Advanced Networking Lab

Thomas Hendriks Jakob Struye

February 26, 2017

Lab 3

Performance Measurements

In this lab, the performance and throughput in wireless networks will be investigated. Using tools like `iperf`, we will record the maximum throughput which can be achieved in wireless networks and have a look at the parameters influencing this throughput.

3.1 Bit Rates

Exercise 1: Basic throughput in IEEE 802.11a

This first exercise will give you an insight into the difference in usable throughput and the available bit rate. To determine the throughput, we will be using a tool called `iperf`. This is a client-server based tool which sends Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) traffic and reports the measured throughput. Consult the man pages for the details about this tool. A second tool to be used is `gnuplot`, in order to plot your results in a graph. More info about `gnuplot` can be found in [**?**] and a nice tutorial in [**?**]. A basic gnuplot script to generate your first plots is provided on the course website. The `iperf` tool is preinstalled on the wireless nodes, while `gnuplot` is available on the lab PCs.

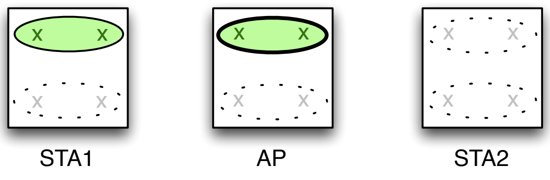


Figure 3.1: Basic throughput setup.

1. Start by configuring the setup as shown in figure 3.1. Use `fc00:grID::3/64` for the access point (AP) and `fc00:grID::1/64` for station (STA)1.

2. IEEE 802.11 supports various bit rates. Using `iw list` you can query the supported rates:
   Which rates are supported in IEEE 802.11a?
   L3-1-1

```
5.0GHz range:
    Bitrates (non-HT):
      * 6.0 Mbps
      * 9.0 Mbps
      * 12.0 Mbps
      * 18.0 Mbps
      * 24.0 Mbps
      * 36.0 Mbps
      * 48.0 Mbps
      * 54.0 Mbps
```

   Which rates are supported in IEEE 802.11b/g?
   L3-1-2

```
2.4GHz range:
    Bitrates (non-HT):
      * 1.0 Mbps
      * 2.0 Mbps (short preamble supported)
      * 5.5 Mbps (short preamble supported)
      * 11.0 Mbps (short preamble supported)
      * 6.0 Mbps
      * 9.0 Mbps
      * 12.0 Mbps
      * 18.0 Mbps
      * 24.0 Mbps
      * 36.0 Mbps
      * 48.0 Mbps
      * 54.0 Mbps
```

3. `iperf` can be used to measure the maximum throughput between two stations. Therefore, a server is started on one end and a client connects on the other end to

this server. A connection is set up and as TCP tries to maximize the throughput on a connection, an estimate of the maximum throughput on a link can be calculated. When using `iperf` with the basic parameters, it will perform a 10 seconds test and report the achieved throughput at the client side.

4. Start a basic `iperf` session between the STA and AP, with the `iperf` server on the AP:
   `AP:~# iperf -V -s`
   `STA1:~# iperf -V -c fc00:grID::3`
   Copy the output of the `iperf` client:
   L3-1-3

```
STA1:~# iperf -V -c fc00:1::3
------------------------------------------------------------
 Client connecting to fc00:1::3, TCP port 5001
TCP window size: 20.8 KByte (default)
------------------------------------------------------------
[  3] local fc00:1::1 port 47969 connected with fc00:1::3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  22.8 MBytes  19.0 Mbits/sec
```

5. Now, using `iperf`, collect the throughput for each available rate and write the results in a file `/mnt/L3-1-4.tcp.txt` On each line, first put the rate followed by a space and then the result in Mbit/s obtained from `iperf`, e.g. `54 30` denotes that a 30Mbit/s was measured when using a rate of 54 Mbps. You can change the rate used at the STA using `iw`, e.g. to 54Mbps, as follows:
   `STA1:~# iw dev wlan0 set bitrates legacy-5 54`
   You can check the actual used bit rate from the output of `iwconfig`:
   `STA1:~# iwconfig wlan0`
   As we will be repeating the collection of these results in the following exercises, it will be easier to use some bash scripting to speed up this process. A helper script can be found in the file `iperf-tcp.sh`. Change this script so it loops over the correct bitrates, and use it with the command `./iperf-tcp.sh fc00:grID::3`. This script generates output that can be directly copied to `/mnt/L3-1-4.tcp.txt`

