Linux client with WiFi and LTE
Project report

Pascal Maissen, Jovana Micic, Noe Wysshaar

University of Bern
pascal.maissen@unifr.ch, jovana.micic@students.unibe.ch,
noemathieu.wysshaar@unifr.ch

1 Protocol Introduction

The task of this project is study of Dynamic Adaptive Streaming over HTTP video delivery in a mobile scenario using WiFi and Long Term Evolution (LTE). Case scenario is that the user is connected through LTE, but periodically gets the internet access through WiFi. The main goal of this project is to evaluate the quality gain in the parallel LTE/WiFi video transmission in comparison to a single LTE transmission using multi-path TCP (MPTCP).

In order to achieve parallel connection of LTE and WiFi we have to use MPTCP. Multi-path TCP is a recent attempt to handle simultaneous use of multiple paths at the transport layer [1]. The core idea of MPTCP is to define a way to build a connection between two hosts and not between two interfaces (as standard TCP does). MPTCP represents the evolution of TCP protocol and it works in all networks where TCP works. Benefits of MPTCP include better resource utilization, better throughput and smoother reaction to failures. Using MPTCP dramatically improves Quality of Experience of video streaming.

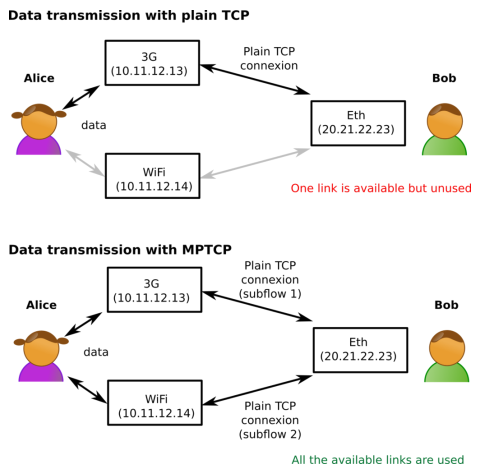

Fig. 1. Difference between TCP and MPTCP.

In the figure 1 is presented graphic explaining the difference between TCP and MPTCP. For instance, Alice has a smartphone with 3G and WiFi interfaces and Bob has a computer with an Ethernet interface. In standard TCP, the connection should be established between two IP addresses. An application can only create one TCP connection through a single link. MPTCP allows the connection to use several paths simultaneously. For this, MPTCP creates one TCP connection, subflow, over each path that needs to be used. So in the case from the figure 1, Alice can use both available links, 3G and WiFi at the same time.

In our project description it was required to use Dynamic Adaptive Streaming over HTTP (DASH). DASH is a streaming technique that enables high quality streaming of media content over the Internet [2]. DASH works by breaking the content into a sequence of small HTTP-based file segments, each segment containing a short interval of playback time of content that is potentially many hours in duration, such as a movie or the live broadcast of a sports event. The content is made available at a variety of different bit rates, i.e., alternative segments encoded at different bit rates covering aligned short intervals of playback time. DASH client can adapt to changing network conditions and provide high quality playback with fewer stalls or re-buffering events. DASH should not be confused with a transport protocol, the transport protocol that DASH uses is TCP.

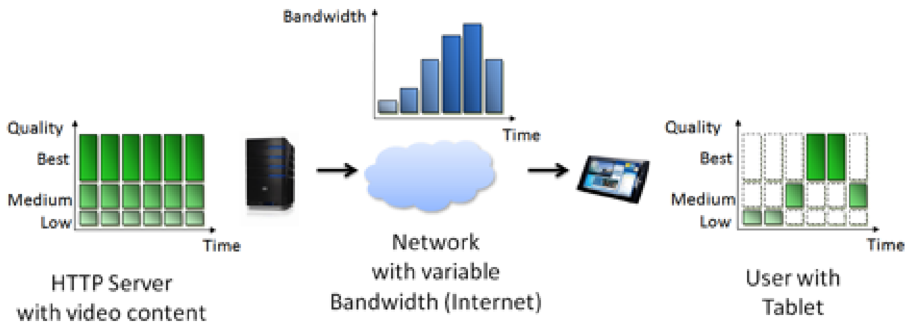

Fig. 2. Concept of Dynamic Adaptive Streaming over HTTP.

In the figure 2 is presented concept of DASH. This concept is consisting of the three parts. First part is HTTP server with video content of different qualities. Second part is network which has varying bandwidth conditions. Third part is user with device which selects appropriate segments for each timepoint. So the main advantage of using DASH is that is flexible, it dynamically adapts to current network conditions and it reuses existing Internet architecture.

2 Experimental setup

For MPTCP to work we needed to install the MPTCP kernel [3]. We downloaded, compiled and installed the kernel on two linux machines. One later acting as client and the other one as server. The setup is illustrated on figure 3.

