Exercises for Security of Wireless Networks

edited by
Luka Mališa, Aanjhan Ranganathan, Joel Reardon and Nikos Karapanos

Course responsible:
Prof. Dr. Srđan Čapkun

Assistants on the course:
Luka Mališa
Nikos Karapanos
Hildur Olafsdottir

Email: sown@lists.inf.ethz.ch

Contents

Preface

The booklet is undergoing a major revision this semester as a substantial amount of new content is being introduced. As you, students, are our targeted audience, we kindly ask you to report to us any problems (content- or grammar-wise) you notice with this booklet.

Many people have contributed to this booklet during the years. Parts of this paper were originally written by Mario Čagalj, Jean-Pierre Hubaux and Imad Aad, under the title "Hands-on exercises: IEEE 802.11b standard"[3]. Several sections where later modified by Simon Thoustrup, Sigurd Hilbert Madsen and Frej Laursen Würtz. Later on, parts of these exercises were re-written/edited by Kasper Rasmussen and then edited by Davide Zanetti and Ghassan Karame.

We hope that you will enjoy the exercises as much as we enjoyed creating them!

Luka Mališa
Aanjhan Ranganathan
Joel Reardon

Chapter 1

Preparing Lab Reports

After each set of lab exercises, groups must prepare and submit a lab report. This report introduces the experiments, documents the findings, and provides answers to all the questions raised in the lab manual. The grade depends on the data, results, and analysis provided in the report and how the questions are answered, along with the quality of report's presentation.

1.1 Quality

Writing suitably scientific reports is an important skill, and as such some of the grade will depend on the writing quality. We expect polished, well-formatted, copy-edited, scientific lab reports that clearly introduce the nature of the experiments, the hypotheses, the results, and the analyses. In the next chapter we provide a simplified example report. The end of this chapter includes a LaTeX template report that may be helpful in formatting your reports.

We expect anyone yet unfamiliar with scientific writing to research this independently. Many resources are available online or at libraries to which ETH students have access. The book "The Elements of Style" by Strunk and White is very quick to read and has many useful tips on professional writing.

1.2 Authorship

While the primary author of each lab report may vary throughout the course, we expect every group member to read and understand the contents of every lab report. Before submitting the report, a draft is to be provided to all group members. They will then take the opportunity to edit, correct, and improve the report. The lead author should appear as the first author and should change so that each group member acts as the primary author of a suitable share of the reports.

For *each* lab report, students should **acknowledge** the resources used to answer the questions raised in the lab manual (books, friends, web sites, discussion with other teams, etc.). Although discussion between teams is not prohibited, each solution should reflect your own views on the problem.

1.3 Submitting

Each lab report must be submitted in printed form right at the beginning of the subsequent lab. This allows approximately two weeks to prepare the report. Missed deadlines will receive a grade of zero. In addition to submitting a printed report, you will also send us a pdf version of the document to our e-mail (`sown@lists.inf.ethz.ch`). The digital version of the report can be submitted any time before the beginning of the next lab.

When sending us the e-mail with your report use the following guidelines:

- Subject: Report <group number>_LabX
- Attachment name: `<group number>_LabX.pdf`

If you are group A1 and you are submitting the report for the first lab, then the submission e-mail would have the subject "Report A1_Lab1" while the attached pdf name would be `A1_Lab1.pdf`.

1.4 Template Report

```
\documentclass[12pt,a4paper]{article}

\usepackage{hyperref}

\title{Template for Security-of-Wireless-Networks Reports}
\author{Author Names}
\begin{document}
\maketitle

\begin{abstract}
\end{abstract}

\section{Introduction}

Example use of bibliography\cite{dd_manpage}

\section{Materials and Methods}

\section{Results}

\section{Analysis}

\bibliographystyle{plain}
\bibliography{bibliography}

\end{document}
```

1.5 Template Bibliography

Example contents of the file `bibliography.bib`.

```
@misc{dd_manpage,
      author = "John Doe",
```

```
      title = "dd(1) - Linux man page",
      month = "Septemper",
      year = "2012",
      howpublished = "\url{http://linux.die.net/man/1/dd}"
}
```

Chapter 2

Example Lab Report

2.1 Foreword

This chapter contains a toy example of a lab report for a single small experiment to give an understanding of the reports we expect to receive.

