


- (b) In the app's statistic section, you can also find charts illustrating the buffer and throughput. Make a screenshot of both of them for all three scenarios and interpret the results!
- (c) When you are asked to rate the video's quality, you are also asked how much you liked the video's content. Why might this be of interest for the app developers?
- (d) How is the concept called that YoMoApp is using for monitoring YouTube streaming quality? Shortly describe the concept. In this context, also name some possible benefits and drawbacks.

Answer (a):

Statistics for scenario 1, device is connected to Wi-Fi:

YoMo statistics for video watched while the device was connected to Wi-Fi, taken from the images below-

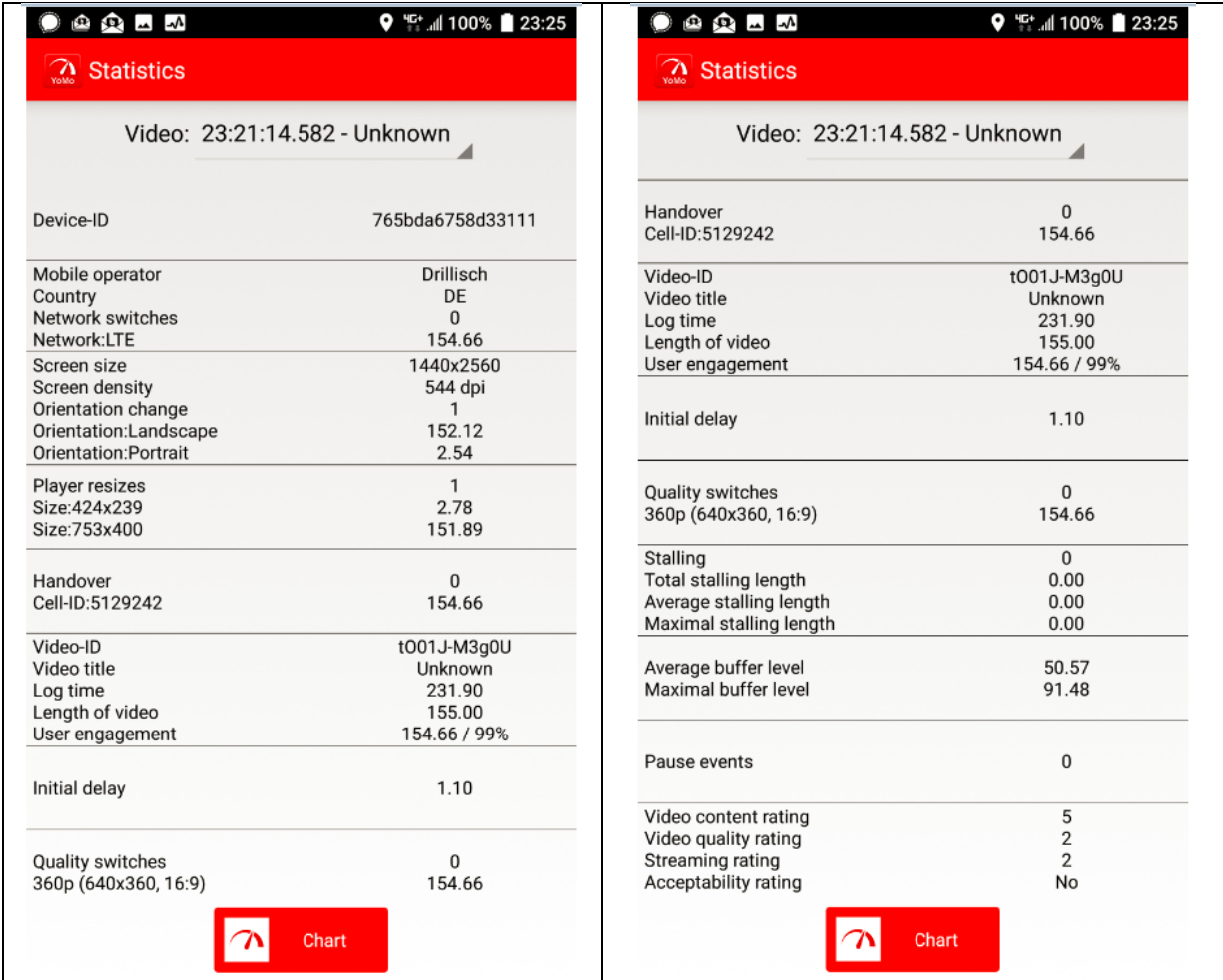
<u>Parameter</u>	<u>Value</u>
duration of the <i>initial delay</i> [seconds]	1.08
how often the <i>video paused</i> during play black	video stalled 0 times
the number of <i>quality switches</i>	0
<i>average</i> buffered play time [seconds]	48.65
<i>maximum</i> buffered play time [seconds]	92.32

<div>  Statistics </div> <div>Video: 23:03:41.413 - Unknown</div>	
Device-ID	765bda6758d33111
Mobile operator	Drillisch
Country	DE
Network switches	0
Network:WIFI	154.64
Screen size	1440x2560
Screen density	544 dpi
Orientation change	1
Orientation:Landscape	152.09
Orientation:Portrait	2.55
Player resizes	1
Size:424x239	2.91
Size:753x400	151.73
Handover	0
Cell-ID:5129242	154.64
Video-ID	t001J-M3g0U
Video title	Unkno...
Log time	342.49
Length of video	155.00
User engagement	154.64 / 99%
Initial delay	1.08
Quality switches	0
360p (640x360, 16:9)	154.64
Stalling	0
Total stalling length	0.00
Average stalling length	0.00
Maximal stalling length	0.00
Average buffer level	48.65
Maximal buffer level	92.32
Pause events	0
Video content rating	5
Video quality rating	2
Streaming rating	2
Acceptability rating	No

Statistics for scenario 2, device is connected to mobile access:

The images below shows YoMo statistics for video watched while the device was connected to mobile data access, taken from the images below-