6. `iperf` will by default use TCP to check the connection, but it is also possible to use a unidirectional UDP stream. Therefore, one can try to feed more data to the network than the network can support and as such measure how much can be actually delivered. Thus, repeat the previous scenario and collect the maximum achievable throughput using `iperf` in UDP mode for each available bit rate. Save these

results in the same format as in the previous item in a file `/mnt/L3-1-4.udp.txt` Again, a script called `udp.sh` is provided that automates this process. The script will try to send a UDP stream that fills the wireless link.

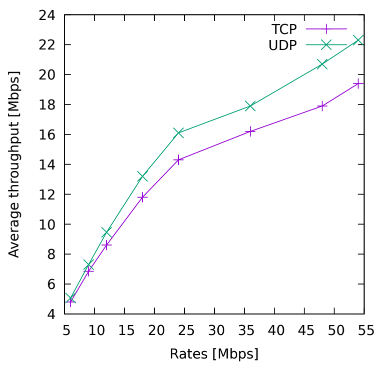
```
AP:~# iperf -V -s -u -l 1452
STA1:~# ./iperf-udp fc00:grID::3
```

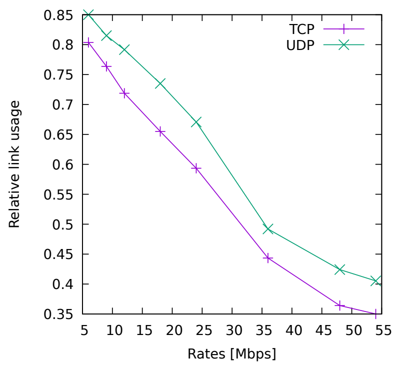
`-l` is the letter l, not the number one!

7. Now plot these results using `gnuplot`. On the course website, a `gnuplot` script, `tput1.gnuplot`, is provided to plot the throughput and the relative link usage over the various available rates. The used commands are straightforward and the script is inline commented so should be self-explanatory. Make sure you understand the various commands in the file. The script produces two PDF files which can be viewed with any regular PDF viewer. Place the `.txt` files you created in the previous steps in the same directory as the `gnuplot` script. Save the generated files in your lab report as `L3-1-4-tput.pdf` and `L3-1-4-usage.pdf`. Add the plots here and shortly discuss what can be observed. Also shortly discuss the difference between UDP and TCP. To generate the PDF files, use:

```
gnuplot tput.gnuplot
```

L3-1-4





UDP's throughput is consistenly higher than TCP's. TCP has larger headers so more overhead. More importantly, TCP's congestion control mechanisms lead to reduced throughput. Throughput grows together with bitrate: the network supports sending more bits per second with higher bitrates.

The throughput does not grow as quickly as the bitrate: with higher bitrates the bits are sent faster and bits are more likely to get lost. Due to this additional loss doubling the bitrate does not double the observed throughput. This translates to a decreasing usage with increasing bitrate.

Exercise 2: Client to client throughput

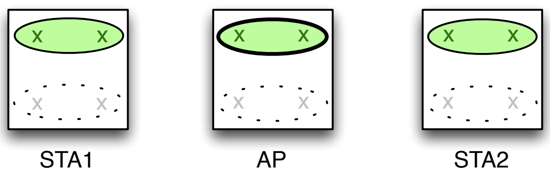
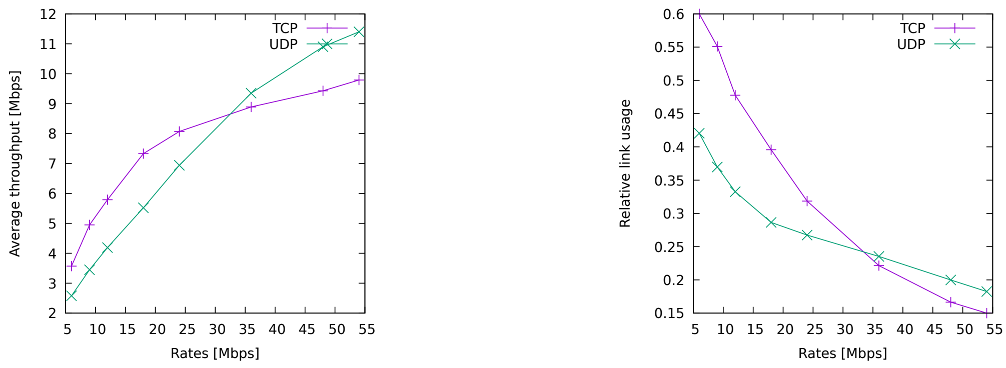


