WirelessLab WS 2016/17

Homework 10: Transport over Wireless Part II

Group 6

Gasper Kojek, Jens Klein

Question 1: Communication between Station and AccessPoint

a) Setup and b) Data gathering

Already covered in previous tasks.

c) Questions:

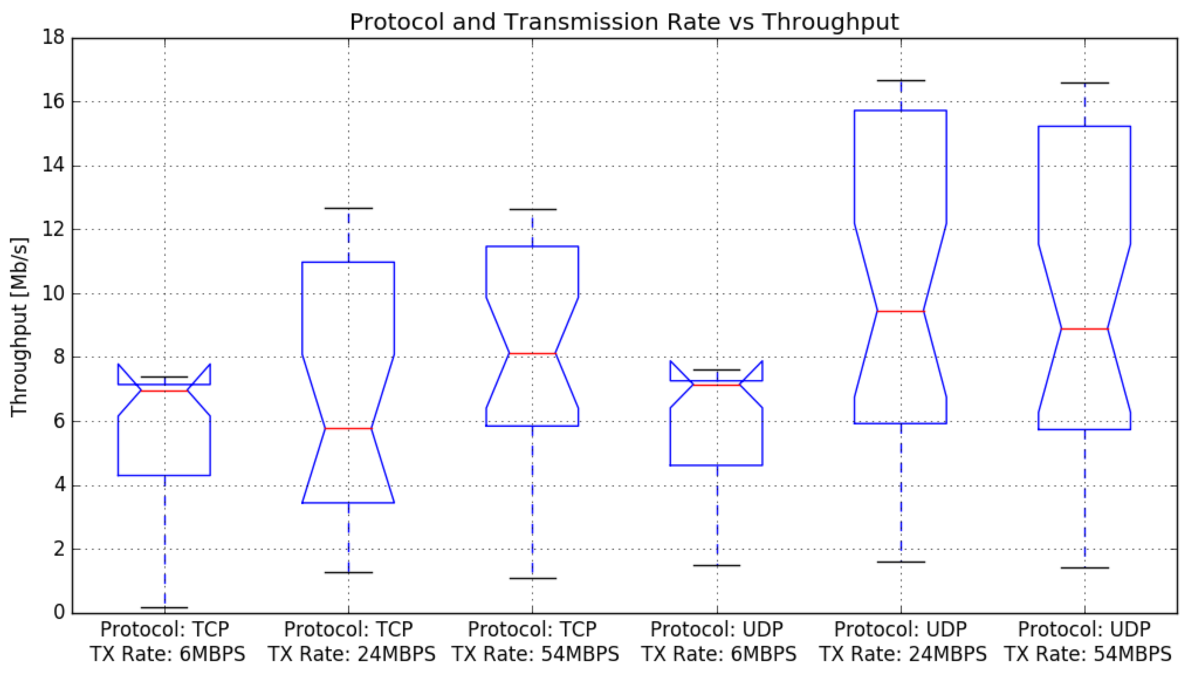
For questions 1 - 3 we used the data from last assignmnet, as stated on ISIS

For questions 4 - 6 we generated new data, using new instructions posted on ISIS: UDP packet size, TCP Reno/Cubic

We started with the data gathering on Tue, 24 Jan 2017 12:23:30 GMT and ended on Tue, 24 Jan 2017 21:40:46 GMT. We used a script (mostly the same as in last homework) which did a test run every 15 minutes (7 UDP packet sizes, 2 TCP congestion protocols).

1. What is the measured TCP throughput in infrastructure mode from station to access point? Compare this throughput with the physical data rate of 54 Mbps that the hardware is configured with. Can you explain the difference (generic, no calculations needed)?

Graph with results from the last assignment (assignment 9):



The difference comes from the fact that the physical data rate of 54Mbps includes all of the traffic, including all headers on each packets (IP header, TCP header, MAC header), all control and management packets (ACKs, etc.). Even if we calculated the throughput of all of bits sent/received, it would still be lower because of lost and retransmitted packets and all the different windows, delays and backoffs of underalying protocols.

2. Is the TCP throughput lower compared to the UDP throughput? If yes, why?

Yes, TCP throughput consistently was lower than UDP (except for 6Mbps, where this setting was the bottleneck). TCP introduces additional overhead, which makes it more robust at the expense of throughput. If we had a really nosy environment TCP would probably perform better in regards of throughput. If we need to implement packet sequence / error correction and detection on top of UDP, TCP would probably perform better as well, because it takes care of all of that.

