UNIVERSITY OF TRENTO
Dipartimento di Ingegneria e Scienza dell'Informazione
Laboratory of Nomadic Communications

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 devices powered by open-source software
2) assess the performance of the network

2. Tutorial steps

1) **Connecting to the Alix nodes.** All Alix nodes run a very compact image of the Ubuntu 10.04 Lucid Server operating system and have installed a set of command-line tools for editing and compiling the firmware. As such the best way to access and manage them is through a system console. You can either connect to the Alix box using a serial cable or log in through a Secure SHell (SSH) session. The second method is preferred: to this end run the following command from a Linux system connected to the same (wired) network that provides connectivity to the Alix nodes:

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

   NOTE 1: in the following use always the password "firmware" when requested.
   NOTE 2: in the following change ALIX-WIRED-IP-ADDRESS with the corresponding Wired IP address listed on the top of each enclosure.
   NOTE 3: you will connect as a non-root user, so that the chances you accidentally erase important parts of the file-system are reduced. However you can run some privileged commands by simply preceding them with the "sudo" command. If you cannot run a command that you really need, ask the facilitator.
   Once you have logged in, you can navigate through folders like with any other Unix-based system. Please note that not all the commands you might have used in the past are available, and you do not have enough privileges to run "apt-get" or "dpkg". If you believe you need a particular tool, please ask the facilitator.

2) **Bringing 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 recognised correctly 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 time if for some reason the firmware fails to load. For this reason, the NIC is not active after power-up and it simply does not run any firmware. To load the driver module and check it was correctly attached to the Linux kernel, run the following:

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

In the above, the first command loads the module, while the second shows the last twenty lines of the kernel log, which may be useful for debugging purposes

NOTE: always load the module with the "`qos=0`" switch, otherwise the driver will not work.

At this point the Wireless NIC will still be not working, so you will bring it up by running

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

where `ALIX-WIRELESS-IP-ADDRESS` is the wireless IP address listed on the top of the enclosure.

3) **Practicing with traffic captures.** Though the interface is now running, it will not have network layer connectivity to any other device (a `ping` command to a given destination will not return a reply). As explained in the first lecture, 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" data link 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 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
```

After this step you should be able to capture Wi-Fi traffic and display the contents of packets on the system console, using the `tcpdump` tool, as follows

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

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

Questions:

a. What type of frame are being displayed by `tcpdump`?
b. Try to figure out how to selectively display only Beacons, based on the fact that for such frames the first byte is always 0x80. You may search on the web for how to specify filters with `tcpdump` (tip: look for BPF expressions).
c. Are you capturing beacons transmitted by the AP nodes configured on specific channels? On which frequency? Try to change the frequency of the NIC using the command "`iwconfig`" as the `tcpdump` command is running. To do this, connect to the node using a second SSH session. Check the command manual page online (nodes have been installed without man pages) for how to change the frequency (tip: "`iwconfig wlan0 channel …`").

   i. Try different channels, e.g. start with 6, then 11, 12, 13 and 14. Can you detect how traffic decreases as you operate closer to channel 14?

d. Focus now on data packets (first byte either 0x08 or 0x88). Is it possible to understand what kind of payload they transport above the MAC layer?
e. Now save the capture to a file using the switch "`-w /path/to/filename.pcap`", but *pay attention* to stop the capture at some point, otherwise the filesystem would likely fill up. Use a filter to capture only data frames and capture at most 1000 frames (use the switch "`-c 1000`"). Once the file is saved, copy it to an external laptop using Secure Copy (if you don't know how to do this, please ask the facilitator for help).
f. Start `Wireshark` on the laptop and open the copied file, explore each data frame expanding the various sections 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 number inside each frame: study the evolution of the sequence numbers for consecutive frames transmitted by the same source.

4) **Creating your first BSS.** Choose one node as AP. Connect to it, load the b43 driver and bring up the wireless interface, assign the IP address on the enclosure (if you have doubts ask the facilitator for help). This is a simple example for configuring a Basic Service Set: create and edit the "`hostapd.conf`" file using the command "`vi hostapd.conf`", which will launch an user "unfriendly" editor. If you have trouble with `vi`, try first with Google (`vi` may seem strange as compared to any recent editor), and if you get stuck ask the facilitator for help. 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 traffic from other networks in the building interfering with your tests.

```
# sample hostapd file
interface=wlan0
driver=nl80211
dump_file=/tmp/hostapd.dump
ctrl_interface=/var/run/hostapd
ssid=YOUR_NETWORK_NAME
hw_mode=g
channel=14
beacon_int=100
auth_algs=3
wpa=0
rts_threshold=2347
# supported_rates=60
# basic_rates=60
```

The last two lines are commented and if used would tell `hostapd` to use only the listed datarates, in this case only 6Mb/s (actually they are datarates multiplied by ten). Start the `hostapd` tool using the file just edited as configuration, so that the NIC can start beaconing (i.e. advertising the network you are creating)

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

Switch `-B` is optional and can be used to start the tool in background: for this first experiment start it in foreground so that you can have a look to the log. From this point on, the kernel will pass all management frames, such as association/authentication requests directly to `hostapd`, which in turn will process them and add stations to the list (maintained in the kernel) of associated stations. Check the kernel log (especially the last 20 lines), by running the command "`dmesg | tail -20`".

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

5) **Adding 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 was done in the previous steps). Let's also check whether the station detects the newly created BSS or not. For this, execute the following command

```
$: iwlist wlan0 scan
```

and check if the BSS name you configured on the AP is reported. If you do not find it in the list, run the same command again.

Similar to the AP, on a station we need an application for managing its association to a specific BSS. We will use the `wpa_supplicant` tool for this. Create and edit the file "`wpasupplicant.conf`" and replace the name of the network you want to join with that configured on the AP.