Figure 3.2: Client to client throughput setup.

In the previous exercise, the throughput was measured between a STA and the AP it was connected to. In this exercise, we are interested in the throughput between two STAs associated with the same AP.

1. Create the test setup as shown in figure 3.2. Configure STA2 with IP address `fc00:grID::2/64`.
2. Collect the TCP and UDP throughput measures as in the previous exercise and save them to `/mnt/L3-2-1.tcp.txt` and `/mnt/L3-2-1.udp.txt`
3. Adapt the provided `gnuplot` script and create graphs for the new measurements. Include them in your report.
4. Comment on the obtained results and compare the results from this exercise with those from the previous exercise. Explain the difference.
   L3-2-1



For low bitrates TCP performs better than UDP. Due to TCP's congestion control there are fewer collisions compared to UDP. For lower bitrates this makes TCP perform better than UDP. For higher bitrates the influence of TCP's additional overhead becomes larger and UDP is more performant than TCP again.

For both TCP and UDP, throughput in this case is roughly half of the previous test. Each packet is transmitted twice (to the AP and from the AP), halving the total throughput between source and destination.

The effect of increasing bitrate on the usage is similar to the previous test.

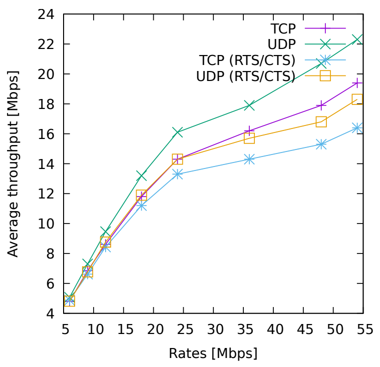
3.2 Network Settings

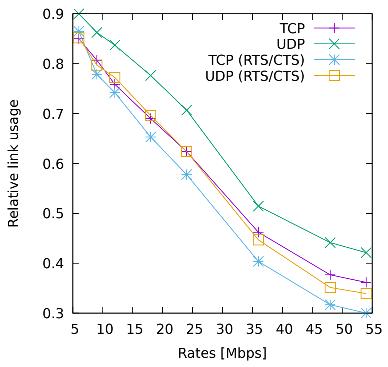
Exercise 3: RTS/CTS

In lab 2, we enabled the Request to Send (RTS)/Clear to Send (CTS) mechanism to show that in IEEE 802.11, stations can reserve the medium in order to avoid collisions. In this exercise, we will study the effect of RTS/CTS on throughput.

1. Enable RTS/CTS on all devices.
   `:~# iw phy phy0 set rts 1000`

2. Repeat the iperf tests from STA1 to AP. Save your logs to `/mnt/L3-3-1.tcp.txt` and `/mnt/L3-3-1.udp.txt` Again modify the gnuplot script to create new graphs and include them in your report. To make the comparison more easy, modify the gnuplot script so that you plot both the results from exercise 1 and this exercise on the same graph. What do you observe? Why?
   L3-3-1





For both TCP and UDP throughput is consistently lower with RTS/CTS enabled. The only communication in our tests is direct traffic between STA1 and AP (only STA1 $\rightarrow$ AP in case of UDP). There were few collisions even without RTS/CTS. This means RTS/CTS mainly adds overhead, leading to lower throughput.

Even with RTS/CTS enabled UDP still outperforms TCP: TCP's lower performance is caused by its additional overhead and congestion control, which is not changed by having RTS/CTS enabled.

Exercise 4: Frame length

The standard Maximum Transferable Unit (MTU) for ethernet frames is 1500 bytes. IEEE 802.11 however allows an MTU up to 2274 bytes. In this exercise, you will measure the effect of frame size on throughput.

