WirelessLab WS 2016/17

Homework 7: Rate Control Algorithms

Group 6

Gasper Kojek, Jens Klein

Question 1: RCA in mac802.11

Network & Terminology:

- N6 = Node 6 (172.17.5.10 on wlan0, 172.17.3.106 on br-lan)
- N15 = Node 15 (172.17.5.11 on wlan0)
- ST = Stepping Stone (172.17.3.1 on eth1)

a)

Setup:

- checking connectivity:
  - **N6 <-> N15**:

    N6: `ping -I wlan0 172.17.5.11 | head -n 2`

    Output:

    ```
    PING 172.17.5.11 (172.17.5.11): 56 data bytes
    64 bytes from 172.17.5.11: seq=0 ttl=64 time=0.615 ms
    ```

    Connected

  - **N6 <-> ST**:

    N6: `ping -I br-lan 172.17.3.1 | head -n 2`

    Output:

    ```
    PING 172.17.3.1 (172.17.3.1): 56 data bytes
    64 bytes from 172.17.3.1: seq=0 ttl=64 time=0.645 ms
    ```

    Connected

- checking channel on wireless, either (1,6,11)
  - N6: `iw wlan0 info | grep channel`

    Output:
    `channel 11 (2462 MHz), width: 20 MHz (no HT), center1: 2462 MHz`

  - N6: `iw wlan1 info | grep channel`

    Output:
    `channel 11 (2462 MHz), width: 20 MHz (no HT), center1: 2462 MHz`

- N15: `iw wlan0 info | grep channel`

  Output:
  `channel 11 (2462 MHz), width: 20 MHz (no HT), center1: 2462 MHz`

  All cards are on the same channel (11)

- set transmission power to 1.0 dBm on sender
  - N15: `iw dev wlan0 set txpower fixed 100; iw wlan0 info | grep txpower`

    Output (1st line):

    `txpower 1.00 dB`

- set antenna A fixed for both sender and receiver
  - N6: `echo "fixed-a" > /sys/kernel/debug/ieee80211/phy0/ath5k/antenna`
  - N15: `echo "fixed-a" > /sys/kernel/debug/ieee80211/phy0/ath5k/antenna`
- check if ministrel is enabled
  - N15: `dmesg | grep "ieee80211 phy0"`

    Output:

    ```
    [   14.225247] ieee80211 phy0: Selected rate control algorithm 'minstrel_ht'
    ```

    Ministel is enabled

- start nc on ST to receive the trace
  - ST: `nc -l -p 8080 > "trace-ch11-1dbm-ant1-$(date +%s).cap"`

    File naming structure:

    `trace-<ch + channel>-<tx_power + dbm>-<ant + antenna used>-<unix timestamp>.cap`

- start iperf server on receiver
  - N6: `iperf -s -u`
- start tcpdump on receiver without capturing the udp body, forward to ST
  - N6: `tcpdump -i wlan1 -s 104 -w- udp | nc 172.17.3.1 8080`
    - `-s 104` snaplen of 104 bytes on the frames. enought to have udp header
    - `-i wlan1` interface of the monitor
    - `-w-` outputs to STDOUT
- start iperf client on sender
  - N15:

```
for i in `seq 12`; do
    iperf -c 172.17.5.10 -u -b 55M -t 30 -l 1024
    sleep 2s
done
```

- Repeat with different txpower and antenna settings:
    - txpower = 10.00dBm, Ant 1
        - n15: `iw dev wlan0 set txpower fixed 1000; iw wlan0 info | grep txpower`
        - Output (1st line):

          ```
          txpower 10.00 dBm
          ```

        - ST: `nc -l -p 8080 > "trace-ch11-10dbm-ant1-$(date +%s).cap"`
    - txpower = 20.00dBm, Ant 1
        - n15: `iw dev wlan0 set txpower fixed 2000; iw wlan0 info | grep txpower`
        - Output (1st line):

          ```
          txpower 20.00 dBm
          ```

        - ST: `nc -l -p 8080 > "trace-ch11-20dbm-ant1-$(date +%s).cap"`

```
* txpower = 1.00dBm, Ant 2

    * n15: `iw dev wlan0 set txpower fixed 100; iw wlan0 info | grep txpower`

    * Output (1st line):

        ```
        txpower 1.00 dBm
        ```

    * n6: `echo "fixed-b" > /sys/kernel/debug/ieee80211/phy0/ath5k/antenna`

     * n15: `echo "fixed-b" > /sys/kernel/debug/ieee80211/phy0/ath5k/antenna`

    * ST: `nc -l -p 8080 > "trace-ch11-1dbm-ant2-$(date +%s).cap"`

* txpower = 10.00dBm, Ant 2

    * n15: `iw dev wlan0 set txpower fixed 1000; iw wlan0 info | grep txpower`

    * Output (1st line):

        ```
```
```