```
ap_scan=1
network={
  ssid="YOUR_NETWORK_NAME"
  key_mgmt=NONE
}
```

Then run

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

Again switch `-B` is optional and if used tells the tool to go into background. At least during the first attempts, run the tool in foreground and check the logs.
If you did not observe any critical error, then the station should be associated. Try ping-ing 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 to set up 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. **If needed, share this experience with another group**. Then prepare a monitor node by creating a monitor interface on an additional device. Remember to set this on the same channel (14) of the BSS. Do not associate this node to the BSS for the moment. The scenario we will refer to in this example is the one illustrated in Figure 1.

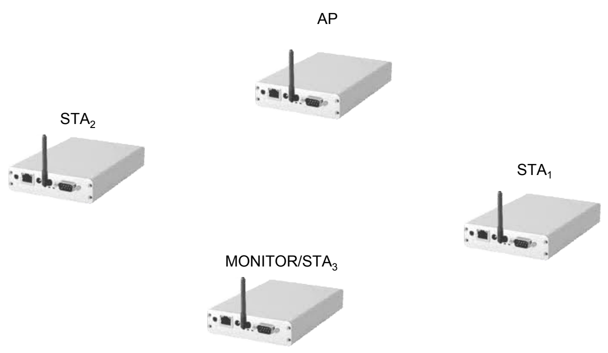

Figure 1. Reference scenario for today's experiments

6) **Connectivity test.** To verify that everything is working as expected, start capturing traffic and use `tcpdump` on the monitoring node. Create a filtering rule to ensure `tcpdump` captures both ICMP and control packets (remember that acknowledgments are control frames and their first byte is 0xD4. You can check online how to specify multiple rules in a joined fashion).

   a. Determine how much time passes between a frame and its corresponding acknowledgment. For accurate time measurements it is necessary to save the traffic to a pcap file (remember to store the capture at some point) and analyse this using `Wireshark` on your laptop. You first have to copy the pcap file using "scp". 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`). Ask the facilitator for help if experiencing difficulties.
   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 (it is important to read these carefully, as you will need them again in future experiments)
      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 a long or a short preamble was used), followed by a payload that is transmitted at the corresponding data rate. Therefore take the length expressed in bytes, multiply by 8 (to obtain the number of bits) and divide by the data rate to compute the number of microseconds spanned by the payload (round to the next integer). Then sum this with 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 + 6, where 20µs is the duration of the preamble (PLCP + SIGNAL symbol), S is the number of OFDM symbols composing the transmission, each of 4µs, and 6µs is the duration of the mandatory pause after each frame. To determine S, first compute the number of bits BITS (length of the frame in bytes multiplied by 8), then sum 22 (16 bit for the SERVICE field, 6 bit for the mandatory tail padding) and divide by the number of bits per symbol NDBPS,

which depends on the MCS. More specifically, for increasing MCS equal to 6,9, 12, 18, 24, 36, 48 and 54Mb/s we have NDBPS = 24, 36, 48, 72, 96, 144, 192, 216. Finally, approximate to the next integer.

Pay attention: consider the total number of bytes in the frame reported by `Wireshark`. Double-check your findings with the facilitator.

c. Finally, take a close look at both the data frame (ICMP packet in this case) and the acknowledgment and take notes about the composition of the various headers.

7) **Throughput tests with a rate control algorithm active.** For measuring pure throughput performance we will use an application that can generate greedy traffic so that we can saturate the wireless link. For this, we will use the `iperf` tools.

Start the `iperf` server on the AP, configuring throughput reporting with a 1 second granularity, and the use of UDP as the transport protocol

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

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

```
$: iperf -c WIRELESS_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 cannot 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 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 this? Create a filter including the initial 0xD4 ack byte and the MAC address of the STA. If you experience problems ask the facilitators for help). What is the MCS chosen by the AP for these ack frames? Who is actually choosing the rate of the ACKs? Can the kernel be involved in this operation?
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 used during the technical session. Tip: `cd` to `/sys/kernel/debug/ieee80211/phyN/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 the 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 could do it but you would have to change the `hostapd` configuration file (the last two lines commented in the above example). Repeat the data transmission experiment using `iperf` and verify, both with `tcpdump` and by checking the debugfs (rc_stats table), that the rate is actually fixed to 6Mb/s. Check the MCS of the acknowledgment frames, you will need it hereafter. By following the same approach as in the connectivity test above (6), 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 16, and a slot duration of 20μs.

Remember that for having a good match between the experimental measurements and the computed throughput, it would be better to avoid interference from neighbouring networks, 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 use a longer network cable so that you can move the STA, while maintaining connectivity. You can also try to impair channel conditions by simply rotating the antennas of the AP so that their polarization does not match that of the stations. Another possibility is to reduce the power emitted by a station (check online command "`iwconfig -i wlan0 txpower…`"). Switch the node back to automatic MCS choice (check online 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 or further reduce the station power. Once you settled for a given distance and a given MCS, reconfigure the STA from scratch (i.e. 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 you experienced.
   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 the one observed at 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`" when in invoking both the server and the client. Also, remember not to use the switch "`-b xx`" at the client side.
    a. Is the throughput increasing or decreasing? Try to explain 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 from the AP. Run two `iperf` UDP sessions to the AP and observe the throughput attained 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` on one station, so that you can determine the throughput of the other station when transmitting alone).
    c. Try to quantify the 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, checking whether the loss rate increases linearly with the number of nodes or not.