Summary 4

Alexander Pastor

July 16, 2017

Contents

1 Organizational

1.1 Thesis Outline

1 Introduction

- 1.1 Abstract
- 1.2 Motivation and Context
  - 1.2.1 LTE-U/LAA
  - 1.2.2 Defining the scope

2 Fundamentals and Preparation

- 2.1 MAC Protocols
  - 2.1.1 MAC Introduction
  - 2.1.2 CSMA/CA
  - 2.1.3 CSMA/CA with CTS/RTS
  - 2.1.4 (Slotted) ALOHA
  - 2.1.5 ...
- 2.2 GNU Radio
  - 2.2.1 What is GR and GRC
  - 2.2.2 Motivation to use GR
  - 2.2.3 Limitations of GR

3 Measurement Results and Interpretation

- 3.1 Scenario Overview / Justification
- 3.2 Scenario A
- 3.3 Scenario B
- 3.4 Scenario C
- 3.5 ...
- 3.6 Interpretation

4 Conclusions

5 Appendices

- 5.1 How-to: Conduct Measurements Automatically
- 5.2 Alphabetical Acronym Expansions
- 5.3 Alphabetical Index
- 5.4 Bibliography / References
- 5.5 Statement of Authorship

1.2 Device Parameters

Remote PC IP-address: 134.130.223.151
USRPs: 10.0.0.8, 10.0.0.10

1.3 Measurement Plan

For CW 25 the following measurement parameters were commissioned:

- 2 sets of 5 measurements each with a duration of 5 minutes.
- SIFS: 1ms / 3ms
- DIFS: 5ms / 15ms
- Backoff slot: 2s / 6ms
- The results should be plotted with either Matlab or Matplotlib

rest tbd

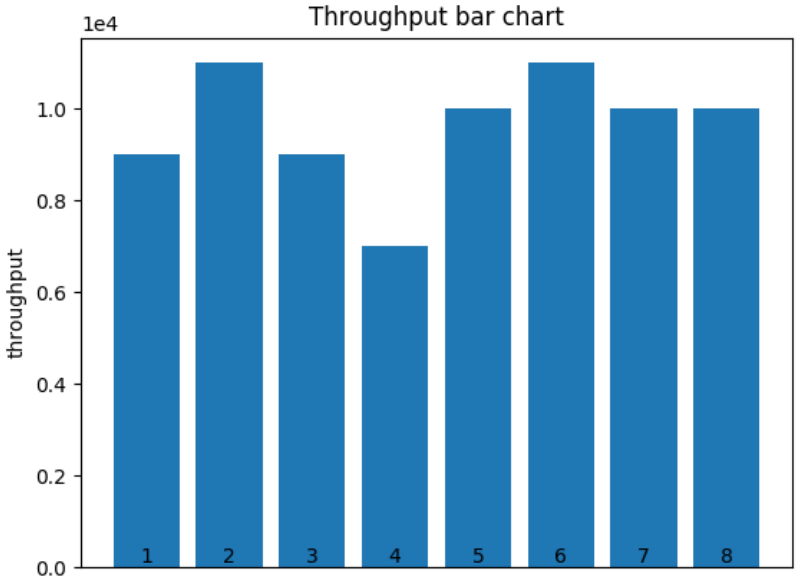
Note: The DIFS time in 802.11 can be calculated as twice the backoff time plus one SIFS time.

