service Set: to this end you can use command "`vi hostapd.conf`", that starts a user "unfriendly" editor. If you have trouble with `vi`, try first with Google (`vi` is weird if compared to any recent editor), then ask the lab assistant. Choose a name for your network (agree with your colleagues in other groups to avoid using the same name) and start using channel 14 so as to avoid external traffic interfering with your tests. Start the software hostapd using the file just edited so that the NIC can start beaconing (i.e., advertising the network you are creating)

```
$: hostapd -B hostapd.conf
```

From this point on, the kernel will pass all management frames like association/authentication requests directly to hostapd that will in turn process them and add stations to the list (maintained in the kernel) of associated stations. Verify it's running by issuing command "`ps`" and checking if the process name appears in the list. Check also the kernel log (especially the last 20 lines) by running command "`dmesg | tail -20`".

a. Which information is displayed in the log? Try to provide an answer ☺

5) **Add a station to the BSS.** Connect to another node with SSH: we will call this node STA. Before connecting to the BSS we just created, let's first load the b43 module, bring the wireless interface up, and assign the IP address chosen for the wireless interface (repeat what done in the previous steps). Let's also check whether the station senses the newly created BSS or not: run the command

```
$: iwlist wlan0 scan
```

and look if the BSS's name you configured in the AP is reported. If you don't find it, run the same command again.

Like on the AP, on a station we need an application for managing its association to a specific BSS, we will use wpa_supplicant. Edit the file "wpasupplicant.conf" and replace the name of the network you want to join with that configured on the AP. Then run

```
$: wpa_supplicant -B -i wlan0 -c wpasupplicant.conf
```

If you didn't get any critical error, then the station should be associated. Try pinging the IP address of the AP from the STA, and the IP address of the STA from the AP (use "`ping`" from the command line).

In the following we will need a couple of stations and also a separate monitor node (which we will eventually use as third station): to this end configure a third node as a station, assign it a new IP address and add it to the same BSS. Prepare also a monitor node: create a monitor interface, remember to set it on the same channel (14) of the BSS. Leave this node not joined for the moment. The scenario we will refer is the one reported in Figure 1.

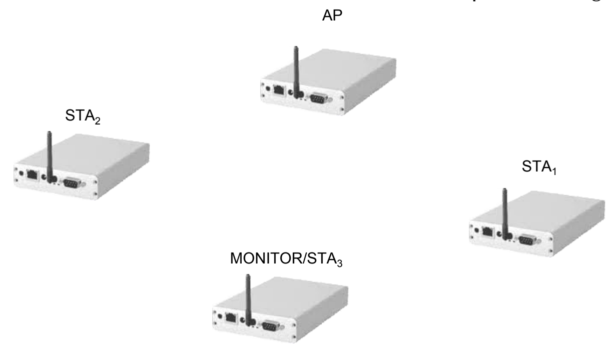

Figure 1 Reference scenario for today's experiments

6) **Connectivity test.** To verify everything is working as expected, start capturing traffic and use tcpdump on the monitoring node: create a filtering rule so that it captures both ICMP packets and control ones (remember that acknowledgment are control traffic starting with 0xD4: check with Google how to specify multiple rules are joined as logical and).

   a. Determine how much time passes between a frame and its corresponding acknowledgment. Actually for an accurate time measurement it is necessary to save the traffic to a pcap (remember to store everything in `/tmp`) and analyze it using Wireshark: in this way it is possible to use the MAC timestamp captured by the wireless NIC and stored inside the Radiotap Header which can be explored only with Wireshark (not with tcpdump).
   b. To check that a Short InterFrame Space (fixed to 10μs for 802.11b/g) separates each data frame and its corresponding acknowledgment you should also evaluate the transmission time of the data frame. To this end you should consider the following rules (important, study accurately as you will need this also in the future)
      i. If a frame is transmitted using one of the DSSS encodings (MCS = 1Mb/s, 2Mb/s, 5.5Mb/s and 11Mb/s), then the frame is composed of a preamble of duration either 192μs or 96μs (Wireshark reports whether long or short preamble is used), followed by a payload that is transmitted at the corresponding datarate, so takes the length in byte, multiply by 8 and divide by the datarate to compute the number of microseconds, then sum the duration of the preamble.
      ii. If a frame is transmitted using one of the OFDM encodings (MCS = 6, 9, 12, 18, 24, 36, 48 and 54Mb/s) the duration in microseconds is given by 20 + 4 * S, where S is the number of OFDM symbols composing the transmission. To determine S, first compute the number of bits BITS (length of the frame in byte multiplied by 8), then sum 22 and divide by the number of bits per symbol NDBPS, which depends on the MCS (specifically for increasing MCS we have NDBPS = 24, 36, 48, 72, 96, 144, 192, 216).

      Pay attention: consider the total number of bytes in the frame reported by Wireshark: please double-check your findings with the lab assistant.
   c. Finally take a deeper look at both the dataframe (ICMP packet in this case) and the acknowledgment and takes notes about the composition of the various lower headers. Double-check your findings with the lab assistant.

7) **Throughput tests with rate controller.** For measuring pure throughput performance we will use an application that can generate greedy traffic so that we can saturate the wireless link. Use Google to study the command line of `iperf` which we briefly saw in the seminar.