2.2 Introduction

Unix-based operating systems provide virtual character devices to generate data streams. Such streams are useful in a variety of contexts, such as testing throughput for other devices and filling the content of a dummy file. Among these devices are `/dev/zero` to generate a stream of binary zeros, `/dev/random` to generate a stream of hardware (or true) randomness, and `/dev/urandom` to generate a stream of random data without the same constraints of true randomness offered by the `/dev/random` device.

The simplest device, `/dev/zero`, can be implemented by simply clearing the data buffer for all read operations. More complicated are the random devices, which both rely on an entropy pool of random data collected by the measurement of random hardware sources. These sources vary, but a common approach is to sample the duration between hardware interrupts, as interrupts are unpredictable and human driven. The difference between */dev/random* and */dev/urandom* is that the former will always block until it has suitable random data, while the latter will simply reuse stale randomness when necessary instead of blocking, limiting its application to non-cryptographic settings.

In this work, we quantify the rate at which these devices can generate data. Given their mode of operation, we hypothesize that the rapidest will be `/dev/zero`, as no data needs to be read from another source to write the result. The next will be `/dev/urandom`, which needs to read from the entropy pool and possibly perform additional operations on old randomness to generate (cryptographically-unsuitable) pseudorandom data. Finally, `/dev/random` will be vastly slower than both these devices once the entropy pool is exhausted, because it will block until new entropy is added.

source	throughput while idle	throughput while typing
`/dev/zero`	390.1 ± 2.1 MB/s	389.3 ± 5.6 MB/s
`/dev/urandom`	10.0 ± 0.2 MB/s	10.0 ± 0.2 MB/s
`/dev/random`	15.2 ± 8.5 B/s	52.3 ± 18.3 B/s

Table 2.1: Read throughputs for character devices.

2.3 Materials and Methods

We perform our measurements on a commodity Thinkpad T410 laptop computer running Linux version 3.4.0-rc5. The throughput is measured using `dd`[1], reading from the appropriate input device and writing to `/dev/null`; this was done 16 bytes at a time. Each device was read for ten seconds and the average throughput is then reported. We performed the experiment 5 times and report the 95% confidence intervals under the assumption that the resulting throughputs are normally distributed. We execute two variants for each experiment: one where the computer is unused, and another where a human operator types a fixed paragraph at a normal typing speed so as to generate fresh entropy. Between experiments, the computer was used for a minute to generate fresh entropy for the entropy pool.

2.4 Results

The results from our experiment are presented in Table 1. The throughput measure for `/dev/random` suffers from the fact that it is blocking: longer durations of our experiment had no effect once the entropy pool is depleted.

2.5 Conclusion

Our experiment confirmed our hypothesis. The `/dev/zero` had the largest throughput, orders of magnitude larger than `/dev/urandom`. Effectively no throughput is offered by `/dev/random`: once the entropy pool is exhausted, then no data is provided until new entropy is added.

Typing while performing the experiment seems to feed the entropy pool. More data was provided by `/dev/random` when a user typed on the keyboard than when the computer was idle. Keyboard input offered no such benefits for the other, non-blocking devices.