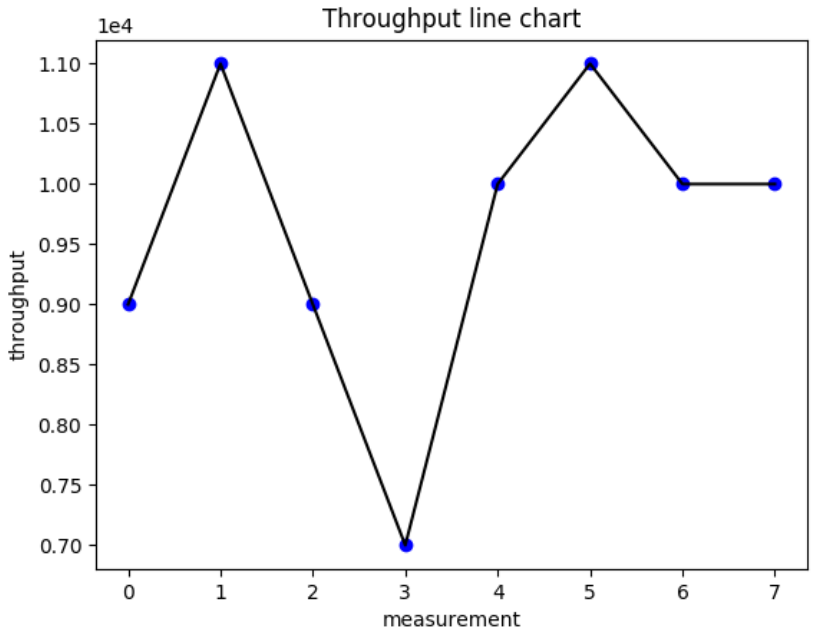
2 USRP Hardware Problems

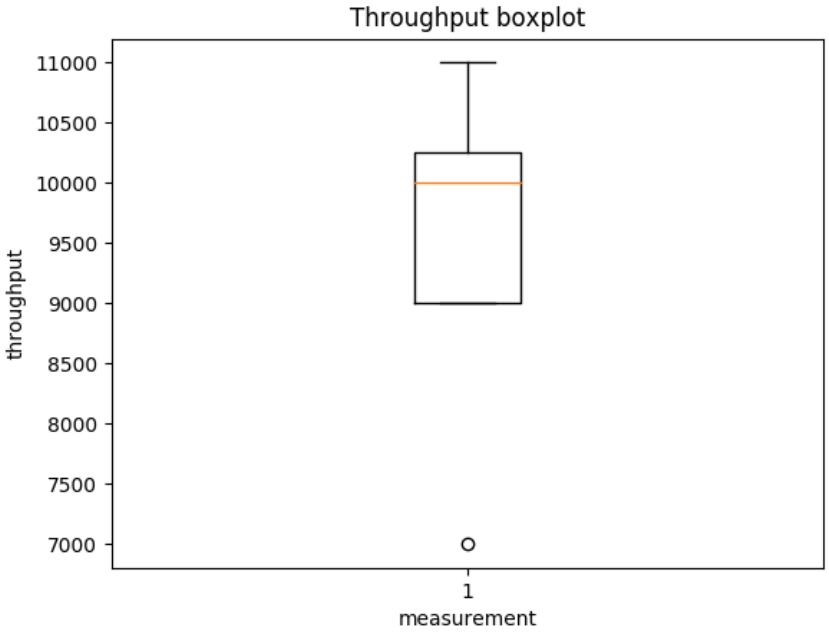
2.1 How It Is Supposed to Work

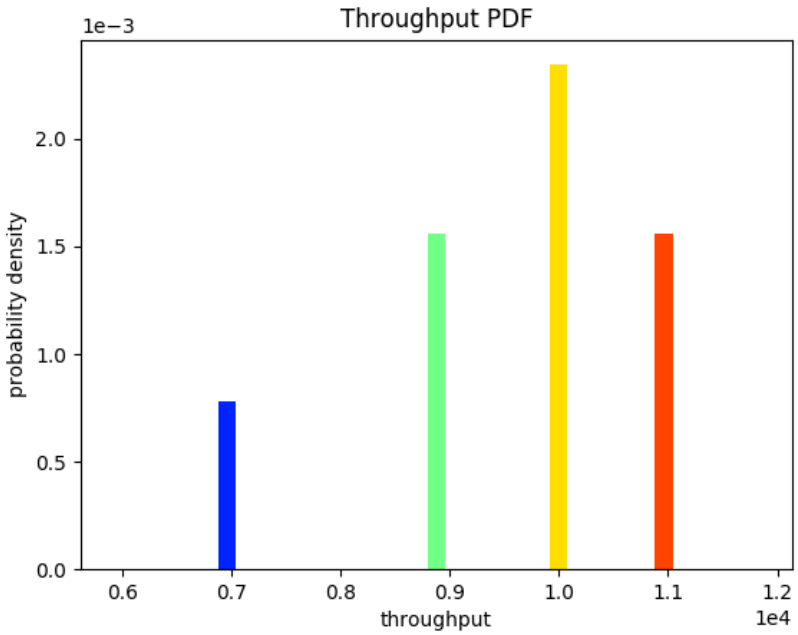
The following plots show the output of a successful measurement with two USRPs using CSMA/CA, although measurement 4 is a little bit messed up as well. Note that wrong x-axis on the line chart has been fixed by the time writing this.

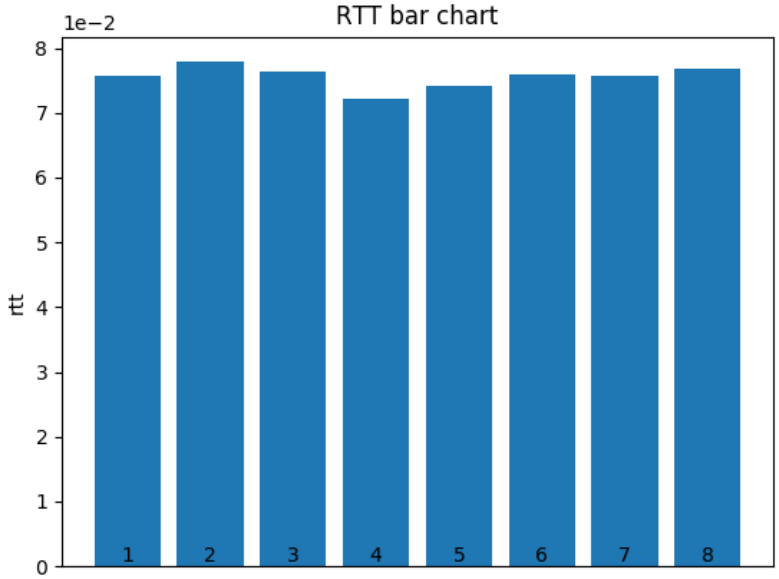
Note: Always take a look at the y-axis scaling. While such data representation might confuse the skimming reader it allows for higher resolution. Also note that not every plot adds more information, this also has the purpose of showcasing different plot types.