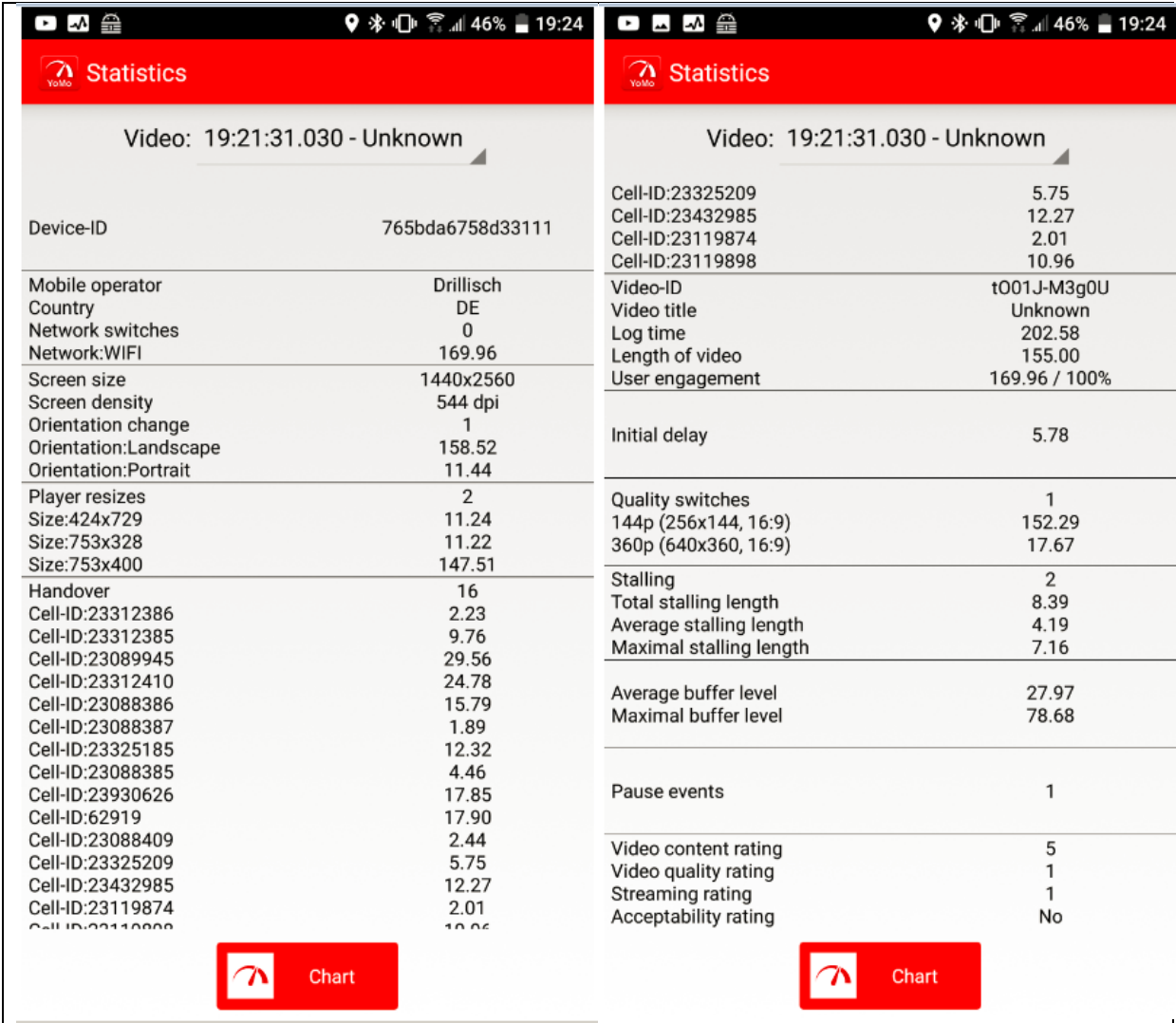
<u>Parameter</u>	<u>Value</u>
duration of the initial delay [seconds]	1.10
how often the video paused during play black	video stalled 0 times
the number of quality switches	0
average buffered play time [seconds]	50.57
maximum buffered play time [seconds]	91.48



Statistics for scenario 3, device is connected to Wi-Fi while traversing:

The images below shows YoMo statistics for video watched while the device was connected to Wi-Fi and traverse on high speed train, taken from the images below-

Parameter	Value
duration of the initial delay [seconds]	5.78
how often the video paused during play black	video stalled 2 times
the number of quality switches	1
average buffered play time [seconds]	27.97
maximum buffered play time [seconds]	78.68



Conclusion and analysis:

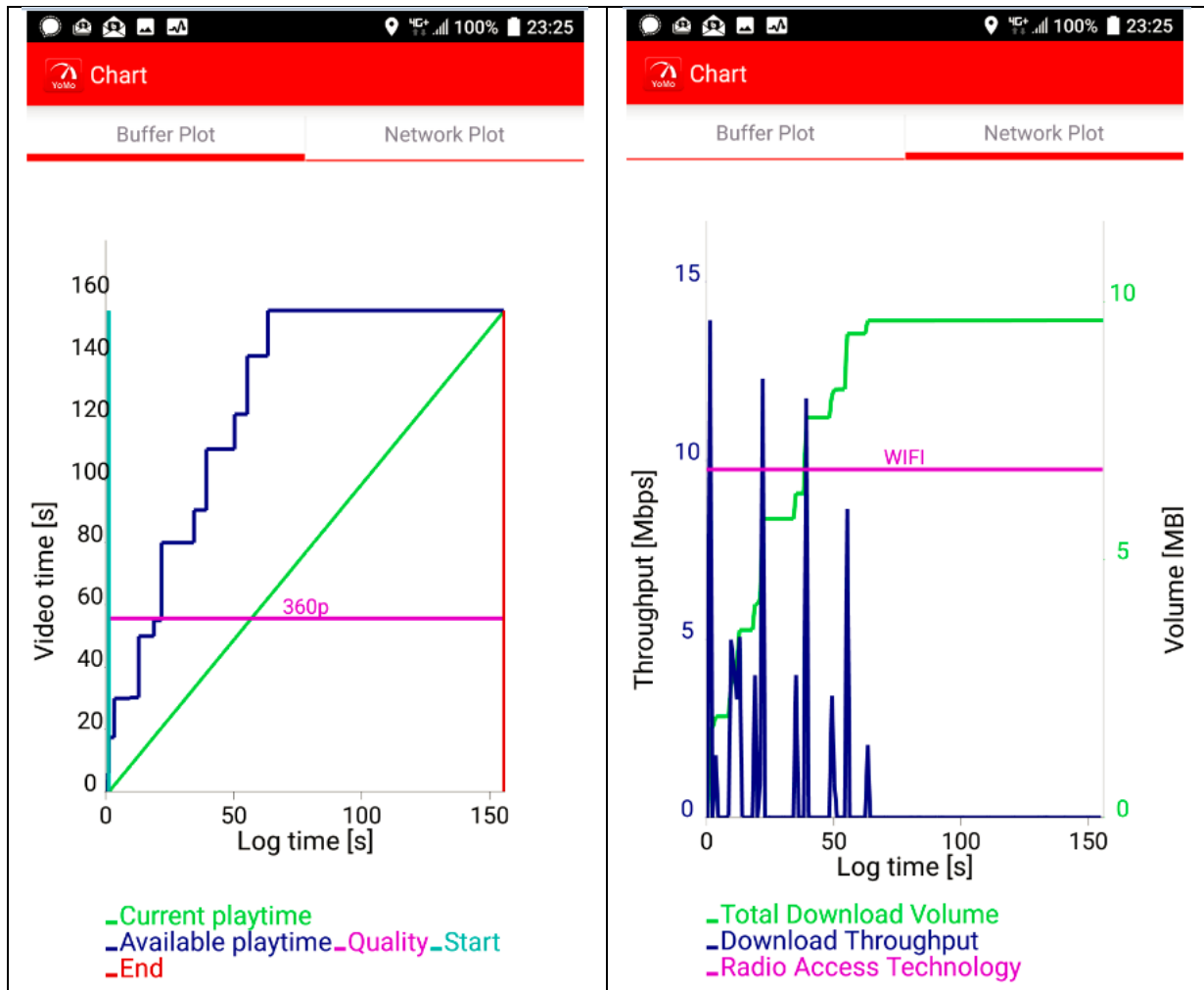
Parameter	Value		
	Wi-Fi	mobile data access	Wi-Fi + traverse
duration of the <i>initial delay</i> [s]	1.08	1.10	5.78
<i>video paused</i> during play black	video stalled 0 times	video stalled 0 times	video stalled 2 times
the number of <i>quality switches</i>	0	0	1
<i>average</i> buffered play time [s]	48.65	50.57	27.97
<i>maximum</i> buffered play time [s]	92.32	91.48	78.68

We can see that the different between Wi-Fi and mobile data access are neglectable. However, traversing does affect the video QoS: initial delay goes up, we getting a quality switch, and the available buffer decreeing.