Bibliography

[1] dd manpage
http://linux.die.net/man/1/dd

Chapter 3

Introduction

3.1 IEEE 802.11b standard

The scope of the IEEE 802.11 [2] standard is to provide specifications for wireless connectivity for fixed, portable and moving stations within a local area. It defines over-the-air protocols necessary to support networking in a local area. This standard provides MAC and physical layer functionality. The extension IEEE 802.11b (used in these exercises) gives accommodation of transmission rates of up to 11 Mbps and operates in the 2.4 GHz band. The IEEE 802.11 standard takes into account power management, bandwidth, security and addressing, since these are the significant differences from wireless to wired LANs. The MAC layer specification of the IEEE 802.11 standard provides radio channel access control functions, such as addressing, access coordination etc. The Distributed Coordination Function (DCF) is the primary access protocol for the automatic sharing of the wireless medium between stations and access points. This DCF uses a carrier sense multiple access/collision avoidance (CSMA/CA) protocol for sharing the wireless medium. As shown in Figure 3.1, the DCF delays frame transmissions right after the channel is sensed idle for DIFS (DCF InterFrame Spacing) time. It waits for an additional random time, backoff time, after which the frame is transmitted. The backoff time is bounded by the contention window size CW. This is applied to data frames in the basic scheme, and to RTS frames in the RTS/CTS scheme. The backoff time of each station is decreased as long as the channel is idle. When the channel is busy, backoff time is freezed. When backoff time reaches zero, the station transmits its frame. If the frame collides with another frame (or RTS), the sender times out waiting for the ACK (or the CTS) and computes a new random backoff time with a larger CW to retransmit the frame with lower collision probability. When a frame is successfully transmitted, the CW is reset to CW min. The network allocation vector (NAV) of all other stations is set to the frame duration field value in RTS/CTS and DATA headers.

Because of the possibility of partial network connectivity, wireless LAN protocols must take into account the hidden terminal problem (this occurs when a station is able to receive frames from two different stations but these two stations can not hear each other). To solve this, a virtual carrier sense mechanism through the exchange of control frames is used (cf. Figure 2). These are the Request to Send (RTS) and the Clear to Send (CTS) frames. The RTS and CTS frames contain a duration field that defines the period of time that the medium is to be reserved to transmit the actual data frame and the returning ACK frame. All stations within the reception range of either the originating station (which transmits the RTS) or the destination station (which transmits the CTS) shall learn of the medium reservation. Thus a station can be unable to receive from the originating station, yet still know about the impending use of the medium to transmit a data frame. The RTS/CTS control frames should not be used for short data frames, since they would add traffic. According to IEEE 802.11b standard, the use of RTS/CTS mechanism is

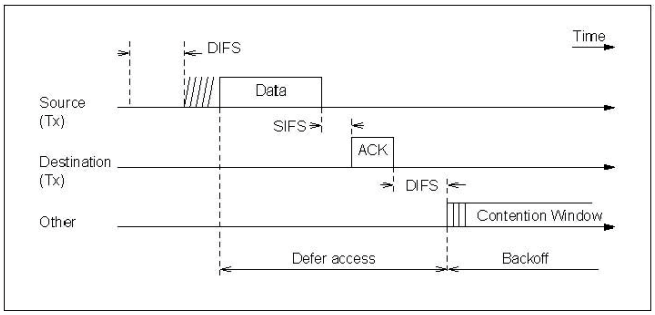

Figure 3.1: The distributed coordination function (DCF) of IEEE 802.11 operating in the basic mode.

optional.

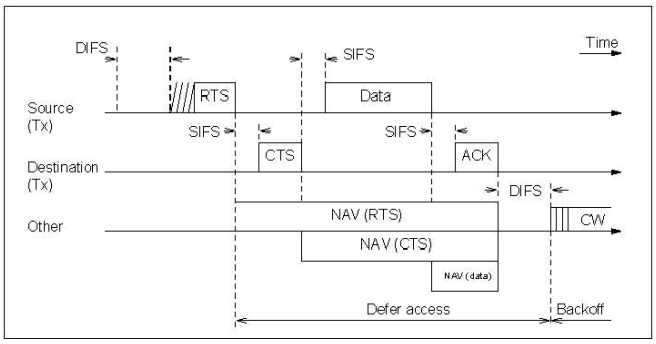

Figure 3.2: The distributed coordination function (DCF) of IEEE 802.11 operating in RTS/CTS mode.

IEEE 802.11 can be used in two different operating modes:

i Infrastructure mode (all communication goes through an access point (AP)).

ii Ad-Hoc mode (no AP is present; all communication is peer-to-peer).

3.2 The Wireless Medium

The wireless medium is a shared medium, where all traffic is available for everyone in the vicinity of the network. It is similar to a wired hub connected network, where every packet in the network is sent to all connected stations. But in a wireless network the user does not even need to be connected in order to eavesdrop on the traffic.