3. What is (are) the limiting factor(s) in the TCP protocol?

The core principles and their implications in TCP protocol:

- TCP three-way handshake introduces a full roundtrip of latency.
- TCP slow-start is applied to every new connection.
- TCP flow and congestion control regulate throughput of all connections.
- TCP throughput is regulated by current congestion window size.

As a result, the rate with which a TCP connection can transfer data in modern high-speed networks is often limited by the roundtrip time between the receiver and sender. Further, while bandwidth continues to increase, latency is bounded by the speed of light and is already within a small constant factor of its maximum value. In most cases, latency, not bandwidth, is the bottleneck for TCP.

4. What is the optimum packet size (accuracy 1 byte) and the corresponding data rate for maximum UDP throughput?

Commands used:

For this (and next question) we went with UDP throughput with variable packet size (option `b)` on ISIS), as this was something we haven't done before and were interested in the results.

We still use 54Mbps as the transmission rate on both nodes.

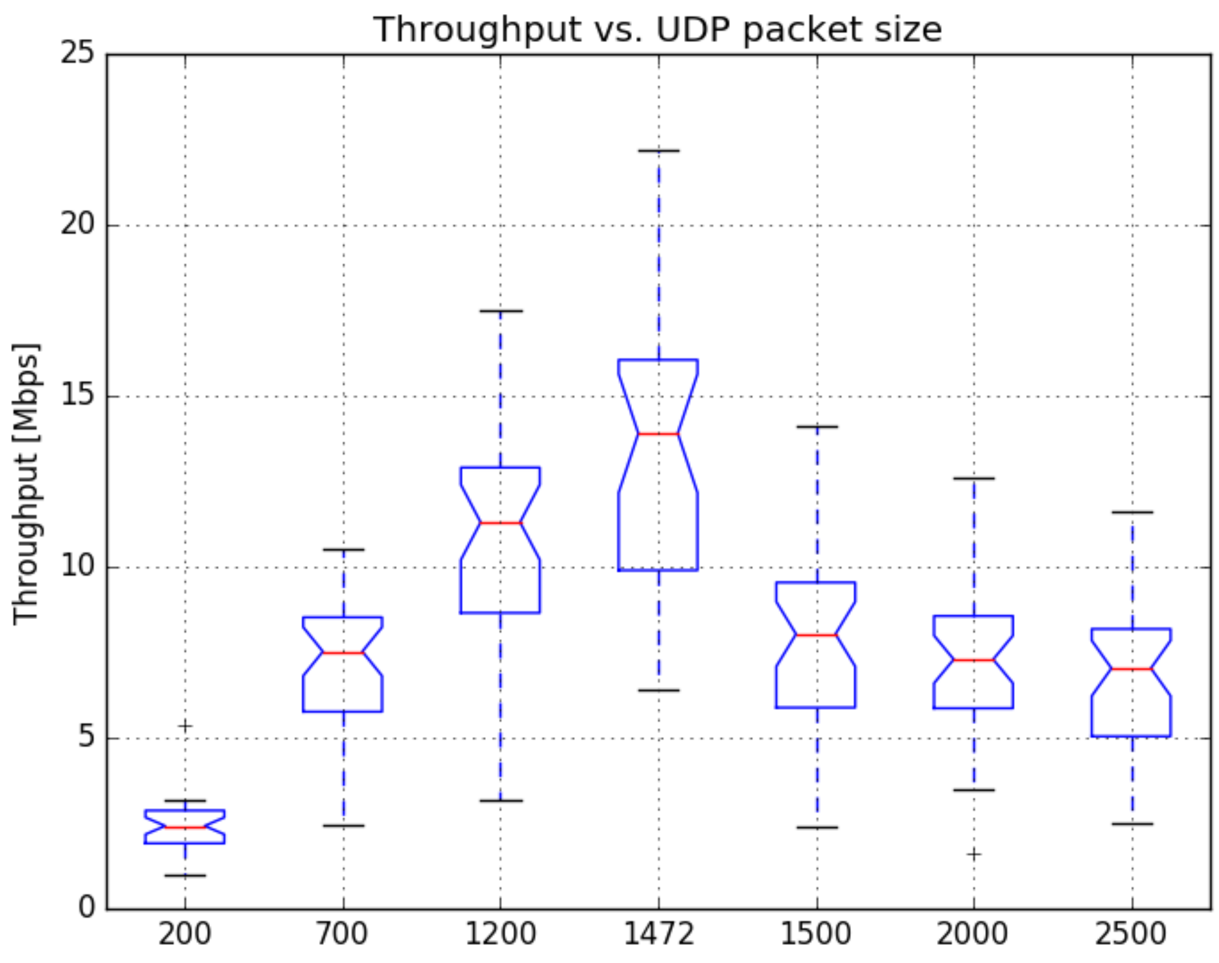
Access point: `iperf -s -u`

- `-u` : use UDP instead of TCP

Client: `iperf -c 172.17.5.10 -u -b 50M -t 60 -l <size>`

- `-u` : use UDP instead of TCP
- `-b 50M` : We used 50Mbps as stated on ISIS
- `-t 60` : Each run lasts 60s
- `-l <size>` : select UDP packet size

Our test results:



Explanation:

The optimum UDP packet size is usually 1472 bytes for Ethernet. For Ethernet the MTU (Maximum transmission unit) is usually 1500 (which is also in our nodes). UDP header size is 8 bytes and IPv4 header size is usually 20 bytes (but can be up to 60 bytes). In the case of Ethernet, the IP packet will additionally be wrapped in a MAC packet (14 byte header + 4 byte CRC) which will be embedded in an Ethernet frame (8 byte preamble sequence). This adds 26 bytes of data to the IP packet, but doesn't count against the MTU.