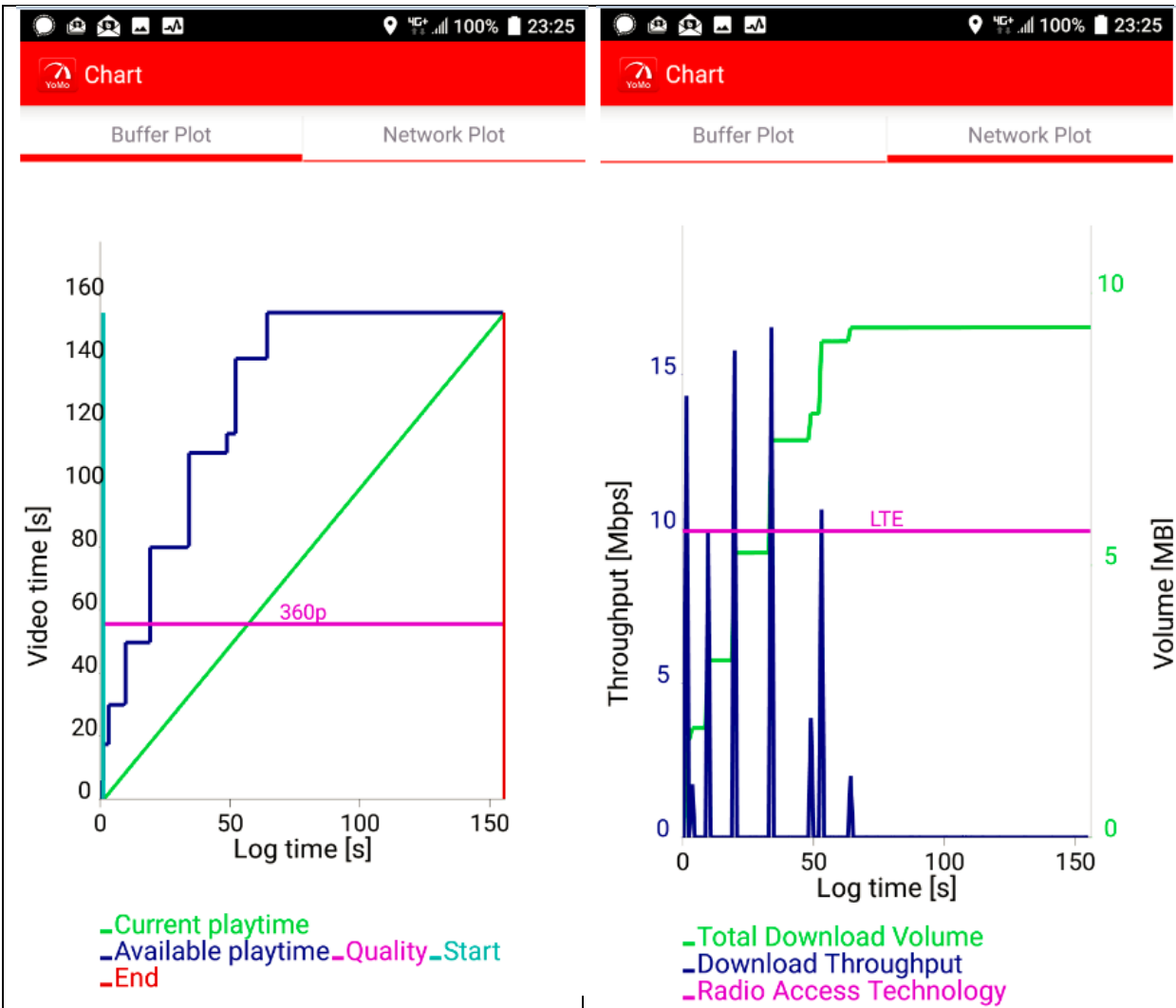
The reason for that could be because the train router, to which the device was connected, switched between base stations, operation which require handover procedure which is increasing the network delay.

Answer (b):

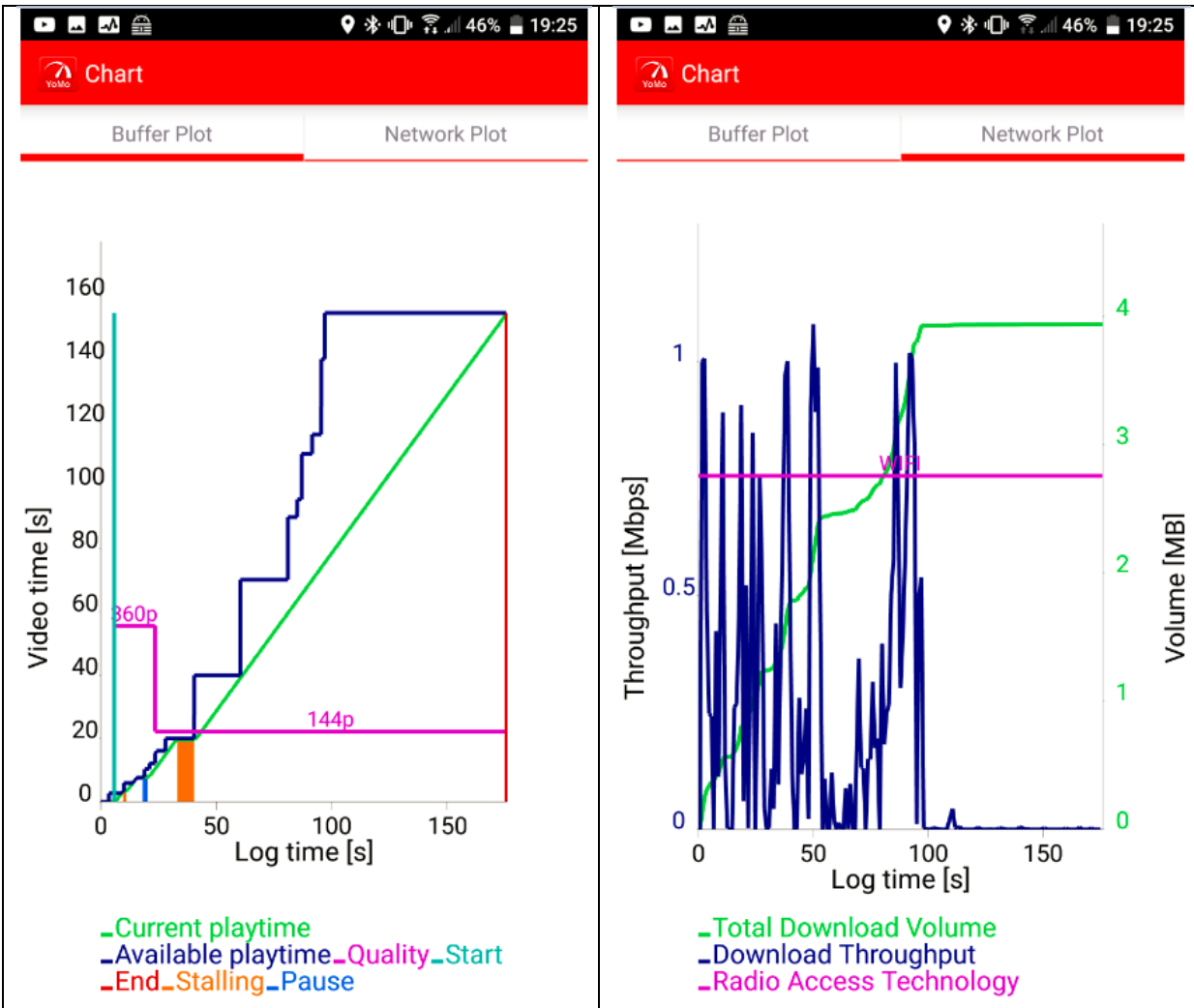
Illustrating buffer and throughput for scenario 1, device is connected to Wi-Fi:



Illustrating buffer and throughput for scenario 2, device is connected to mobile access:



Illustrating buffer and throughput for scenario 3, device is connected to Wi-Fi while traversing:



Conclusion and interpretation:

The *Wi-Fi* and *mobile data* access gives similar results for **buffering** and **play time**: the buffer is built up continually, in a way that make sure that there is always enough buffer in order to play the video without any delays.

However, there is a little different in how this is achieved: although both finish downloading the entire video in ~60 seconds, the mobile data access had picks of higher throughput than the Wi-Fi, and though there were a little different in the total downloaded video over time.

In contrast, the Wi-Fi while traversing suffered from very low throughput: 1 Mbps compare to 14 to 16 for the stationary Wi-Fi and the data mobile access, and so the buffer was building too slowly, causing a switch to lower quality around second 20, and 2 stall events. The first stall event happened while the video was playing in the higher quality, and the second happened while switching to the lower quality. After switching to lower quality, the buffer was blinding up continually without causing delays, and the whole video finish downloading around second 100.

Answer (c):

The subjective experience rating of the user is important, since no objective measurement can quantize the real experience the user had in reliable way every time. For every QoS metric, exists conditions where the metric gives good evaluation, even though the real experience is not so good, and vice versa.

The only one who can truly rate the user experience, is the user himself (giving that he gives his true opinion)

Answer (d):

The concept the YoMoApp used called “crowd sourcing”: using the crowd (- the public) in order to source **out** video quality tests.

The idea is to use the public in order to obtain information regarding video quality.

Benefits (for the YoMo way of crowdsourcing):

- potential for many evaluators
- low cost – no need to pay the ‘workers’
- information about the actual user experience (assuming honesty)

Drawbacks (for the YoMo way of crowdsourcing):

- it possible that non/very few will actually rate the video
- need to develop the test environment and advertise it
- users might cheat, or be careless while rating
- interested parties might give fake rating

Question 2: (35 Points) *Evaluating the Impact of Using Video Segments of Variable Duration*

For this task, you need the Big Buck Bunny full movie, provided here: <https://service.inet.tu-berlin.de/owncloud/index.php/s/h6DvYK8tmBTBQCi>.

Consider for all tasks variable bitrate encoding with a CRF of 30.

- (a) Use ffmpeg to encode and segment the video. Let the encoder freely choose the segment duration within a range of 0 to 10 seconds (e.g. segment 1 has a duration of 6 seconds, segment 2 has a duration of 3 seconds, etc.)

Answer: the conversion is done using FFMPEG and the following command:

```
ffmpeg -i ../images/big_buck_bunny_480p24.y4m \
-force_key_frames "expr:gte(t,n_forced*10)" \
-threads 1 \
-c:v h264 -profile:v main -crf:v 30 \
-f segment -segment_time 5.0 -segment_time_delta 5.0 \
-segment_list ../q2/a/big_buck_bunny.m3u8 \
../q2/a/big_buck_bunny%03d.ts
```

Flags used:

- `-threads 1` defines how many threads FFMPEG will use
- `-i` defines input source
- `-force_key_frames "expr:gte(t,n_forced * 10)"` defines positions where to enforce keyframe insertion using an expression. The current setting allows to force keyframe every 10 seconds.
- `-c:v h264 -profile:v main` defines the use of H.264 video codec with a main profile of H.264 codec
- `-crf:v 30` defines the crf quality setting of 30 for the video stream number
- `-f segment` enforces creation of output files according to the ffmpeg segmentation rules
- `-segment_time 5.0` defines the target segment length in seconds. Segment will be cut on the next keyframe after this time has passed unless you force a key frame
- `-segment_time_delta 5.0` allows to specify time boundaries within which a keyframe can be chosen (-5 seconds or +5 seconds)
- `-segment_list ../q2/a/big_buck_bunny.m3u8` defines playlist output filename and format
- `../q2/a/big_buck_bunny%03d.ts` defines video output path and format

- (b) Analyze:
the average segment duration
the standard deviation of segment duration
the segment durations along the video
the segment durations using a box plot

Answer: We obtain the following results:

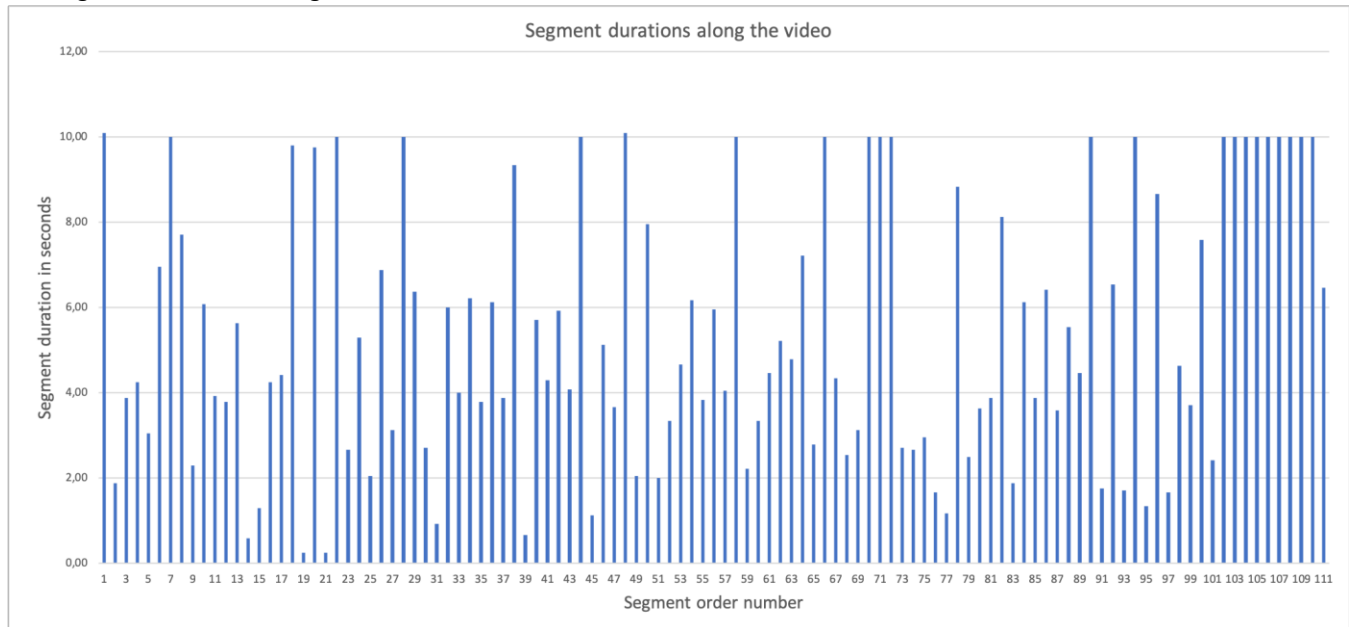
The average segment duration: 5.37 seconds