```
        txpower 10.00 dBm
        ```

    * ST: `nc -l -p 8080 > "trace-ch11-10dbm-ant2-$(date +%s).cap"`

* txpower = 20.00dBm, Ant 2

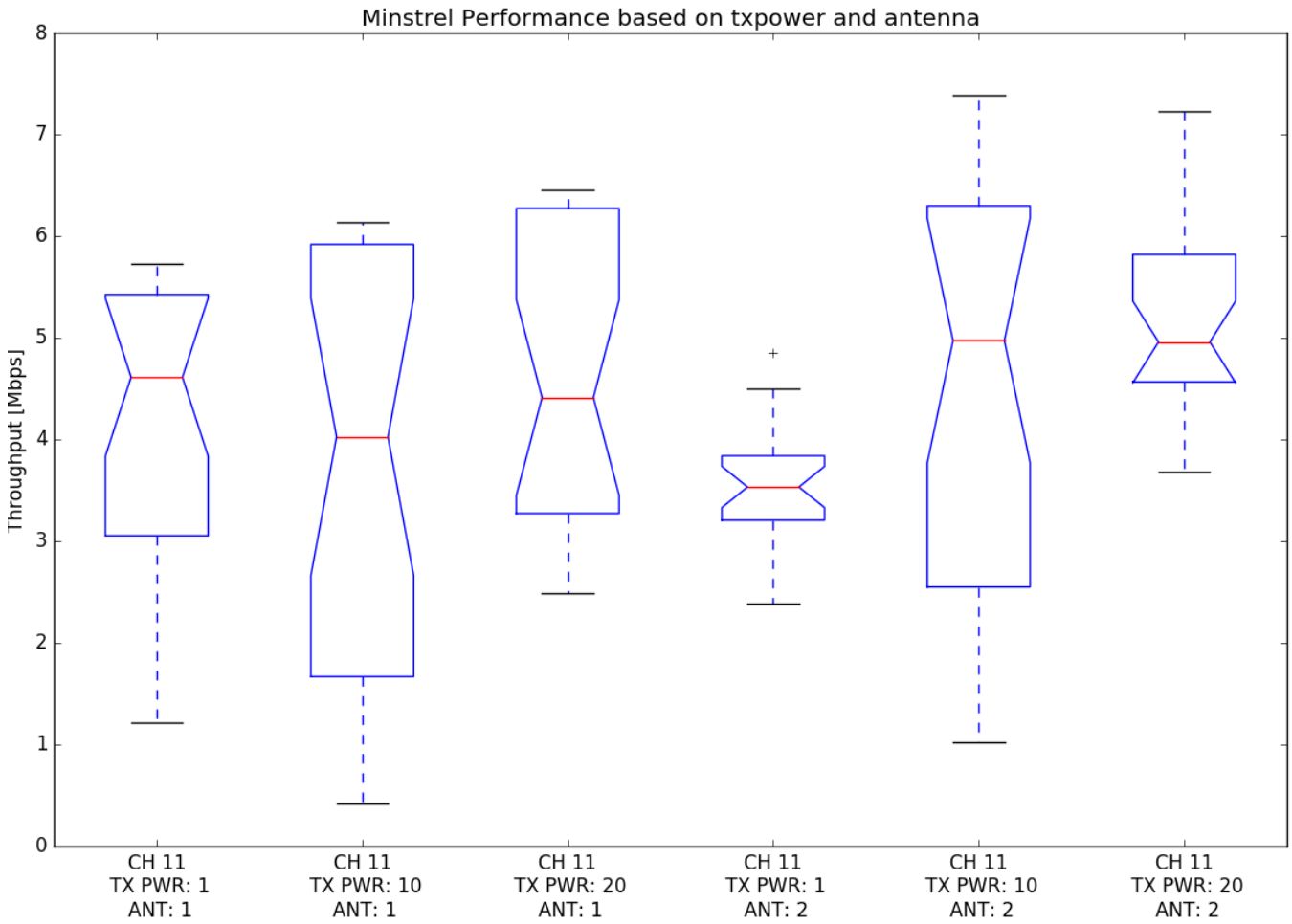
    * n15: `iw dev wlan0 set txpower fixed 2000; iw wlan0 info | grep txpower`

    * Output (1st line):

        ```
        txpower 20.00 dBm
        ```

    * ST: `nc -l -p 8080 > "trace-ch11-20dbm-ant2-$(date +%s).cap"`
```

- Repeat all steps at a different time of the day

# b

Since we decoded the timestamp and other meta information in the file name we can you our pyhton script `capture_times.py` to give us information about all caputres in a table style.