To get the optimal UDP packet length:

```
1500 MTU - 20 IPv4 header - 8 UDP header = 1472 bytes
```

We have also confirmed that UDP packets with data length of 1472 bytes have the highest throughput, which has a median of ~13 Mbps.

References:

- StackOverflow

- MTU
- PDU
- IPv4
- UDP
- Ethernet frame

5. Why is the throughput lower if you use a packet size higher than the optimum? Explain the optimum value as identified previously.

The Maximum Transmission Unit (MTU) is the largest possible frame size of a communications Protocol Data Unit (PDU). For most Ethernet networks this is set to 1500 bytes and this size is used almost universally on access networks. Our nodes are configured with MTU of 1500 bytes as well. As stated before, UDP header size is 8 bytes and IPv4 header size is usually 20 bytes (but can be up to 60 bytes). In the case of Ethernet, the IP packet will additionally be wrapped in a MAC packet (14 byte header + 4 byte CRC) which will be embedded in an Ethernet frame (8 byte preamble sequence). This adds 26 bytes of data to the IP packet, but doesn't count against the MTU.

The explanation for the drop of throughput is that because when we exceed the MTU, which we do if we set the packet size higher than the optimum, the sender has to fragment (split) the packet into two packets to fit in the smaller size tunnel (specified with the MTU). This includes additional processing, overhead and new packet(s), which lowers the throughput. Because we need 2 frames on the medium to transmit one UDP packet the throughput drops by roughly 50%, which can be observed on the graph as well.

6. Check the congestion control algorithm in your node `Hint : sysctl -a | grep congestion`. Please carry out measurements twenty times each on TCP Reno and TCP Cubic by sending two parallel iperf streams distributed during the day and plot the throughput and explain your observations.