1. Continue from the previous setup, but make sure RTS/CTS is turned off on all nodes:
   ```
   iw phy phy0 set rts off
   ```

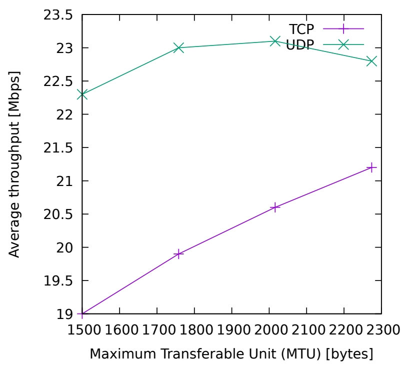
2. On STA1, reset rate control to its default (automatic) value:
   ```
   STA1:~# iw dev wlan0 set bitrates
   ```

3. Perform the following tests with 4 different maximum frame sizes: 1500, 1758, 2016 and 2274. The frame size can be controlled by changing the MTU. *If you do this, do this on all nodes!*
   ```
   ifconfig wlan0 mtu 1758
   ```

4. For each of the MTUs mentioned, start with a TCP `iperf` test between STA and AP as in the previous exercises and make plots for each frame size. Generate a results file called `/mnt/L3-4-1.tcp.txt` containing MTU and bit rate achieved. The file should have the same format as the ones produced this far, except that the first column now contains your MTU setting rather than the link speed.
   ```
   STA1:~# iperf -V -c fc00:grID::3
   ```

5. Repeat the tests, now using UDP. For `iperf`, use the `-l` parameter, followed by the current MTU setting minus 48 (e.g. 1452 if the MTU is set to 1500) to generate packets that are large enough to fill the MTU. ***This must be done on both client and server!*** Save your results to `/mnt/L3-4-1.udp.txt`.
   ```
   STA1:~# iperf -V -u -l 1452 -c fc00:grID::3 -b 54M
   ```

6. Modify the `gnuplot` script to generate the same graph as before. Plotting only the throughput versus MTU suffices. A relative link usage graph is not required. What do you observe? Why? Be as precise as possible.
   L3-4-1



UDP has a fairly small header per datagram. As overhead per datagram is already fairly limited, there is not much room for optimization by putting more data in each datagram. Additionally a larger datagram is more likely to get lost in a wireless network than a smaller one. We observe this when increasing the MTU from 2016 to 2274: the additional packet loss due to the larger frame size outweighs the decrease in per-bit overhead, leading to a lower throughput.

In the case of TCP there is more per-bit overhead, meaning there is larger room for improvement by putting more data in each segment. Increasing the MTU consistently increases throughput for the tested range: additional packet loss never outweighs the performance increase of reducing the per-bit overhead. This is however never enough to overtake UDP performance.

3.3 Packet Size

Exercise 5: IP Fragmenting

Changing frame lengths has an effect on the throughput, as you have shown in the previous exercise. However, in that case, traffic was flowing between segments with the same fragment size. When we consider the default setup of an AP which is connected to the Internet, the wide area network (WAN) link is most likely limited to 1500 bytes per frame and thus IP fragmenting comes into play. In IPv6, fragmenting in intermediate hops is not allowed. The end hosts may, however, fragment the IP packets end-to-end to make sure they fit on the link with the smallest MTU. Depending on the original payload length, fragments of various lengths will be created. In this exercise we will have a look at the impact of fragmenting on the overall throughput.

To do so, we will run the `iperf` server on the third node, which is only connected by wire to the AP. Using additional routes, we will create a setup where STA1 and STA2 communicate via the AP. Traffic between STA1 and the AP will be transmitted on the wireless link, while traffic between the AP and STA2 will be transmitted on the wired link.

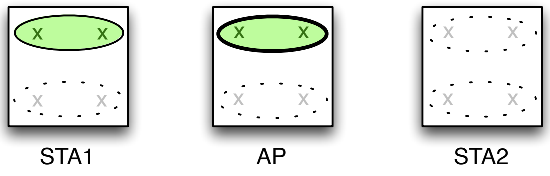


Figure 3.3: Fragmenting setup.