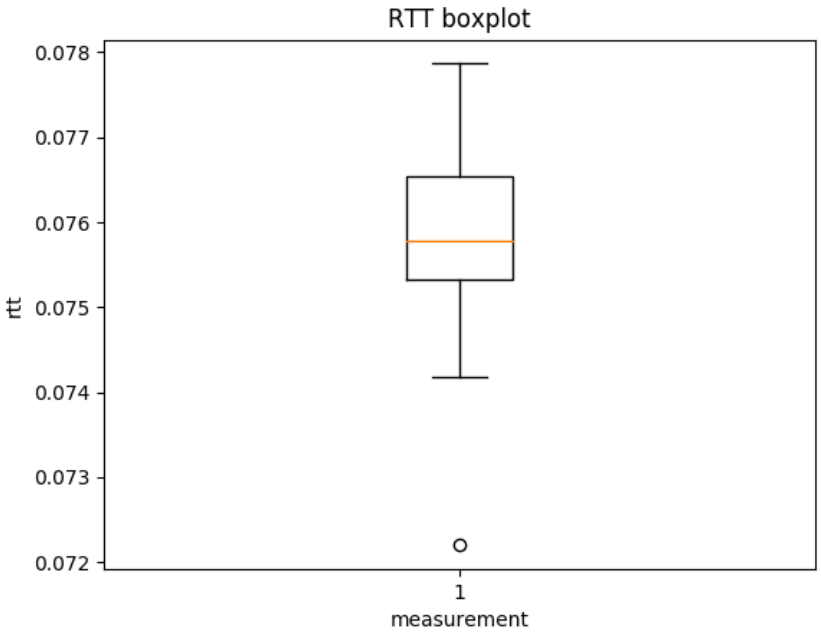


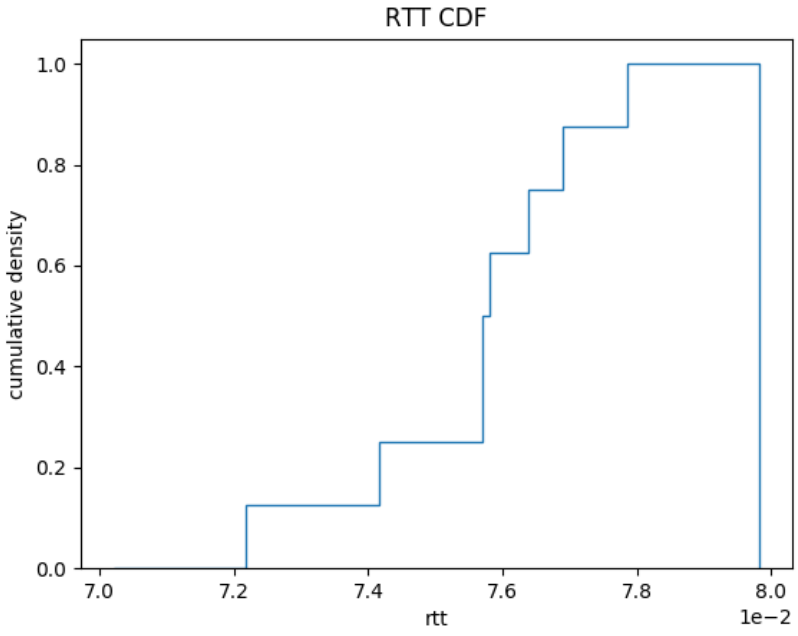

Throughput line chart
1e4
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
throughput
0
1
2
3
4
5
6
7
measurement


Throughput boxplot
11000
10500
10000
9500
9000
8500
8000
7500
7000
throughput
1
measurement


Throughput PDF
1e−3
2.0
1.5
1.0
0.5
0.0
probability density
0.6
0.7
0.8
0.9
1.0
1.1
1.2
throughput
1e4