We assume that segment duration has normal distribution. In this case the average segment duration lies quite close to the expected value of normal distribution.

The standard deviation of segment duration: 3.09 seconds

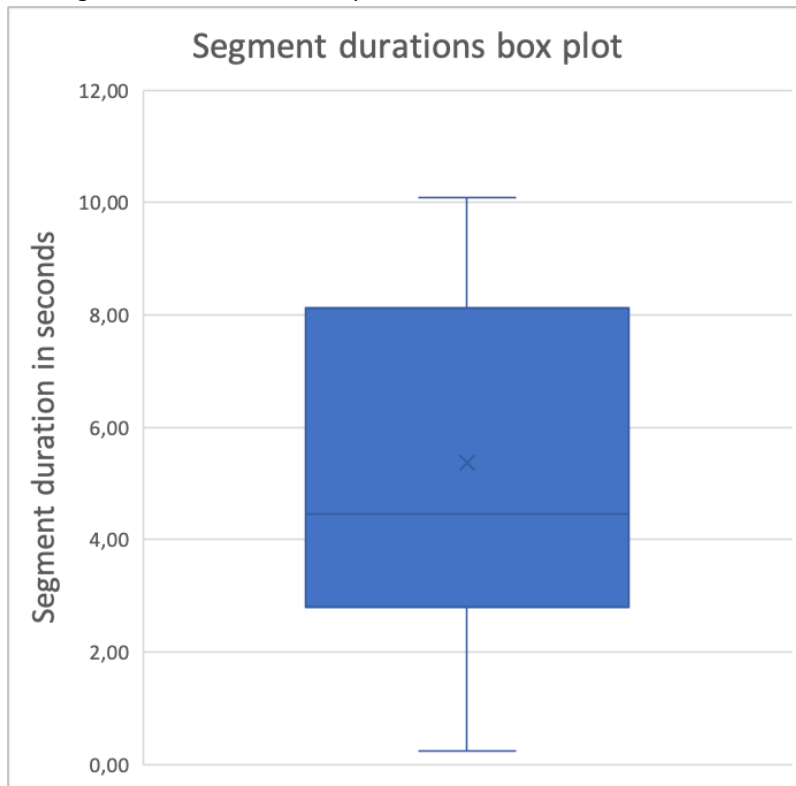
Standard deviation (or first sigma) shows boundaries where almost 70% of all values are expected to land in case of normal distribution.

The segment durations along the video:



As it seems from the “Segment durations along the video” plot, most of the values, as std. deviation states, are located within range 2,28 – 8,46 seconds. Another interesting moment is the sequence of 10 second segments in the very end of the video (102-110 segments). In the original video this sequence is represented by titles. There are no keyframe changes in this section, so the codec can choose the highest possible duration for these segments.

The segment durations in a box plot:



Box plot shows that 50% of values are expected to land within the blue box area (from approx. 3 to 8 seconds), median is at around 4,5 seconds and 99,3% of values are expected to land within 0,2 – 10,1 seconds area. Box plot makes it clear that normal distribution of segment duration does not have a very large peak in the middle of it, in contrary, it should look quite flat. It's clear from the observation that 50% of values are expected to land in around 3 to 8 seconds area. That means that the other 50% of values are expected to land outside of that area.

- (c) Encode and segment the video so to obtain the same average segment duration as resulting from subtask a). Thereby, you can floor or ceil to the nearest half second (e.g. 4.5s if your average from a) is 4.37s)

Answer:

The average segment size in task a) is 5.37. Two settings were tested with floor and ceiling values. Better, closer to original result was achieved for the segment duration value of 5.0 seconds. The average value achieved is 5.11 seconds. Conversion was done with ffmpeg and the following command:

```
ffmpeg -i ../images/big_buck_bunny_480p24.y4m \  
-threads 1 \  
-c:v h264 -profile:v main -crf:v 30 \  
-f segment -segment_time 5.0 \  
-segment_list ../q2/c/big_buck_bunny.m3u8 \  
../q2/c/big_buck_bunny%03d.ts
```

- (d) Have a closer on the frames' characteristics. For this task, only consider the fragmented videos resulting from the encoding with crf = 23.

1. How many I-frames in total are required for the different segmentations, i.e. seg dur = 0.5, 2, 4, 8, 12?

2. How much do the I-frames contribute to the overall file size in the different segmentations?

Answer:

Conversion was done using FFMPEG and the following command:

```
ffmpeg -i ../images/big_buck_bunny_480p24.y4m -threads 1 \
-force_key_frames "expr:gte(t,n_forced * $SEG_DUR)" \
-c:v h264 -profile:v main -crf:v 30 \
-f hls -hls_time $SEG_DUR -hls_flags independent_segments -hls_list_size 0 \
../q2/d/big_buck_bunny.m3u8
```

Flags used:

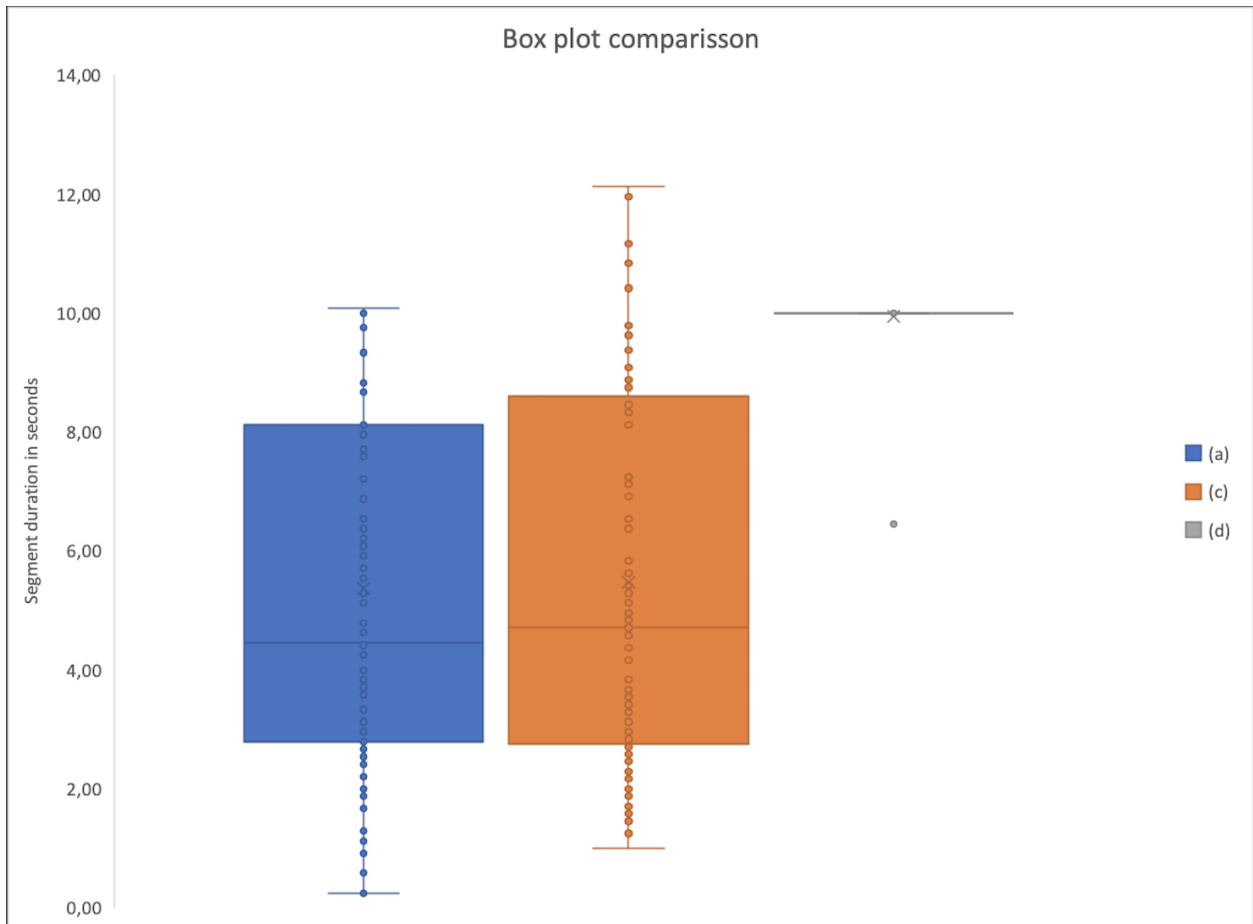
- `-threads 1` defines how many threads FFMPEG will use
- `-i` defines input source
- `-f hls` enforces creation of output file according to the HTTP Live Streaming (HLS) specification
- `-hls_time $SEG_DUR` defines the target segment length in seconds (`$SEG_DUR`). Segment will be cut on the next keyframe after this time has passed unless you force a key frame (see above)
- `-hls_flags independent_segments` adds a flag to playlist signaling that all the segments of that playlist are guaranteed to start with a keyframe
- `-hls_list_size 0` defines number of a playlist entries. 0 means keep all entries (by default 5, which saves only last 5 segments)

(e) Have a look on the average SSIM of the videos. Does it notably change among the different segmentations?

Answer:

We compare results by using the following table and box plot:

<u>Segment task</u>	<u>Number of I-frames</u>	<u>Total (fragmented) video size [MB]</u>	<u>Average segment size [KB]</u>
(a)	194	25,36	233,9
(c)	152	24,17	227,0
(d)	194	24,12	411,55



- (f) Considering your findings above, what would be possible benefits and drawbacks of using variable segment durations?

Answer:

Drawbacks:

If we compare (d) and (c) we can observe that, although (c) has around 20% less I-frames than (d), it weighs more. It can be caused by file headers overhead, as while in (d) we have 60 fragments, in (c) there are 109 of them, almost twice as more. The same goes for (a), while it has the same amount of I-frames as (d) it weighs more than (d), because of file header overhead.

As seen from (a) and (c), putting an overhead on

Benefits:

Smaller segments allow user a more fine-grained video control, as the average waiting time for segment load is smaller due to a smaller segment size. That means user can randomly jump in the video with less waiting time than he'd wait with a constant bigger segment duration.

Another benefit would be in reduced time between quality changes. Smaller segments allow faster quality adjust.

Question 3: (45 Points) Analyzing the Impact of Network Characteristics and Concurrent Flows on Video Streaming Performance

For this tasks, you need a (virtual) testbed consisting of server, a client, and the possibility to shape traffic. You can either setup your own environment or use the existing Vagrant machine from the previous sheet.

- (a) In order to emulate real world network characteristics, network traces are often used in research. Download the three provided network traces from <https://service.inet.tu-berlin.de/owncloud/index.php/s/TuiDICsuKTrwjEA>, <https://service.inet.tu-berlin.de/owncloud/index.php/s/hXmZsjG66QomZdc>, and <https://service.inet.tu-berlin.de/owncloud/index.php/s/WcnOjlcYFK5XkHV>. Implement, e.g. using a bash script, the network traces into your testbed. In case you use Linux namespaces, the following command is helpful:
`sudo ip netns exec ns_srv0 tc class change dev ns_srv0_veth0 parent 1: classid 1:1 htb rate 10000Kbit`
Each line in the trace file represents the throughput in kbps on a per-second scale.

Solution (a):

Please refer to: `code/q3_a-run_trace.sh` and `code/q3_a-get_statistics.sh`

- `code/q3_a-run_trace.sh` enable the user to set the network bandwidth astatically, or to start to set the network bandwidth in accordance to the network trace file.
- `code/q3_a-get_statistics.sh` shapes the network, and then start tapas player. It runs the provided playlist 5 times in each of the required configurations.