The wireless standard is described in IEEE 802.11[1], but this exercise will focus on the a/b/g subsets.

The 802.11b (11Mb/s) and 802.11g (54Mb/s) standards both operate in the 2.4 GHz band, which may cause intererence with other devices that share the frequency: microwave ovens, cordless telephones, bluetooth devices, etc. The 802.11a (54Mb/s) standard operates in the 5GHz band, thus does not suffer from interference from these products.

The wireless medium is as mentioned a shared medium, and in order to limit the interference from other wireless devices, the standards operates within fourteen different channels. One should

[1]Further information can be found at [1].

attempt to use channels which are currently not in use in the area. Nearby channels do still create interference, but as a rule of thumb, channel 1, 6, 11 should be non overlapping. However, this is not entirely true, and it depends on the transmission power of each station. But for the exercises you should at least attempt, if possible, to select channels which are two apart, i.e., 1, 3, 5, etc.

The shared property of the medium has the further unfortunate property that unauthorized persons in some scenarios can access, eavesdrop or manipulate with the traffic, as will be demonstrated in the following exercises. It is trivial for a spacially-proximate person to become a passive eavesdropper, and active man-in-the-middle-style attacks can be mounted without requiring physical access to network cables.

Chapter 4

Laboratory 1: Selfish Behaviour in Wireless Networks

This chapter will take you through the steps of configuring the wireless adaptors, testing throughput, and describing methods for selfish behaviour. The lab setup is illustrated in Figure 4.1.

Contents

4.1 Configuration of wireless client adaptors

This section makes extensive use of the performance testing tool `iperf`. If you are unfamiliar with it, begin by consulting the manual pages for `iperf`:

```
$ man iperf
```

Your goal in this exercise is to setup an operational **Wireless LAN** (WLAN) and sniff traffic. We use Proxim Orinoco 11a/b/g PCI adaptors as our network card.

TASK 1: **Basic settings**

1. Start a root terminal with password **sown**.

   ```
   $ su
   ```

2. Assign an IP address to the wireless adaptor by typing the following:

   ```
   # ifconfig wlan0 10.0.0.xy netmask 255.255.255.0 up
   ```

   where 10.0.0.xy is the IP address assigned to the machine. Use the following addressing scheme to assign IP addresses to machines. All the machines use the prefix 10.0.0 in their IP address. Then we set X and Y as follows:

   x - to the table number.

   y - to 1 in the case of the Router; to 2 in the case of the Station

   For example, if you sit at table number 3 your router should be 10.0.0.31 and your station 10.0.0.32.

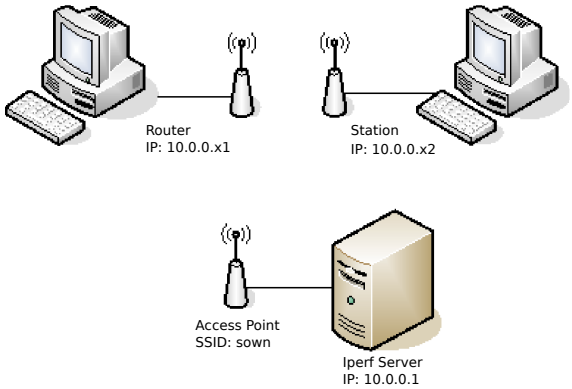

Figure 4.1: Lab overview

3. Verify that the IP was set correctly.

```
# ifconfig wlan0
```

TASK 2: **Infrastructure mode**

This mode is used to have the wireless interface associate with an **Access Point**. As an Access Point is usually connected to a wired network, this provides for access to the LAN. Note that in **Infrastructure Mode**, an adaptor scans frequency channels to find an Access Point; we do not have to set the channel manually.

1. The lab instructor will start an access point and an iperf server.
2. Start a root terminal.
3. Configure the interface with the following properties using the **iwconfig** tool:

   SSID - (*essid*) - Service Set ID (SSID) is a unique identifier that client devices use to associate with either the AP or the other client. This value **MUST** match the SSID of an access point (AP) with which we want to communicate. In our case, the SSID is *sown*. Do not forget that SSIDs are **case sensitive**.