RTT bar chart
1e−2
8
7
6
5
4
3
2
1
0
rtt
1
2
3
4
5
6
7
8


RTT boxplot
0.078
0.077
0.076
0.075
0.074
0.073
0.072
rtt
1
measurement

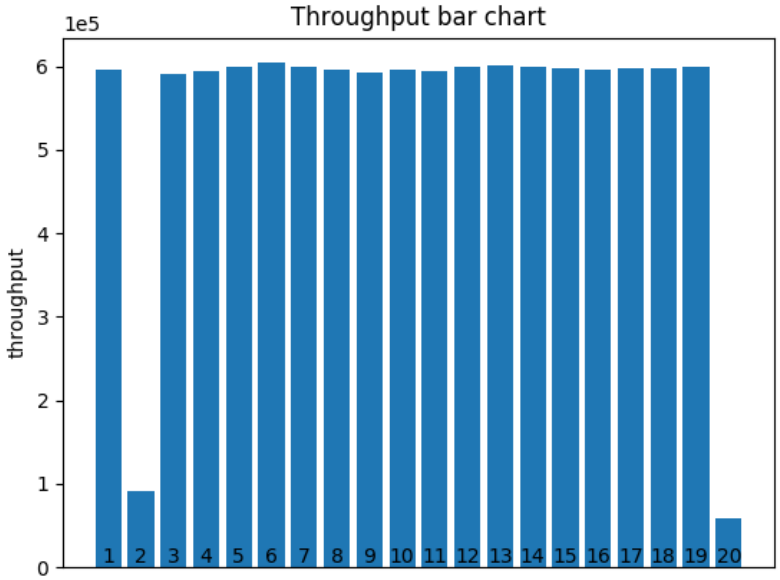

RTT CDF
1.0
0.8
0.6
0.4
0.2
0.0
cumulative density
7.0
7.2
7.4
7.6
7.8
8.0
rtt
1e−2

2.2 Failure Pattern 1

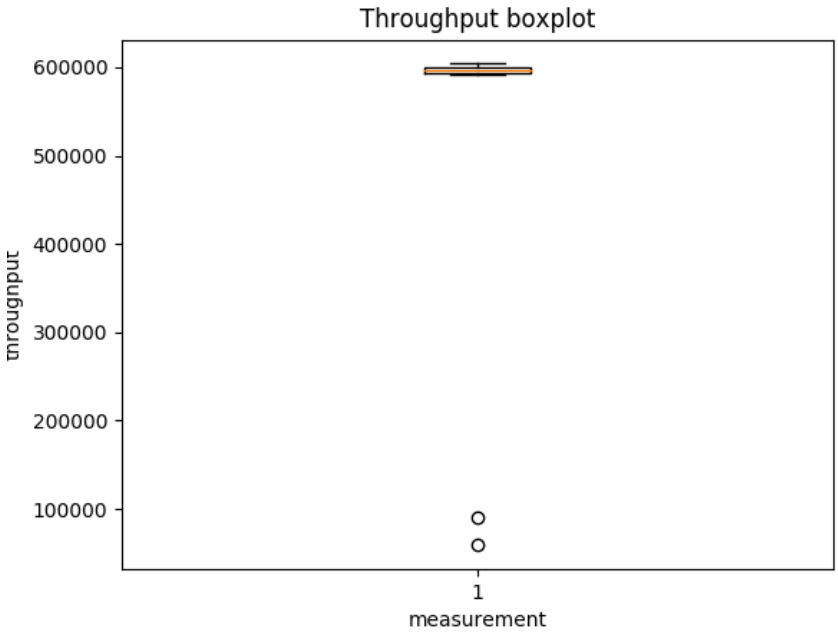
The following six figures visualize an issue I'm experiencing with the USRPs whose origin I have difficulties to track down. In seemingly random patterns the devices fail for the rest of the transmission. To make matters worse, it is not obvious whether the problem is enitrely hardware-related or has some software-related component. Let me first explain what happens in the following plots, though.

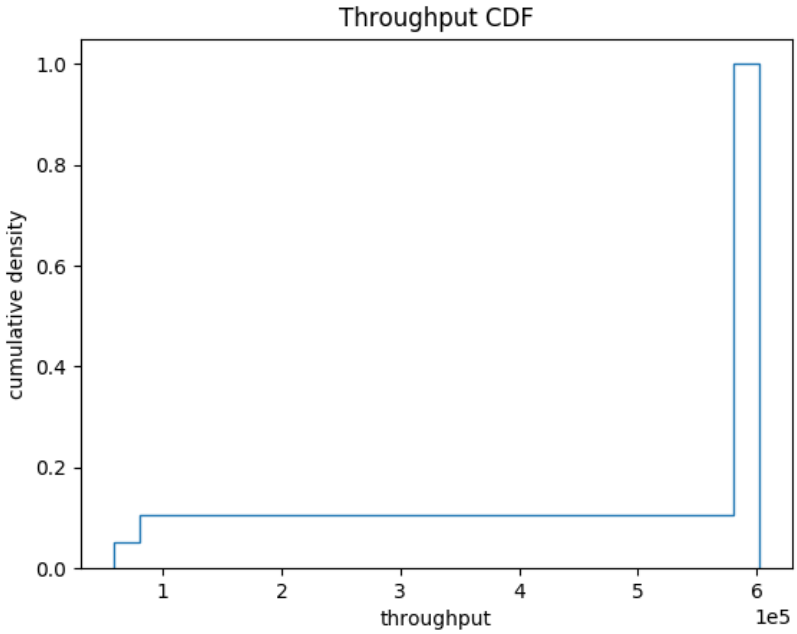
In the throughput bar chart it is obvious the device failed during the measurements 2 and 20. The logs of the measurement and this suggest that this could very well be a software problem. After 17 seconds the sender doesn't send any frames anymore (frame_probe 850 is not reached anymore). Once the sender fails it doesn't recover until the measurement script and the next measurement is launched, which is a reason to suspect a software failure on either my side or Gnu Radio's.