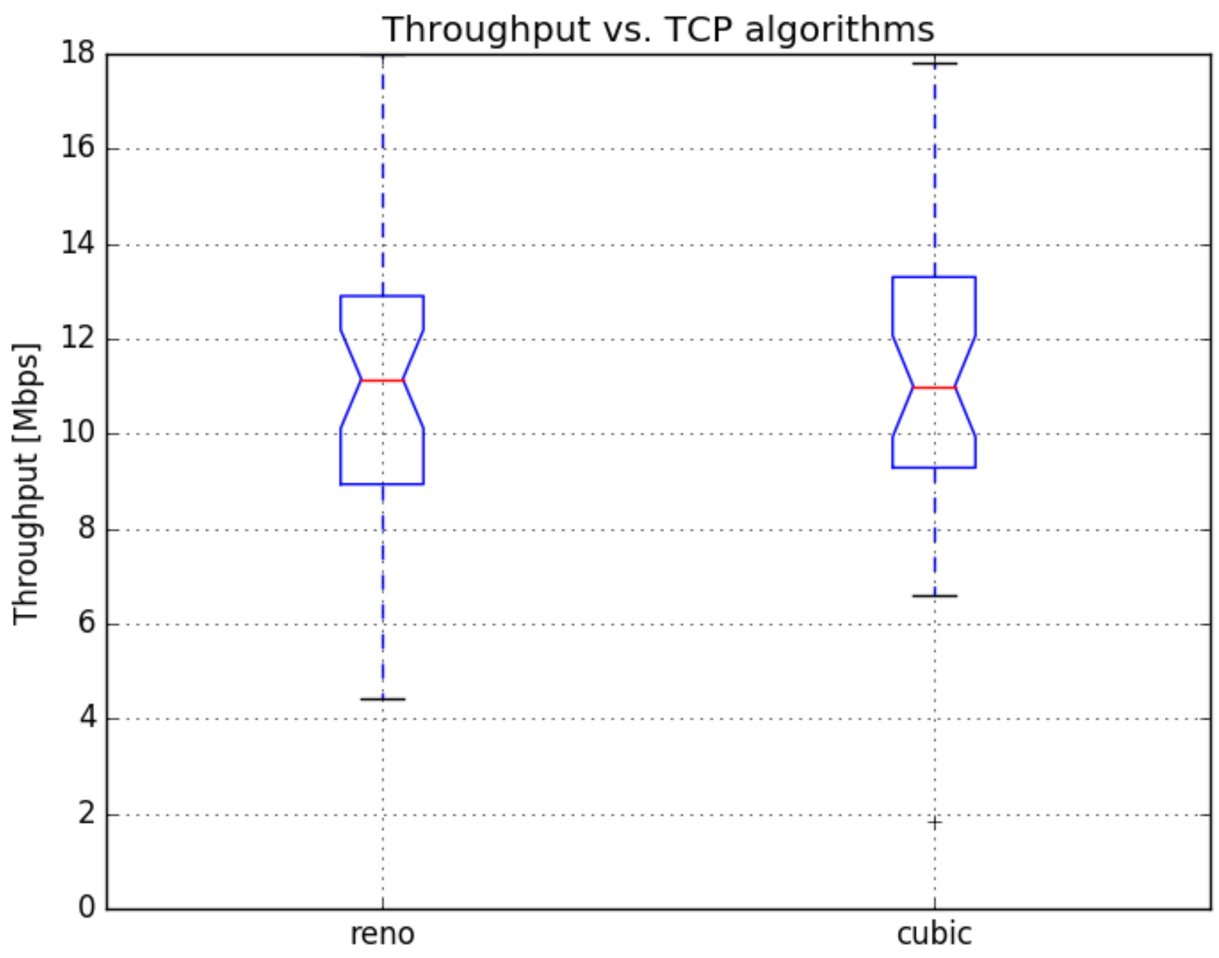
Commands used:

Access point: `iperf -s`

Client: `iperf -c 172.17.5.10 -b 56M -t 60 -Z <reno/cubic> -P 2`

- `-b 56M` : We used 56Mbps to saturate the medium
- `-t 60` : Each run lasts 60s
- `-Z <reno/cubic>` : Choose the congestion control algorithm
- `-P 2` : Do 2 TCP streams in parallel

Our test results:



Explanation:

When we compare Reno and Cubic we can see that there is no real difference. Reno median is a bit higher than Cubic, but their confidence intervals overlap, so that doesn't tell much. We can see that Cubic may be more stable, as it has narrower whiskers. This both falls in line with other comparisons that we can find on the internet. Cubic is stated as the most stable of TCP congestion control algorithms, it is also the one used by default with linux kernels 2.6.19 and upwards. Reno is used by default with Windows XP and earlier (some use New Reno - not tested here). Both should work sufficently well for general networking scenarios.

References:

- TCP congestion control

Question 2: Communication between two Ad-Hoc nodes

Terminology

- N6: Node 6
- N15: Node15
- ST: Stepping Stone
- PC: Personal Computer

a) Setup

- First we setup up, what was the AP (N6):

  - Disable the AP interface: `ifconfig wlan0 down`
  - Add ibss type i-face: `iw phy phy0 interface add ah0 type ibss`
  - Bring it up: `ifconfig ah0 up`
  - Check it: `ifconfig ah0`

    Output:

    ```
    ah0       Link encap:Ethernet  HWaddr 00:1B:B1:07:DB:9B
              UP BROADCAST MULTICAST  MTU:1500  Metric:1
              RX packets:0 errors:0 dropped:0 overruns:0 frame:0
              TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
    ```

  - Create ad hoc network: `iw dev ah0 ibss join grp6-adhoc 2462` We expect most of the other students to take the example frequency on CH1 so we go on CH11 (2,462 GHz)

  - Check settings `iw dev ah0 info`

    Output:

```
Interface ah0
    ifindex 1830
    wdev 0x392
    addr 00:1b:b1:07:db:9b
    ssid grp6-adhoc
    type IBSS
    wiphy 0
    channel 11 (2462 MHz), width: 20 MHz (no HT), center1: 2462 MHz
    txpower 30.00 dBm
    txpower 30.00 dBm
```

looks good.