```
Ordered Overview of capture-time and settings
=============================================================
| 16:41:07 12.12.2016  | ANT:   ant2 | TXPWR:  1dbm | CH:  ch11 |
| 17:02:44 12.12.2016  | ANT:   ant2 | TXPWR: 10dbm | CH:  ch11 |
| 17:12:56 12.12.2016  | ANT:   ant2 | TXPWR: 20dbm | CH:  ch11 |
| 17:28:09 12.12.2016  | ANT:   ant1 | TXPWR:  1dbm | CH:  ch11 |
| 17:47:10 12.12.2016  | ANT:   ant1 | TXPWR: 10dbm | CH:  ch11 |
| 17:59:30 12.12.2016  | ANT:   ant1 | TXPWR: 20dbm | CH:  ch11 |
| 13:28:57 14.12.2016  | ANT:   ant1 | TXPWR: 10dbm | CH:  ch11 |
| 13:39:26 14.12.2016  | ANT:   ant1 | TXPWR:  1dbm | CH:  ch11 |
| 13:53:01 14.12.2016  | ANT:   ant1 | TXPWR: 20dbm | CH:  ch11 |
| 14:06:16 14.12.2016  | ANT:   ant2 | TXPWR:  1dbm | CH:  ch11 |
| 14:16:56 14.12.2016  | ANT:   ant2 | TXPWR: 10dbm | CH:  ch11 |
| 14:45:22 14.12.2016  | ANT:   ant2 | TXPWR: 20dbm | CH:  ch11 |
-------------------------------------------------------------

12 captures in total
```

Boxplot: Throughput vs txpower and antenna using minstrel
```