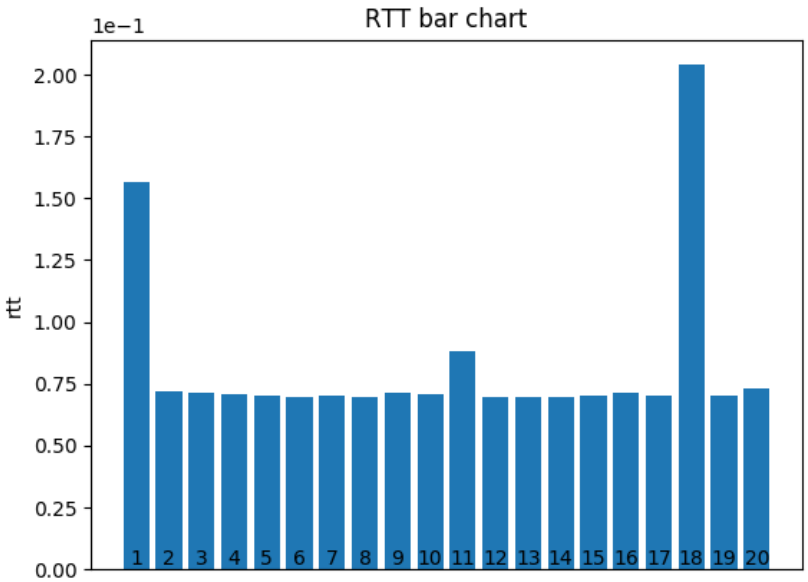
Note that the obviously wrong RTT in measurement 18 is due to the fact that retransmissions are taken into account when calculating the RTT. A fix for that is yet to be found. Alternatively, or as an alternative mode taking retransmissions into account with retransmission counters to calculate the frame delay instead of the RTT is being considered. However, at the moment there exist practical implementation barriers that are to be removed.