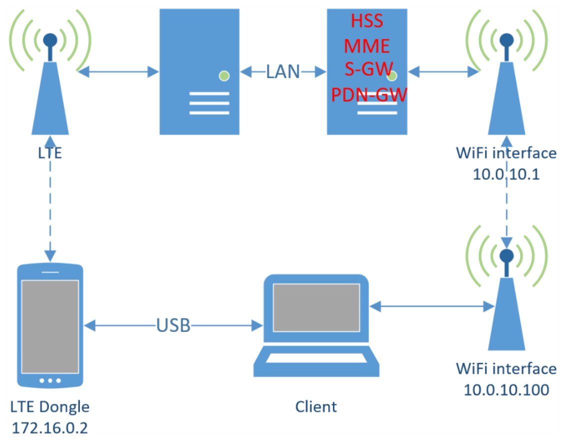

Fig. 3. Hardware setup used for the experiments.

Both computers needed at least two network interfaces. The client was equipped with a wireless network card for the WiFi connection and for the connection over LTE we used a USB dongle. On the serverside a custom WiFi access point was created with the help of hostapd [1]. to which the client connected. In addition, the server needs to be connected with an LTE antenna. To our luck this computer was already set up for this task and had all necessary LTE components such as the Home Subscriber Server (HSS), Mobility Management Entity (MME), Serving Gateway (S-GW) and PDN Gateway (P-GW). Among some minor difuculties regarding the establishment of the LTE connection (presumably due to a poor LTE dongle) everything was set up and ready for testing.

In the first part of our experiment we just tested throughput of the data without video streaming. We had two different scenarios:

1. LTE and WiFi working in parallel. WiFi was turned off after 10 seconds and then turned on after 10 seconds.
2. WiFi was turned off and LTE was on. WiFi was turned on after 10 seconds.

In the second part of our experiment we wanted to test data throughput in a scenario when user is watching a video content from the Internet. So the

[1] hostapd `http://w1.fi/hostapd/`

first step was to find appropriate video which we will stream over the Internet. Appropriate video means that bitrate of the video has to be larger than 25 Mbps because of the restulst we got in the first experiment. We choose *4K Ultra HD Firework* video[2]. In the table 1 is presented general information about video.

Attribute	Value
Format	MPEG-4
File size	1.19 GiB
Duration	1mn 47s
Overall bitrate	95.7 Mbps
Width	3 840 pixels
Height	2 160 pixels
Writing library	x264 core 114

Table 1. General information about video.

The next step was to prepare video for DASH streaming. We encoded a video in different qualities to provide multiple representations according to the DASH standard. For transcoding the video to the H.264 standard we used x264 library[3]. This library is a free software library for encoding video streams into the H.264/MPEG-4 AVC compression format. Example of the shell script used for creating video of 5 Mbps bitrate is presented in the following listing.

```
bitrate=5000

x264 --output intermediate_${bitrate}k.264 --fps 60 --preset
slow --bitrate $((bitrate)) --vbv-maxrate $((bitrate*2))
--vbv-bufsize $((bitrate*2*2)) --min-keyint 120 --keyint 120
--scenecut 0 --no-scenecut --pass 1
--video-filter "resize:width=1280,height=720" firework.mp4

MP4Box -add intermediate_${bitrate}k.264 -fps 60
out_${bitrate}k.mp4 rm intermediate_${bitrate}k.264
```

Listing 1.1. Shell script for encoding the video.

Output of this script is a raw H.264 video which data is encapsulated in a mp4 container. MPEG-4 Part 14 or mp4 is a digital multimedia container format most commonly used to store video and audio. Using script presented in the listing, we created different representations. Description of all representations is presented in the table 2.

We choose this range of bitrates because we want to be sure that we will use different representations during the streaming depending on that if we use only LTE or WiFi and LTE in parallel. After we created all representations, the next

[2] Firework video `http://4ksamples.com/4k-uhd-fireworks-sample/`
[3] x264 `https://www.videolan.org/developers/x264.html`

Representation ID	Resolution [px x px]	Bitrate [Mbps]
1	1280 x 720	5
2	1280 x 720	10
3	1920 x 1080	15
4	1920 x 1080	20
5	1920 x 1080	25

Table 2. The different representations of the encoded video.

step was to "dashify" the content so it is appropriate for the DASH protocol. For that purpose we used MP4Box[4], multimedia packager available in GPAC. It is used for performing manipulations on multimedia files like AVI, MPG, TS, but mostly on ISO media files (e.g. MP4, 3GP). In the following listing is presented a command used for "dashifying" content.

```
MP4Box -dash ts -frag ts -rap -out DASHFILE.mpd
-profile live VIDEO_QUALITY1.mp4 VIDEO_QUALITY2.mp4 ...
```

Listing 1.2. Command for dashyfing content using MP4Box.

The most important file of previous process is *DASHFILE.mpd*, created manifest, which is needed for the stream itself. Media presentation description (MPD) describes segment information. It is an XML document describing where the various media resources present in the content are located. The segment is a continuous part of a media resource. The segment is the smallest part of the media that can be located in the MPD. All produced content, including the MPD manifest, we store on the university Apache server. After that we were able to run the experiment and measure data.