   Start `iperf` server on the AP, ask for reporting throughput every second, and configure it for using UDP as transport

```
$: iperf -s -u -i 1
```

On the station start `iperf` client and configure it to send greedy traffic to the IP address of the wireless interface of the AP, report statistics every second, use UDP as transport, and run for 1000 seconds (that means until you stop it).

```
$: iperf -c IP_OF_AP -u -i 1 -t 1000 -b 54M
```

The last switch "`-b BMAX`" can be used for limiting the bandwidth at BMAX. As we are using 802.11g, we can not exceed the nominal maximum 54Mb/s, so this asks for saturating the wireless link.

a. Determine the maximum throughput that can be achieved by the `iperf` session. To this end you should kindly ask to your colleagues in the other groups to stop transmissions for a while so that you can use the entire channel for this test.
b. In the meanwhile run tcpdump and filter UDP frames only: what is the MCS chosen for the transmission? Filter then the acknowledgments that the AP sends to the STA (how to do it? Create a filter including the initial 0xD4 ack byte and the MAC address of the STA, if you have problems ask the lab assistant): what is the MCS chosen by the AP for these ack frames? Who is actually choosing the rate of the ACKs? Can be the kernel involved in this operation? Double-check your findings with the lab assistant.
c. Explore the debugfs at the STA for the AP node: in particular take a look to the rc_stats table (please refer to the slides of the seminar: suggestion, `cd` to `/sys/kernel/debug/ieee80211/phy0/netdev:wlan0/stations` then?) Does the rc_stats table confirm what you observed with tcpdump?
d. Explore the debugfs at the AP for the STA node: what kind of information do you get?

8) **Throughput tests with fixed rate** Repeat the previous measurements by fixing the MCS on the station. Forcing a given MCS disables the rate control algorithm: all frames are transmitted and likely retransmitted after collisions with same fixed rate. To fix the rate at 6Mb/s run the following:

```
$: iwconfig wlan0 rate 6M
```

Please note that it is not possible to fix the rate at the AP side using a similar command: you can do it but you have to change the hostapd configuration file; we will see this in the future. Repeat the transmission experiment with `iperf` and verify both with tcpdump and by checking the debugfs (rc_stats table) that the rate is actually fixed to 6Mb/s. Check out the MCS of the acknowledgment frames, you will need it hereafter. By following the same approach as in **Connectivity test.** To verify everything is working as expected, start capturing traffic and use tcpdump on the monitoring node: create a filtering rule so that it captures both ICMP packets and control ones (remember that acknowledgment are control traffic starting with 0xD4: check with Google how to specify multiple rules are joined as logical and). try to compute the maximum throughput that can be achieved with the chosen MCS. To this end you have to take into account

a. the duration of a single UDP frame;
b. the duration of a single acknowledgment;
c. the SIFS between the data frame and the acknowledgment;
d. the average time spent by the transmitter before accessing the channel: consider a value of the contention window (minimum) set to 32, and a slot duration of 20μs.

Remember that for having a good match between the experimental measurement and the computed throughput it would be better to avoid interference from neighboring network, so please ask your colleagues from other groups to stop experiments for a while.

9) **Experiments with impaired channel conditions** Try now to keep the STA further away from the AP: to this end ask the lab assistant a longer network cable so that you can move the STA, if needed also outside the lab while maintaining connectivity. You will need also to plug the node to a battery. Switch the node back to automatic MCS choice (check with Google how to do that using "`iwconfig`"). Repeat the throughput experiments and try to determine the maximum throughput and the maximum MCS that is chosen by iperf. If it is still high (e.g., like 36Mb/s or more) move the node further away. Once you decided for a given distance and a given MCS, reconfigure the STA from scratch (e.g., kill wpasupplicant and restart it): then fix the rate to the one immediately faster than that you just determined. Run a capture to observe what happens and examine the iperf statistics at the AP

a. What's going on?
b. Is the iperf session experiencing losses? Is the trace giving evidence of losses? Check the retransmission bit (for this you need to capture the trace and analyze it with Wireshark). Finally check the rc_stat table and try to count how many losses did you experience.
c. Restart again the station and fix the rate to the one it was chosen before by the rate controller: what happens?

d. Finally put back the rate controller to auto: is the throughput now higher or lower than in the previous point? Why?

10) **Experiments with TCP** Repeat the performance assessment experiments with iperf but this time use TCP instead of UDP. Simply remove the "`-u`" at both the server and client invocation. Remember also to not use switch "`-b XX`" at client.
    a. Is the throughput increasing or decreasing? Try to describe why.
    b. Try to compute the average throughput analytically following what you did for the UDP case.

11) **Multi node fairness experiments** Join two stations to the BSS, place them at approximately the same distance to the AP. Run two iperf UDP sessions to the AP and observe the throughput experienced by each session.
    a. Is there any long-term fairness? What about short-term?
    b. Is the total throughput equal, greater or smaller than that experienced by a single station (alternatively stop iperf running on one station so that you can determine the throughput of the other station alone).
    c. Try to quantify losses using the debugfs at each sender: do you get similar statistics?
    d. If not, try to exchange the position of the two nodes and repeat the experiments.
    e. Add a third station to the BSS and repeat the experiments, verifying if losses increase linearly with the number of nodes or not.