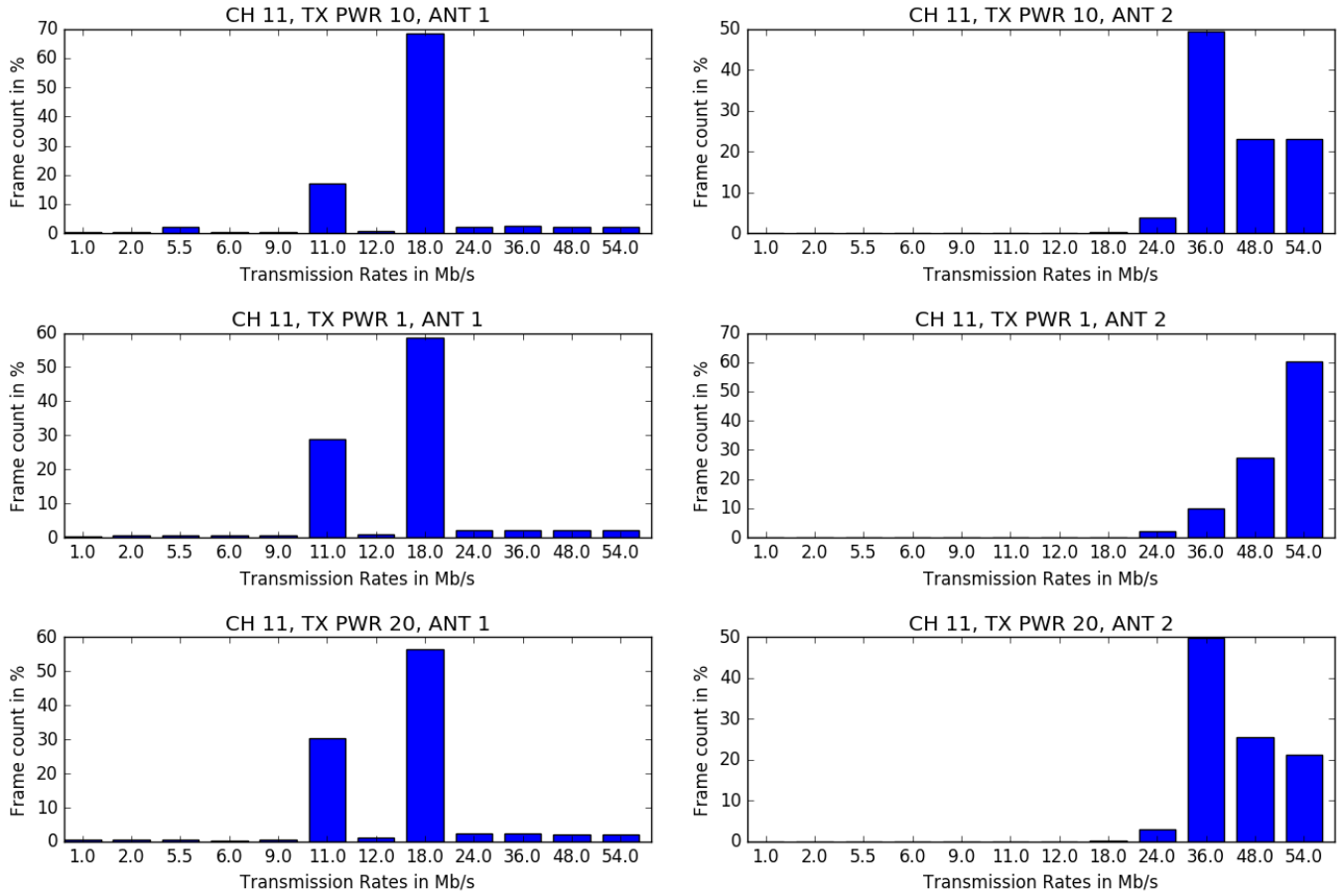


In this experiment minstrel is enabled as a rate control mechanism.
On the y-axis there is the throughput given by iperf server log in Mbit/s
versus the six boxplots.
In total there are 12 runs with basically three factors: which of antenna a and b, transmission power of 1dBm, 10 dBm and 20 dBm and lastly the time factor.
We know from previous assignments and the forum that port 1 has no antenna attached to it and port 2 is a pigtail antenna. For antenna port 1 the medians are between 4 and 5 Mbit/s for all transmission power settings. But the throughput is also spread between 1 and 6 Mbit/s. Taking a glimpse on the raw data this is probably due to the different times the frame were captured. Generally speaking, the power transmission setting does not have much influence on the throughput of antenna port 1. The reason for this might be the short distance between the nodes.

For the pigtail antenna we can see a slightly different behaviour. The distance is still the same but the mean for 1 dBm transmission power shows about 1.5 Mbit/s worse performance. While at 10 dBm the quartiles are of same size as antenna port 1, 1 dBm and 20 dBm show a narrower quartiles.

In conclusion we can not see that the transmission power or the antenna has significant impact on the performance which was not really expected. Assumingly a higher transmission power would lead to better performance. But this is not observable.

Barplot: Distribution of transmission rate selection



The figure above shows six bar plots for the combination of factor transmission power and used antenna port 1 and 2. Column 1 show the relative count of transmission rates on the y-axis for antenna port 1 and column 2 respectively for antenna port 2.
The rates were gathered from the 802.11 frame header picked by the minstrel mechanism of the client. For antenna port 1 we observe two significant peaks at 11.0 Mb/s and 18 Mb/s. with a hasty glance at the raw data we can see that at capture time t1 we have a steady transmission rate of 18.0 Mb/s whereas at time t2 its only 11.0 Mb/s. This timely seperation allows us to conclude that minstrel is selecting a constant rate for all transmission power levels.

For the antenna port 2 we can see that minstrel is selecting way higher rates between 36.0 and 54.0 Mb/s throughout all transmission powers and times. Only at 1 dBm we see that it tends to select more higher rates then 10 dBm and 20 dbm.

If we compare this barplot with the box plot we can conclude that even higher picked transmission rates didn’t necessarily lead to a higher throughput. Which gives a hint that the selection wasn’t optimal at all experiments.

Question 2: Analysis of Rate Control Algorithms

a) How Minstrel selects its rate when there is something to send.

Minstrel

- Minstrel is the standard rate control algorithm in Linux `mac80211`
- ACK based mechanism
- Throughput and probability of success for each rate measured/calculated every 100 ms
- The core of the Minstrel rate algorithm is the EWMA, or Exponential Weighted Moving Average

Exponential Weighted Moving Average:

EWMA controls the balance of influence of both the old and new packet delivery statistics, as shown here:

$$P_{new} = (1 - \alpha) * P_{this} + \alpha * P_{previous}$$

Where:

- P_{new} is the weighted probability of success for this interval, which will be used by the rate selection process.
- *alpha* is a smoothing factor (or the scaling value) in the EWMA mechanism.
- P_{this} interval is the probability of success of this interval before the rate selection, and it is calculated as the ratio of the number of packets sent successfully to the number of packets sent.
- $P_{Previous}$ is the weighted probability of success for the last interval used to select last transmission rate.