1. Start by configuring your setup as shown in figure 3.3. STA2 will only be used as `iperf` server and does not need an active wireless network interface card (wnic).

2. Set `fc00:grID::3/64` as IP for the AP and `fc00:grID::1/64` for STA1.

3. Traffic from STA1 to STA2 will by default use the wired network. To force a wireless hop, we will add some new routes:
   ```
   STA2:~# ip route add fc00:grID::/64 via <wired IP address of AP>
   STA1:~# ip route add <wired IP address of STA2>/128 via fc00:grID::3
   ```

4. IPv6 forwarding is by default not enabled on the lab nodes. We must enable it on the AP for the setup to work. However, due to a bug, the AP will clear its default route when we do this, making it instantly inaccessible over IPv6. Perform the following as a workaround:
   ```
   AP:~# ip -6 route show | grep default
   ```
   You should get output like this:
   ```
   default via fe80::225:90ff:fe61:efac dev eth0 proto kernel metric 1024
   expires 1670sec
   ```

5. Enable IPv6 routing functionality on the AP:
   ```
   AP:~# sysctl -w net.ipv6.conf.all.forwarding=1 && ip route add default
   via fe80::225:90ff:fe61:efac dev eth0
   ```

   The IP address in that command should be the same as that in the output of the previous step.

6. Perform a `traceroute6` from STA1 to STA2 to check if the routes are set up correctly. Include your traceroute6 output below:
   `STA1:~# traceroute6 <wired IP address of STA2>`

   L3-5-1

   ```
   STA1:~# traceroute6 2001:6a8:500:e081:20d:b9ff:fe18:235c
   traceroute to 2001:6a8:500:e081:20d:b9ff:fe18:235c (2001:6a8:500:e081
       :20d:b9ff:fe18:235c) from fc00:1::1, 30 hops max, 16 byte packets
    1  fc00:1::3 (fc00:1::3)  1.965 ms  0.984 ms  0.909 ms
    2  wmn3.mosaic.uantwerpen.be (2001:6a8:500:e081:20d:b9ff:fe18:235c)
        1.455 ms  1.027 ms  0.986 ms
   ```

   The MAC addresses show the wireless interface's MAC address of AP and the wired interface's MAC address of STA2.

7. Now, for each MTU (1500, 1758, 2016 and 2274), perform the following:

   - Set the MTU on the wireless link.
   - Perform a TCP `iperf` from STA1 to STA2.
   - Perform a UDP `iperf` from STA1 to STA2. In each trace, set your frame size to the MTU of the wireless link minus 48. Send at a bit rate of 54 Mbps.

8. The results of your tests should once again be saved to text files like in the previous exercise. Save them to `/mnt/L3-5-2.tcp.txt` and `/mnt/L3-5-2.udp.txt`. Create a throughput graph as in the previous exercise.

9. You should also make `.pcap` trace files from these experiments. However, as the link gets fully saturated, the trace file would become very large. Therefore, we will repeat the experiments sending far less traffic. This of course means that you will not see a difference in `iperf` results. The traces will help you, however, to understand what has happened in the previous exercise.

10. In order to limit the amount of traffic, we will do the following on STA1:

    - Set the rate of `wlan0` to 6 Mbps.
    - for both TCP and UDP `iperf`, add the `-t 2` parameter. This limits the `iperf` trace to two seconds.
    - for UDP `iperf`, omit the `-b 54M` parameter. This way, `iperf` will only transmit at 1 Mbps.

11. Now, do the same `iperf` tests again, using the above parameters for `iperf`. For each run, make a `tcpdump` trace on interface `wlan0` of the AP. Save the traces to `/mnt/L3-5-2.tcp.<MTU size>.pcap` and `/mnt/L3-5-2.udp.<MTU size>.pcap`, respectively.

12. What do you observe? Distinguish between the behaviour for TCP and UDP. Explain your findings by including your graphs and refering to the trace files you made. Be as precise as possible.
    L3-5-2

TODO MAKE FRAGMENTING WORK WITH TCP IF POSSIBLE

Acronyms

AP	access point
CTS	Clear to Send
grID	group ID
MTU	Maximum Transferable Unit
nic	network interface card
RTS	Request to Send
STA	station
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WAN	wide area network
wnic	wireless network interface card