   Network Type - (*mode*) - we want Infrastructure (*managed*).

   802.11-version - (*rate*) - we use the 802.11b (*11M*),

   ```
   # iwconfig wlan0 essid sown mode managed rate 11M
   ```

4. Try to ping other machines in the room to make sure that it worked.
5. There is an iperf server at 10.0.0.1. Try to ping the server.
6. Start the **wireshark** tool to monitor packets exchanged between your station and the server having IP address 10.0.0.1.

   ```
   # wireshark &
   ```

   To start a capture, click *Capture* - *Options* and make sure the *wlan0* interface is selected. Ensure that *Capture packets in promiscuous mode* is *not* enabled. You may want to **update the list of packets in real time** and use **automatic scrolling**.

   To generate traffic that wireshark can capture, ping the server.

7. From the output produced by the wireshark tool, retrieve the MAC address corresponding to the station with IP address 10.0.0.1. Verify that the other communicating party's MAC address is indeed yours (you can find your MAC address with `ifconfig wlan0`). MAC

addresses are used to uniquely identify hardware on the data link layer (in the OSI model) of the network.

8. Wireshark has shown you your messages, however wireless messages can be seen by anyone. To do this, we must put the network card in monitor mode and surveil all the traffic in promiscuous mode. **Restart wireshark completely** and then switch modes:

```
# ifconfig wlan0 down

# iwconfig wlan0 mode monitor channel 3

# ifconfig wlan0 up
```

9. Now run wireshark and start a new packet capture session. This time try with **Capture packets in promiscuous mode** enabled. Observe differences between the promiscuous mode and non-promiscuous mode.

10. **Close wireshark** and return your network card back to managed mode.

As an appendix to your lab report, ensure that the following questions are answered. (Question-answer format is acceptable; it does not need to be organized as a report.) The manual pages for `iwconfig` may be useful: `man iwconfig`. Read through it to be able to answer these questions before proceeding to the next exercise.

- What are all the different modes that iwconfig offers?
- What does each of these modes do?
- Why did we set the channel in monitor mode?
- What is the difference when capturing in promiscuous mode and non-promiscuous mode?

4.2 How much do we get out of 11 Mbps?

IEEE 802.11b adaptors operate at the maximum data rate of 11 Mbps. Supported Data Rates for the Proxim adaptor including the following: **1Mbps, 2 Mbps, 5.5 Mbps, and 11 Mbps**. The data rate of 11 Mbps sounds fairly good (∼1.3 MBps). Note, however, that this is the 802.11b raw data rate in perfect conditions at the physical level of the network. (Recall the OSI network stack: physical/link/network/transport/application. Each of these adds its own overhead and traffic.) In this exercise, you will perform an experiment to measure the observed data rate at the network layer. You will connect to the iperf server in the lab and observe the throughput. (The tool `iperf` is used to check network performance; if you are unfamiliar with it or its parameters, start by reading the corresponding man pages.) Measurements will be performed in a real-life scenario (i.e., several stations competing for the available bandwidth).

In the following tasks you will use the program **iptraf**, which monitors IP network statistics (e.g., data rates). To start this tool, open a new terminal and type **iptraf** at the command line. Then select **"Detailed interface statistics"** and choose the wlan0 interface.

TASK 1: **Wireless iperf server**

In this exercise, an iperf server is running on the AP machine (10.0.0.1). Do the following on *both* machines of your table:

1. Configure the machine to work in infrastructure mode. Using **iwconfig** and **ifconfig**, associate with the SSID **sown**. (Make sure you are in managed mode.)