- Assign it a static inet address

  `ifconfig ah0 172.17.5.10` `ifconfig ah0 netmask 255.255.255.0`

- Setup the second participant which was the STA before on N15:
  - Disable the STA interface: `ifconfig wlan0 down`
  - Add ibss type i-face: `iw phy phy0 interface add ah0 type ibss`
  - Bring it up: `ifconfig ah0 up`
  - Check it: `ifconfig ah0`

    Output:

```
ah0       Link encap:Ethernet  HWaddr 00:1B:B1:01:DC:B4
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

  - Join ad hoc network: `iw dev ah0 ibss join grp6-adhoc 2462`

  - Check settings `iw dev ah0 info`

    Output:

```
Interface ah0
    ifindex 915
    wdev 0x390
    addr 00:1b:b1:01:dc:b4
    ssid grp6-adhoc
    type IBSS
    wiphy 0
    channel 11 (2462 MHz), width: 20 MHz (no HT), center1: 2462 MHz
    txpower 30.00 dBm
    txpower 30.00 dBm
```

looks also good.

- Assign it a static inet address

  `ifconfig ah0 172.17.5.11` `ifconfig ah0 netmask 255.255.255.0`

- Check if with ping if a connection is establishable
  - N6: `ping 172.17.5.11`

    Output (first two lines):

    ```
    PING 172.17.5.11 (172.17.5.11): 56 data bytes
    64 bytes from 172.17.5.11: seq=0 ttl=64 time=0.963 ms
    ```

  - N15: `ping 172.17.5.10`

    Output (first two lines):

    ```
    PING 172.17.5.10 (172.17.5.10): 56 data bytes
    64 bytes from 172.17.5.10: seq=0 ttl=64 time=0.955 ms
    ```

- Make sure we have the exact same HW settings as on the previous experiment. (Commands executed on N6 and N15)
  - Transmission Power: `iw ah0 set txpower fixed 100` (1dBm)
  - Transmission Rate: `iw ah0 set bitrates legacy-2.4 54`
  - Kernel mods remain untouched

b) IBSS vs BSS Throughput

Independend and Infrastructure mode performance comparison

We try to give a theoretical comparison of the two modes in terms of throughput performance. Given the setup constraints we can state that there are no hardware factors on the throughput because the setups are identical (e.g. Channel, TxPower, TxRate ...) only the experiment time is different. To keep this influencing factor as low as possible we acquire the date spread over the day which gives as a good profile.