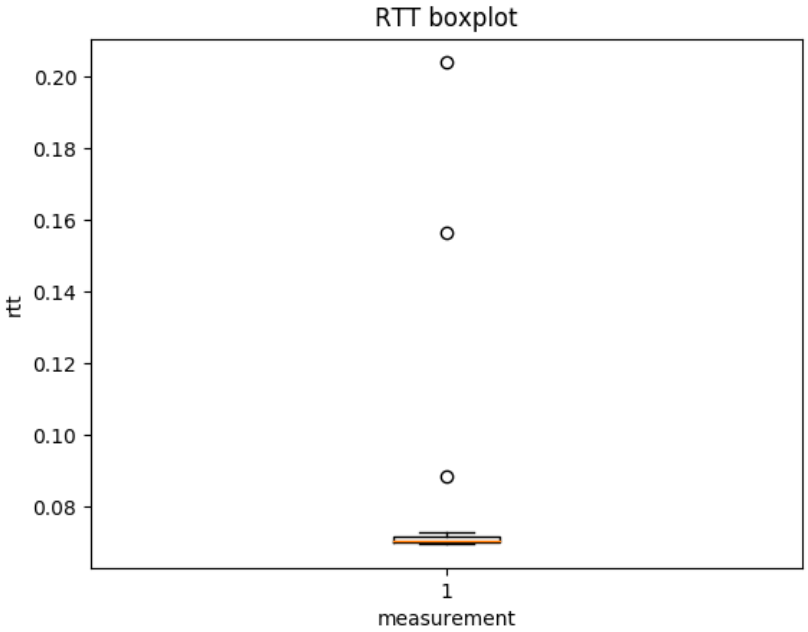


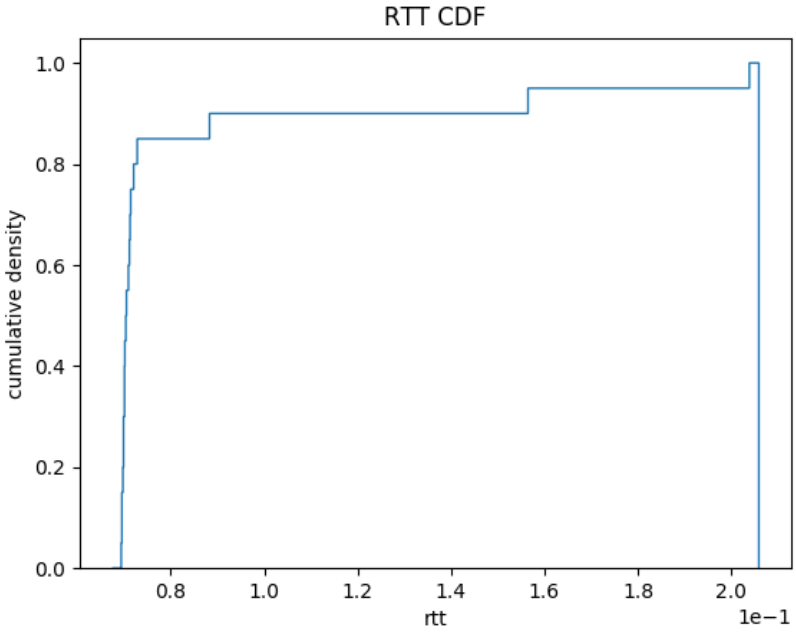
```
### This is from the measurements logs
Measurement 2/20 complete in 100 second(s).
# ... cut ...
Measurement 2/20 complete in 83 second(s).
++++ frame_probe ID: 860 receives a frame at time 71.8424s ++++
++++ frame_probe ID: 1010 receives a frame at time 71.846s ++++
++++ frame_probe ID: 890 receives a frame at time 71.8708s ++++
the 100th message counter  has been visited 89 times
++++ frame_probe ID: 850 receives a frame at time 71.9534s ++++
INFO: Detected an invalid packet at item 8544
INFO: Parser returned #f
++++ frame_probe ID: 860 receives a frame at time 71.9941s ++++
++++ frame_probe ID: 1010 receives a frame at time 71.9994s ++++
++++ frame_probe ID: 890 receives a frame at time 72.0241s ++++
the 100th message counter  has been visited 90 times
++++ frame_probe ID: 850 receives a frame at time 72.0925s ++++
INFO: Detected an invalid packet at item 8640
INFO: Parser returned #f
++++ frame_probe ID: 860 receives a frame at time 72.1332s ++++
++++ frame_probe ID: 1010 receives a frame at time 72.1437s ++++
++++ frame_probe ID: 890 receives a frame at time 72.1685s ++++
the 100th message counter  has been visited 91 times
Measurement 2/20 complete in 82 second(s).
Measurement 2/20 complete in 81 second(s).
Measurement 2/20 complete in 80 second(s).
Measurement 2/20 complete in 79 second(s).
Measurement 2/20 complete in 78 second(s).
Measurement 2/20 complete in 77 second(s).
Measurement 2/20 complete in 76 second(s).
Measurement 2/20 complete in 75 second(s).
Measurement 2/20 complete in 74 second(s).
Measurement 2/20 complete in 73 second(s).
Measurement 2/20 complete in 72 second(s).
# ... cut ...
Measurement 2/20 complete in 2 second(s).
Measurement 2/20 complete in 1 second(s).
```


Throughput boxplot
600000
500000
400000
300000
200000
100000
throughput
1
measurement


Throughput CDF
1.0
0.8
0.6
0.4
0.2
0.0
cumulative density
1
2
3
4
5
6
1e5
throughput


RTT bar chart
1e−1
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
rtt
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


RTT boxplot
0.20
0.18
0.16
0.14
0.12
0.10
0.08
rtt
1
measurement

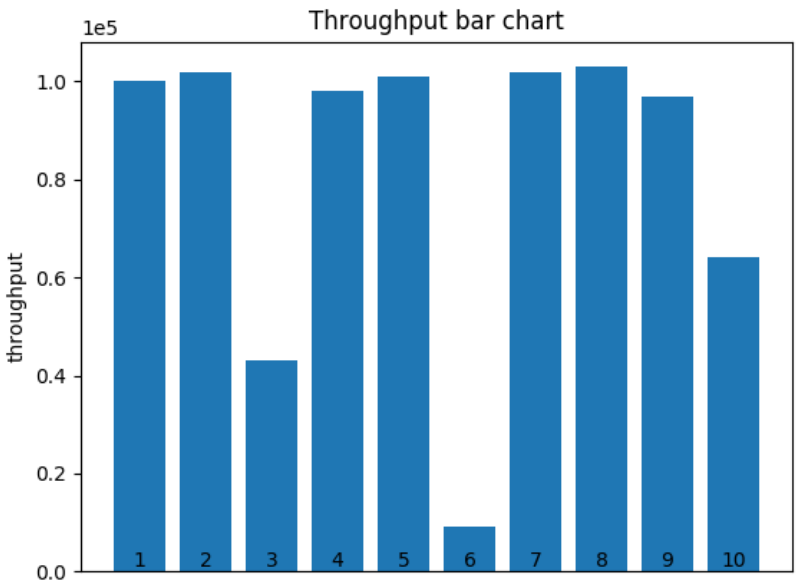

RTT CDF
cumulative density
1.0
0.8
0.6
0.4
0.2
0.0
0.8
1.0
1.2
1.4
1.6
1.8
2.0
rtt
1e−1