Minstrel retry chain

Start

Rate sensing

Every 100ms a timer starts the statistic calculations for the last 100ms

Calculate probability of success for all transmission rates for the last 100ms using EWMA

Calculate throughput for all transmission rates for the last 100ms

Throughput =

Probability of success * Mega bits transmitted

time for 1 try of 1 packet to be sent on the air

Get the Best throughput rate

Get the Second best throughput rate

Get the rate with highest probability of success

Use these rates for the retry chain

Send a packet

Minstrel retry chain

Type of packet (minstrel)

90%

Normal transmission packet

10%

Sample packet – lookaround

Random rate

< Best

Random < Best

> Best

Random > Best

Success

Best throughput

Failure

< max rate 1 retries (4)

No. of retries

> max rate 1 retries (4)

Next best throughput

< max rate 2 retries (2)

> max rate 2 retries (2)

Best probability of success

< max rate 3 retries (2)

> max rate 3 retries (2)

Lowest baserate

< max rate 4 retries (1)

> max rate 4 retries (1)

Failure of transmission

b) Comparison with the *Rate Control Algorithm Evaluation Paper*

In the paper they comapare different rate control algorithms, focusing on Minstrel, PID and at the end PIDE. We did our tests using only Minstrel, but with different received signal strength configurations, using different transmission powers and different antennas.

Their test setup was much more extensive. They tested it using three vastly different configurations:

- **Controllable platform**, where they connected the transcievers using a co-axial cable, with an attenuator in between.
- **Semi-controlled over-the-air** experiments, where they used actual antennas, but they did them on an empty channel during the night, where there was not interference.
- **In-the-wild over-the-air** experiments, where they used actual antennas, but on a shared channel during office hours, when there were other people using the channel, walking around, using microwaves. These were the real-life experiments, the same that we did.

Furthermore, all experiments they did were performed using IEEE 802.11a. This has two benefits, the first being that the 5 GHz frequency band is currently much less used than 2.4 GHz, and secondly it means that all transmission rates use the same family of modulation and coding methods. On the other hand, we used 802.11g, which uses 2.4GHz, which is used by much more devices, with addition of other interference, not tied to 802.11 standards.

They tested five different scenarios in each group of experiments: static channel, dynamic channel, fading channel, progressive increase/decrease of channel quality and a sudden channel quality change. We tested just one scenario, the static channel scenario, where we did 6 different experiments. We used three different transmission powers (1dBm, 10dBm, 20dBm) and two different antennas (one is attached, one is without the actual antenna). This all amounts to different received signal strength, but a static channel for each run (as far as we control it).

Realistically, we can only compare our results with *In-the wild over-the-air* experiment results from the paper, but even here there isn't much to compare, given that we only did tests using Minstrel, without testing PID rate control algorithm and in a totaly different environment. We can see that in their tests even in the real-world experiments Minstrel outperformed PID, but the difference wasn't that big. Sadly we can not confirm nor deny this data, as we didn't test PID ourselves.

We can see that their test locations S1 and S3 produced similar throughput to our tests. Interestingly, when comparing those runs to our runs, we can see that Minstrel in their case chose mostly 6Mbps transmission rate, while in our tests it chose mostly 18Mbps (runs without the antenna) or 36Mbps - 54Mbps (runs with the antenna). Their location S2, for which Minstrel chose mostly 18Mbps got a throughput around 13 Mbit/s, while we only got around 4.5 Mbit/s. Their location S4, at which Minstrel chose mostly transmission rate of 36Mbps, got an even better throughput of around 23 Mbit/s, while we still got only marginally better results at around 5 Mbit/s.

Comparing these results is interesting, as in their results the differences in throughput when comparing which transmission rate Minstrel chose are significant, whereas in our test the differences were very small. On the other hand, we have no information on what the environment was like when they did their tests for the paper, so we have no way to actually compare the results with any level of confidence, with the only thing we can say with certainity is that the environment must have been quite different in our experiments versus the experiments for the paper.

The only explanation for such low throughpout in our experiments is that our enviromnet is extremely noisy, as the received signal strength we got was very good, above -65dBm, for which in the *Controllable platform* experiments from the paper, the Minstrel should pick the 54Mbps transmission rate and should have gotten a throughput upwards of 30Mbit/s.

One explanation we can see, why Minstrel would select such high transmission rates even in a very noisy environment, is that with high transmission rate, the packet takes much less time to transmit and has better chances of not getting destroyed by another packet than a slow transmission rate, where the packet is “on the air” for a longer time. We can then account the low throughput to a very busy environment, where there is a lot of devices talking at the same time, so even though the SNR we got is very good, the throughput is severely limited by MAC because of multiple devices trying to transmit.