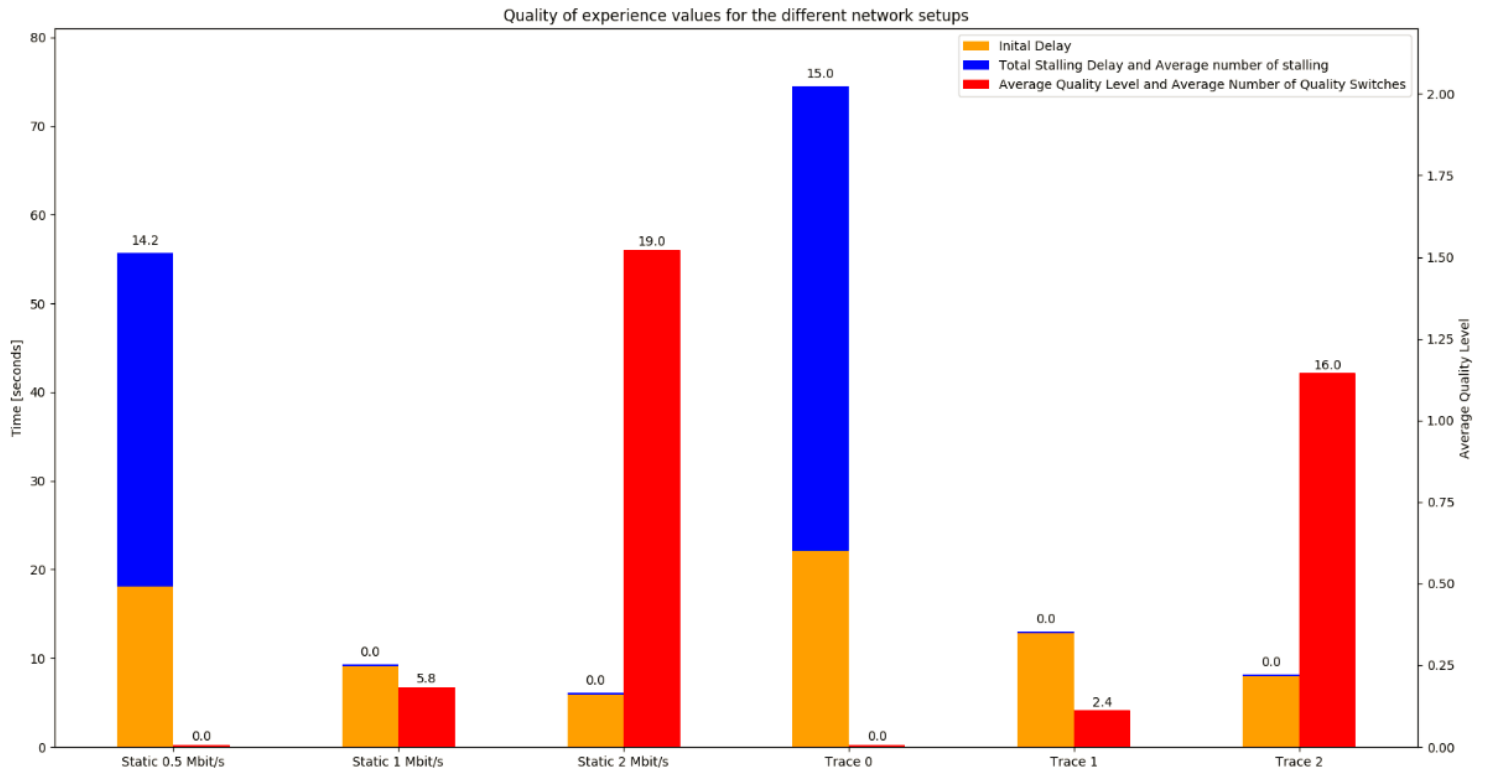
- (b) Download the provided videos <https://service.inet.tu-berlin.de/owncloud/index.php/s/orDS1ZclGdegAxT>. Perform at least five measurement runs for the following configurations:

- static bandwidth of 0.5 Mbps
- static bandwidth of 1 Mbps
- static bandwidth of 2 Mbps
- network trace0
- network trace1
- network trace2

Analyze the Quality of Experience-relevant metrics (initial delay, average quality, number of quality switches, stalling number and duration) for all configurations.

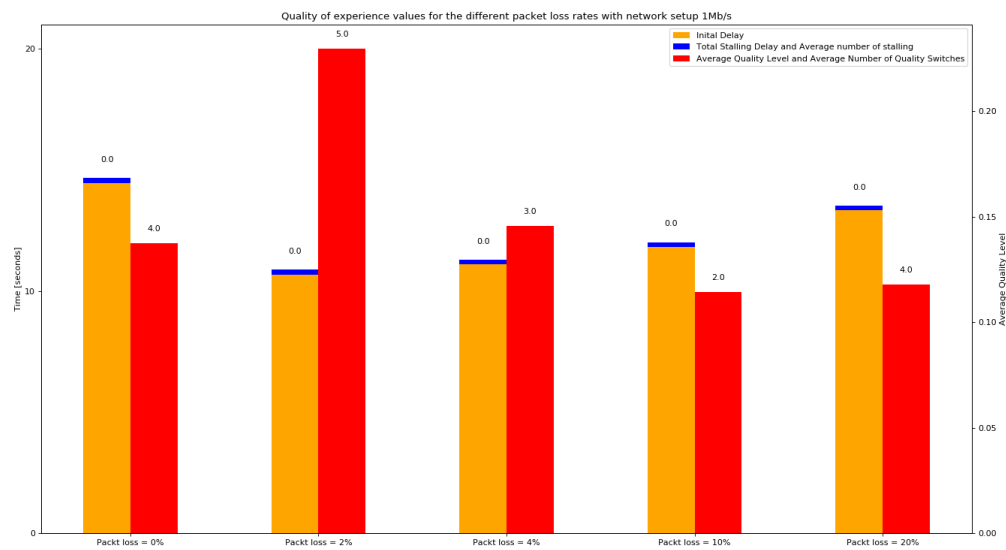
Solution (b):

- As can be seen in the figure below, higher bandwidth leads to better performance, and fixed bandwidth is preferable to varied bandwidth.
- Higher bandwidth → lower initial delay, less stalling, higher quality. Also, fixed bandwidth beats its equivalent varied bandwidth in all the categories.
- Due to the necessity to download the first segment before starting to play, there is always initial delay, but from certain bandwidth stalling is completely eliminated.
- The reason for the quality switches with the static 2Mbit/s network is the difference in the files size.



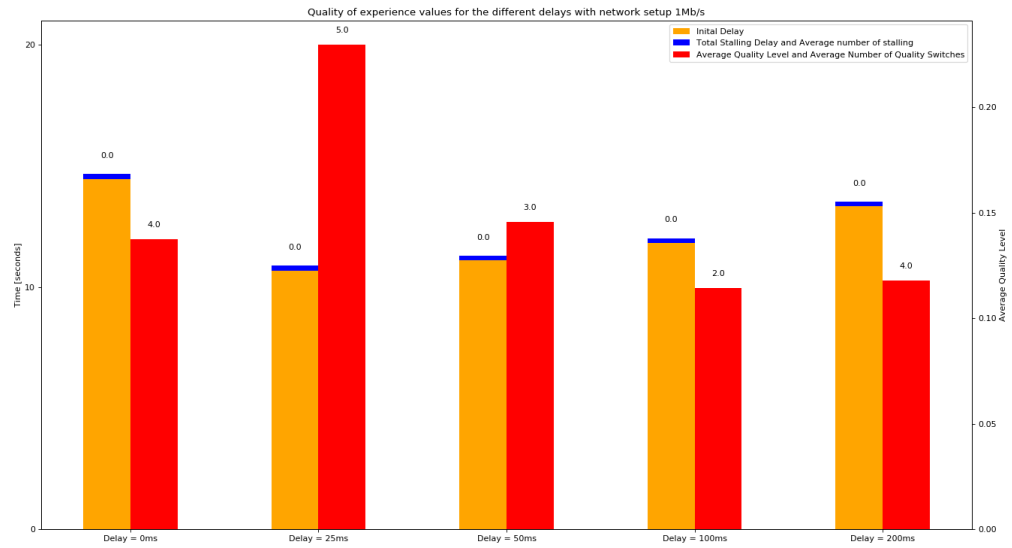
(c) Use network trace1 to evaluate the impact of packet loss values of $ppl = \{2; 4; 10; 20\} \%$ on the QoE-relevant metrics.