2. Set up an iperf session between your machines and the server (10.0.0.1), by typing:

```
# iperf -c 10.0.0.1 -i 2 -t 1000 -u -b 100000000
```

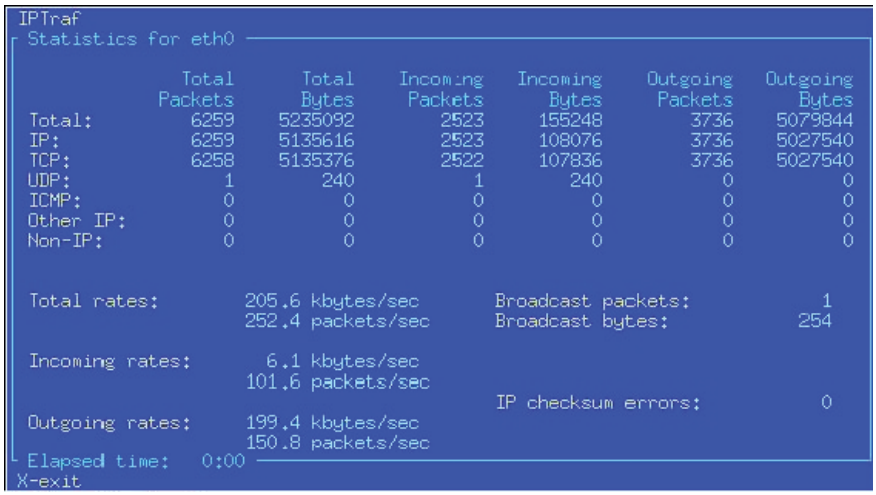

Figure 4.2: Iptraf screen

3. Observe the data rate with **iptraf**. Collect results on the observed throughput.
4. Wait until the measurements stabilize. Observe the distribution of bandwidth over the machines.

TASK 2: **Different Channels** In the previous exercise, all machines competed for bandwidth on a single channel. Now, we divide the class into 3 groups of 4 machines (i.e., two teams pair together). The teams will work together to complete the exercise. Each group will use a unique channel for communication in this exercise. One member of each team (e.g., the station of the lower group number) acts as the wireless base station, which we call the **AP machine**. The others will be clients in managed mode. We use the tool `hostapd` to act as a wireless access point. There is already a configuration file for this exercise called `hostapd.basic.conf`, which can be found in the `lab_material` subdirectory in the user's home directory: e.g., `/home/router7/lab_material/hostapd.basic.conf` – if there were a machine called **router7**. Lab material will always be found in this directory. It should provide the correct communication channel for this exercise, but confirm it by reading the contents of the file first.

1. In this exercise, only three computers will compete for bandwidth. The remaining machines in the lab are separated into different channels. We will measure the resulting throughput. **Formulate a hypothesis** on the resulting throughput as compared to the previous exercise.
2. The computer that acts as the host should first stop all programs using the network (iptraf, wireshark, etc.) and start the access point:

   ```
   # hostapd -d hostapd.basic.conf
   ```

3. Other computers should then select the channel and essid provided by the hostapd configuration on the machine running the access point:

   ```
   # iwconfig wlan0 essid <essid> channel <channel>
   ```

4. All computers should then set their IP address to communicate:

   ```
   # ifconfig wlan0 10.0.0.XY netmask 255.255.255.0 up
   ```

5. The AP machine should run an iperf server

   ```
   # iperf -s -u
   ```

6. The other machines now connect to the running iperf server using the *group's* AP machine's IP address:

   ```
   # iperf -c <AP's IP> -i 2 -t 1000 -u -b 10000000 -d
   ```

   The extra command `-d` tests both upload and download throughputs. Be sure to use it for all the experiments in your group.

7. Wait until the throughput stabilizes and observe the result. Compare it with your hypothesis. Observe how it compares with other clients on your channel. Compare bandwidth with other groups. Are the data transfer rates similar?

8. Try moving around the wireless antennas closer and further from the group's AP and observe the results. Observe the fairness properties of the medium.

9. Stop and start the iperf server.

The next two tasks will look at how to interfere with the throughput of neighbouring machines.

TASK 3: **Rate Changing** In this task, some machines will change their data transfer rate and observe the effect this has on all the machines in their group. While the transfers from the previous task continue, one member of each group will change their rate from 11M to 1M: decreasing their data transfer rate to about 10% of its previous value. As always as a group, *formulate a hypothesis* before beginning on the result that decreasing the data rate will have on the throughput both the lowered-rate machine and the non-lowered-rate machines.