The difference in the modes that can impact the performance is the way the network is managed. In an BSS the AP basically the manager of all to it connected nodes, but in an IBSS there is no such responsibility. That introduces additional traffic on the medium namely beacon frames, which every node in an IBSS sends out periodically. With a growing number of nodes the number of beacons grows and occupies the medium. In an BSS the number of beacons is independent from the number of STA in range.

The minimal network of an BSS consists of an AP and a STA where we have one beacon emitter (AP), in an IBSS we have two STA that both send out beacons. So we'd expect a slightly worse throughput in IBSS mode. But that's might not noticeable. But the throughput would worsen with additional STAs broadcasting beacons, whereas in an BSS that wouldn't have an impact.

Experiment Setup

- start iperf server on N6 with `iperf -s`
- execute script `capture_script.sh` on N15:

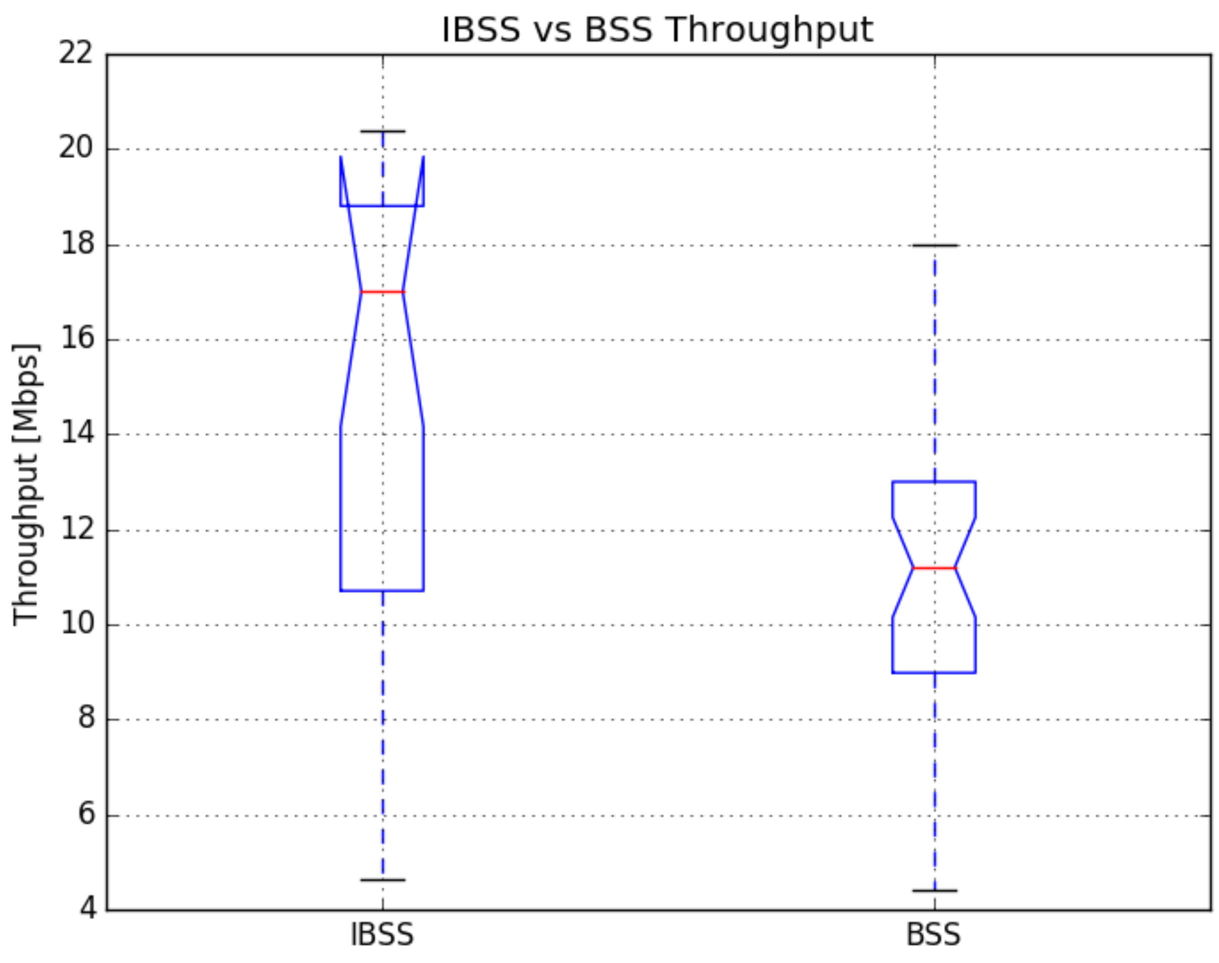
  ```
  for i in `seq 10`; do
    echo "Timestamp: $(date +%s)"

    iperf -c 172.17.5.10 -b 56M -t 60 -Z reno -P 2
    sleep 14m
  done
  ```

- started at Wed, 25 Jan 2017 14:03:32 GMT, did 10 runs every 15min

- started another at Wed, 25 Jan 2017 17:47:04 GMT, did 10 runs every 15min

On run takes 60 s then we wait for 14 min and run the next round. The iperf run is identical to that one in question 1, task c subtask 6. We took tcp reno for comparison.

Test results



Explanation

The left plot is the throughput boxplot of the experiment in an ibss network mode. On the right side for comparison, we have the corresponding from the previous task.

The ibss network mode in general performed better in terms of throughput (median at 17 Mbps) than the bss (median at 11 Mbps). But the ibss plot reveals that in this experiment the throughput is much more volatile. This result is not what we expected in our presumption.

We think that the difference between these data sets can only be caused by the different days we captured the data. Although we tried to match the time of the day the traffic can be totatly different on two distinct days.