Solution (c):



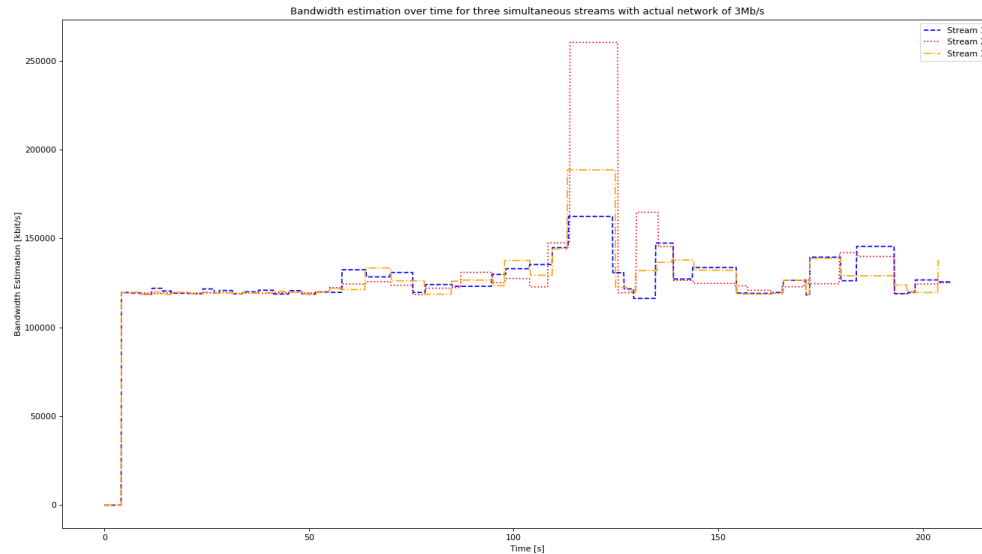
(d) Use network trace1 to evaluate the impact of delays of $delay = f25; 50; 100; 200\text{gms}$ on the QoE-relevant metrics.

Solution (d):



- (e) Set a bandwidth of 3 Mbps and start three video clients simultaneously. Plot the estimated throughput over time for all clients. What is the average estimated throughput for each client? Corresponds the result your expectations? Please explain!

Solution (e):



Question 4: (30 Points) *Design your own HAS Heuristic*

Solution (a):

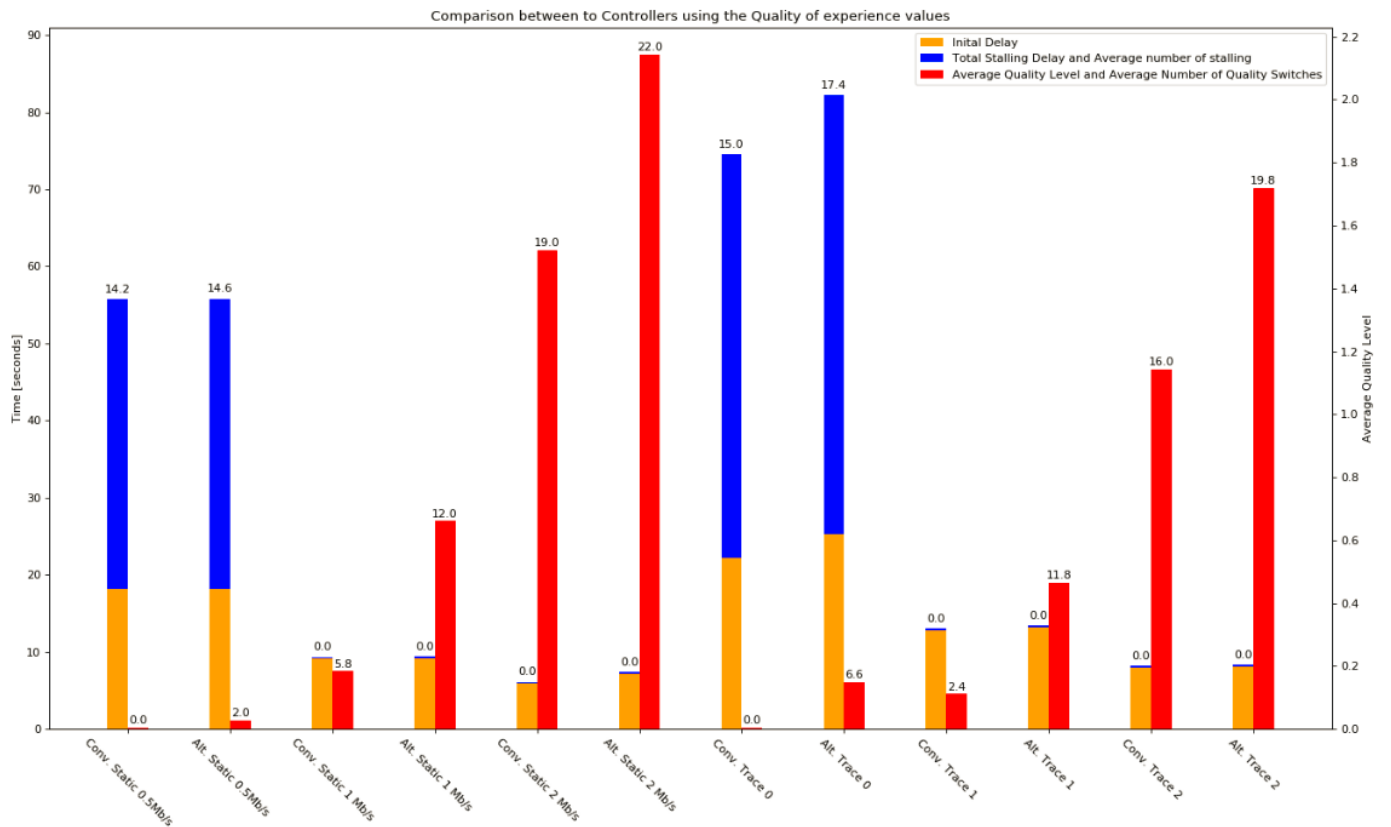
After logical solutions did not improve the performance, we concluded that the network traces are completely random, and that “smart” heuristics might actually result in worse performance.

Following this realization, we decided to try and remove the EWMA filter from the solution, using only the data from the last chunk downloaded.

As can be seen in the image below, this approach proved to work best, giving us some significant improvements:

- Initial delay – almost equal
- Average quality – we got significant improvement of the quality, in exchange to stalling.
- Number of video interruptions – the number of video interruptions went up

- Total duration of video interruptions – the duration of video interruptions went up



Solution (b):

ASAP – ‘As Simple As Possible’, Because sometimes simplicity wins over sophistication.