1. Ensure that your group has an iperf server and clients running with a stable, shared bandwidth.

2. Select one non-AP member of your group to decrease their data rate. They should then run the following command:

   ```
   # iwconfig wlan0 rate 1M
   ```

3. Wait until the throughput stabilizes. Observe the result and contrast it to your hypothesis.

4. Now, on the remaining clients, decrease the data rate to 1M. Wait until the throughput stabilizes and observe the results.

5. Return the rate-reduced machines to 11M before continuing with the next exercise.

   ```
   # iwconfig wlan0 rate 11M
   ```

6. Stop and start the iperf server.

TASK 4: **Selfish Backoff** The wireless medium, being shared and relatively uncoordinated, has techniques to detect and avoid network congestion, thus improving bandwidth utilization and fairness for all participants. By simply not adhering to the techniques, one can unilaterally act against the best social good for private gain.

In this task, we will experiment with using different backoff strategy; backoff refers to the time one waits without sending data if one observes congestion on the wireless medium. We've prepared a modified wireless driver that sets the backoff timer to zero, effectively performing no backoff.

As always, read through all the steps of the exercise and *as a group formulate a hypothesis* on the results before beginning. We will test the results for one machine with the modified driver and all machines with the modified driver. **Write down your hypotheses** on the resulting throughput for all machines for all experiments. We assume that the setup continues from the previous task: one AP machine running both hostapd and an iperf server along with 3 clients associated with the AP running iperf clients.

1. Stop all iperf clients and restart the iperf server running on the AP.
2. Select one non-AP member of your group to change their wireless driver. In addition to stopping the iperf client, they should stop any other usage of the wireless card, e.g., iptraf or wireshark) and take down their wireless network:

   ```
   # ifconfig wlan0 down
   ```

3. To swap drivers, go to the `lab_material/modified_driver` subdirectory of the user's home directory and run the following script:

   ```
   # sh driver-modified.sh
   ```

4. If swapping the driver produced error-like output, inform a lab assistant!
5. Swapping drivers makes resetting the parameters for both `iwconfig` and `ifconfig` necessary. Do this as before to connect to the group's AP, set an IP, and bring back up the wireless card. (The non-modified machines do not need to do this.)
6. Connect to the group's iperf server.
7. Wait until the throughput stabilizes. Observe the result for the modified machine and the other machines. Contrast it to your hypothesis. Check if it is still fair.
8. Move the antennas closer and further from your group's AP. Observe if this has any effect on the modified AP or the non-modified AP.
9. Now, its not just one person who can cheat. Make the other clients use the modified driver in the same way as above. Make sure that the AP machine always uses the normal driver.
10. Observe the results when all modified machines are using the modified driver.
11. Move the various cheating clients closer and further from the AP. Observe the results. Compare with moving the antennas when the wireless cards were not cheating.
12. If you want to return to the original driver to compare moving around the antenna, you can run the `driver-vanilla.sh` script.

In your lab report, ensure that the following questions are answered, appropriately divided into introduction-like, theory-like, experiment-like, and analysis-like sections.

- What were your hypotheses for each task?
- What was the reasoning behind these hypotheses?
- How does the data rate compare to the actual throughput?
- How is the available bandwidth shared when all the computers connected to the same AP?
- How does the throughput change when computers are spread to different channels?
- What happens to the throughputs when one person reduces their data rate?
- What is backoff and what does it do?
- What happens when one machine ignores backoff?
- What happens when all machines ignore backoff?
- Why does backoff effect throughput as it did?
- How do the results compare to your hypothesis?
- If the results differed from expectation, explain and analyze why this was the case. Otherwise, the explanation of the reasoning is sufficient and claim instead that the experiment gave evidence towards your hypothesis.

Bibliography

[1] Ieee 802.11. http://en.wikipedia.org/wiki/802.11.

[2] Lan/man standards committee. ansi/ieee std 802.11: Wireless lan medium access control (mac) and physical layer (phy) specifications. ieee computer society, 1999. http://standards.ieee.org/getieee802/802.11.html.

[3] Jean-Pierre Hubaux Mario Čagalj and Imad Aad. Hands-on exercises: Ieee 802.11b standard. In *Laboratory for computer Communications and Applications (EPFL-I&C-LCA)*, November 9, 2004.