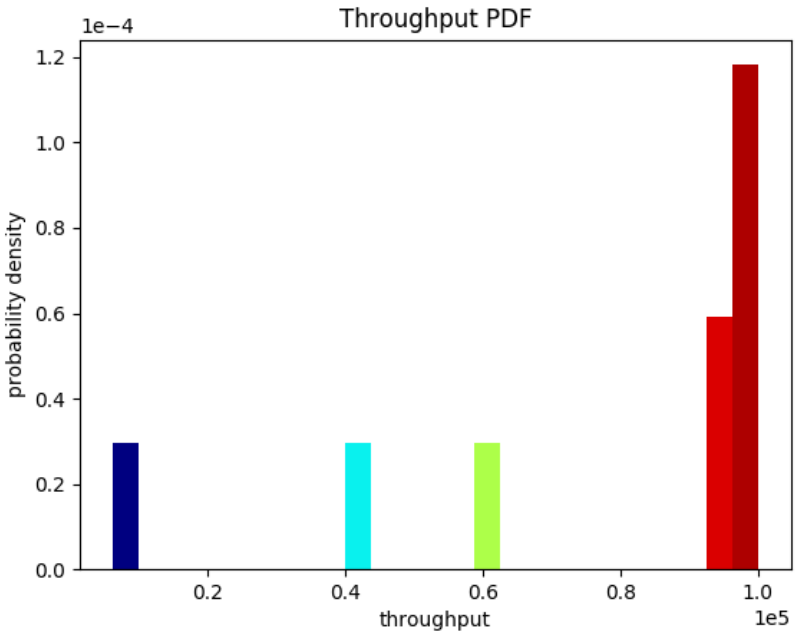
2.3 Failure Pattern 2

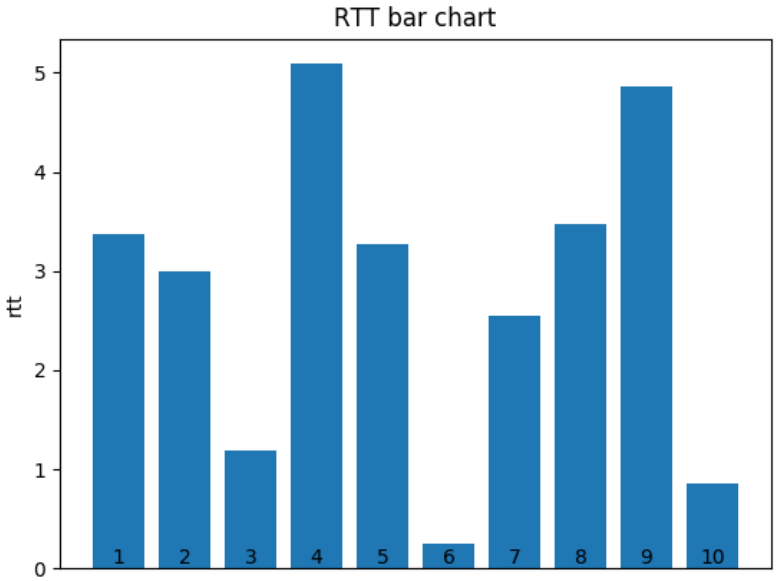
However, the failures are not permanent and show no repetitive pattern. Furthermore, during the problematic sections `uhd_find_devices` does not detect any USRPs anymore. From these two facts one can deduce the problem at least has a hardware component. Here are some plots of a typical measurement with hardware failure.

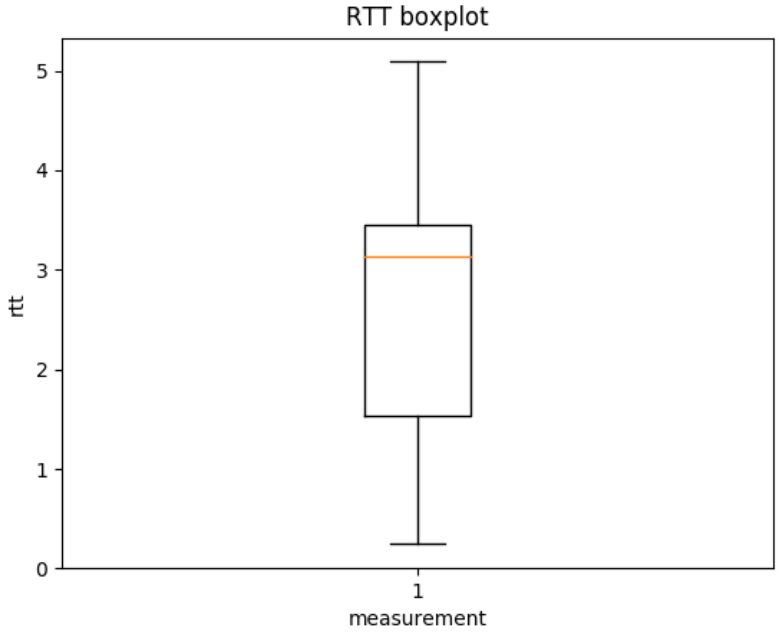
Although for 7 out of 10 measurements the throughput seems reasonable a lot of ACKs got lost during the transmission. Note that due to the problem described in the previous section the RTT calculation is wrong, because lost ACKS affect the RTT value for all following data transmissions (continuous additive error on fail).

Proof for lost ACKs can be found as mentioned in the RTT charts as well as in the logs. Note that the setup and parameters including receive and transmission gains were chosen no different from those in the previous paragraph, which is a strong indication of either hardware failure or external influence which in was avoided in preparation by choosing an unused sending frequency.




Throughput PDF
1e−4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
probability density
0.2
0.4
0.6
0.8
1.0
throughput
1e5


RTT bar chart
5
4
3
2
1
0
rtt
1
2
3
4
5
6
7
8
9
10


RTT boxplot
rtt
0
1
2
3
4
5
1
measurement

2.4 Justification of Measurement Results

In the "flawless" measurement plots both the RTT and the throughput don't differ significantly from the theoretical calculation following:
Theoretical:

$$RTT = a + b + c$$

Measured: 0.075s

Theoretical:

$$TP = t * n * c$$

Measured: tbd

The RTT measurements can be further backed up by measuring the RTT with linear increasing SIFS. The expected RTT then would be a linear function of SIFS.

Analogously, throughput should be a linear function of packet size.

Note that for both such measurements a sufficiently large measurement time must be chosen to counteract various random influences.

3 Good to Know...

3.1 Bash

3.1.1 Efficient Remote Control

```
##Opening a remote connection with X forwarding
#Using -X instead of -Y adds security check, but reduces performance
#Use -C to get it more stable (gzip-compression)
#Optional -c aes128-ctr: for AES 128 encryption
#Optional -4: forces the usage of IPv4 addresses
ssh -YC4c aes-128ctr inets@134.130.223.135

##In a second terminal mount file system to local folder
#This has the advantage of the ability to treat files as though they were local
mkdir -p mnt/134.130.223.135
cd mnt/134.130.223.135

#Tip: in the next line for the last argument type . and then expand with Tab :)
#If it is necessary sshfs also allows user-mapping if file ownership is an issue
sshfs inets@134.130.223.135:/home/inets /home/alex/mnt/134.130.223.135/

#Then edit all you like as though the mounted partition was local
cd source/gr-inets/lib
atom *some_file*
```

To further avoid any repetitive commands I added to my launcher tool:

```
"inets")
    if ( mount | grep inets  )
    then
        echo "The mount point is in use, confirm unmount with your password."
        sudo umount /home/alex/mnt/134.130.223.151
    fi
    echo "Please enter the server password to mount the target directory."
    sshfs inets@134.130.223.151:/home/inets /home/alex/mnt/134.130.223.151
    gnome-terminal --tab -e "bash -c 'ssh -YC4 inets@134.130.223.151'" \
    --tab --working-directory=/home/alex/0_ba/git/measurements \
    #--tab --working-directory=/home/alex/mnt/134.130.223.151 \
    texstudio
    sleep 1
    cd ~/0_ba/git
    git pull
    git fetch
    atom ~/0_ba/git/measurements/measurement.sh
    ;;
```

3.1.2 Listing Directories Only

Stackoverflow Source

```
##Possibility 1 (fastest):
echo */
#List all subsubfolders as well:
echo */*/

##Possibility 2 (straightforward ls):
ls -d */

##Possibility 3
#where ^ means beginning of a line
ls -l | grep "^d"

##Possibility 4
#If you need to list and process all directories in a bash-script (slow)
for i in $(ls -d */); do echo ${i%%/}; done
```

3.1.3 Redirecting GRC Stuff to Files

If one wants to do heavy analysis, built-in Linux console tools including, but not limited to utilities such as `awk`, `grep`, `sed`, `cat` might be a great help. Since executing a GRC flowgraph is running a generated python script, we can achieve our goal easily by the simple means of:

```
## Just as a concept
#These could be some lines in my alohaTestSuite.sh
mkdir -p logs
time python2 theoretical_aloha_rx.py &> logs/theoretical_aloha_rx.log
time python2 theoretical_aloha.py &> logs/theoretical_aloha.log
```

3.2 Python

3.2.1 **kwargs

3.2.2 List Comprehension

3.2.3 map and zip

3.3 Other

3.3.1 Atom: Folder-Wide Search for Substrings

Use the Ctrl+Shift+F shortcut and all "project folders" will be searched. To add multiple folders use the Ctrl+Shift+A shortcut.

3.3.2 Using a static IP address for USRPs

```
#Get to know your interface names
ifconfig
#E.g. set eth0 interface IP to 10.0.0.100/24
#Where up opens the eth0 interface
#and is not necessary if it showed up when using infconfig
ifconfig eth0 10.0.0.100 netmask 255.255.255.0 up

###Just FYI:
#Shutdown network interface
ifconfig eth0 down
##List all interfaces
ifconfig -a
#OR
ip link show
```

3.3.3 Git Personal Access Tokens

3.3.4 Latex input and include