For playing the video, we used MP4Client[5]. It is provided by GPAC[6] and it is a highly configurable multimedia player. It can be used from a command-line, as a GUI or browser plugin. In our experiment we used it from command-line, running the command presented in the following listing.

```
MP4Client <server_address>/DASHFILE.mpd
```

Listing 1.3. Command for running MP4Client player.

We used Wireshark for capturing the data. In the second experiment, we had only one scenario where LTE and WiFi were active from the beginning. After 20 seconds, we turned off WiFi and after 20 seconds, turned on WiFi.

[4] MP4Box `https://gpac.wp.imt.fr/mp4box/`
[5] MP4Client `https://gpac.wp.imt.fr/player/`
[6] GPAC `https://gpac.wp.imt.fr/`

3 Results and Analsys

As it was mentioned in the previous section, for the first part of the project we measure data throughput for the two case scenarios. In the following text we will present results we got from this experiments.

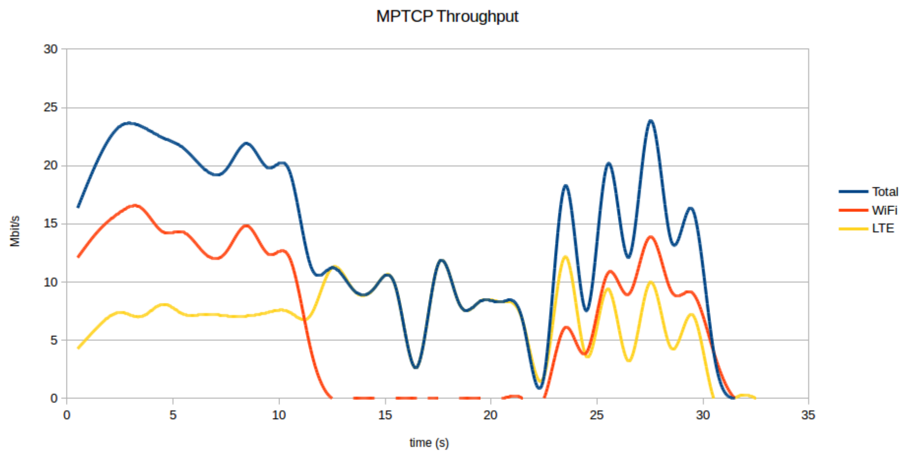

Fig. 4. Graphic representing the MPTCP throughput when WiFi is turned off after 10 sec.

In the figure 4 is presented throughput in the scenario when we have in parallel WiFi and LTE from the beginning of the experiment. After 10 seconds, we turned off WiFi. As we can see from the graphic, LTE continues to work but total throughput becomes lower and equal as LTE throughput. After 10 seconds, we had again turned WiFi on and it can be seen clearly on the figure that we have some WiFi throughput after 20 seconds. The maximal total throughput achieved in this scenario is around 24 Mbps.

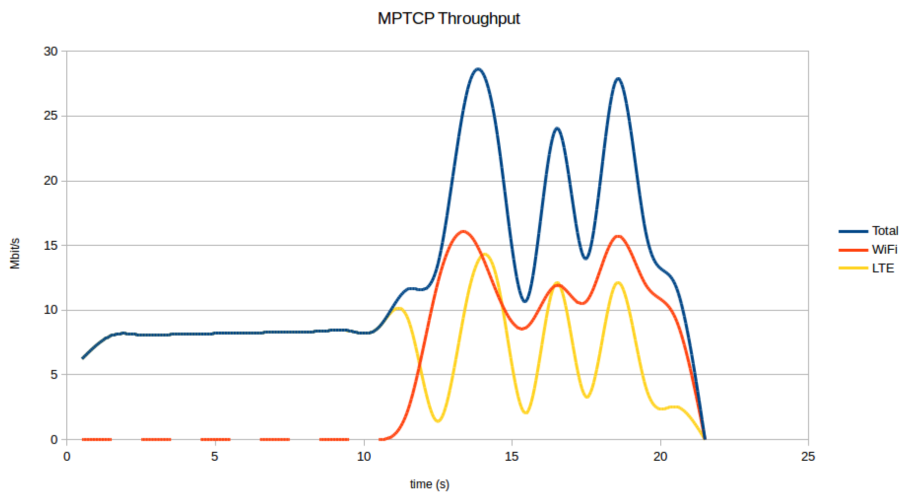

Fig. 5. Graphic representing the MPTCP throughput when WiFi is turned on after 10 sec.

In the figure 5 is presented throughput in the scenario when only LTE is turned on from the beginning of the experiment. After 10 seconds, WiFi is turned on. From the graphic we can see that there is increase of total throughput when WiFi is on. The maximal total throughput achieved in this scenario is around 28 Mbps.

The interesting observation from both graphics is that in the moment when the WiFi is again turned on, we have decrease of the LTE throughput. Reason for this could be specific MPTCP protocol implementation but we can not claim this with great certainty.

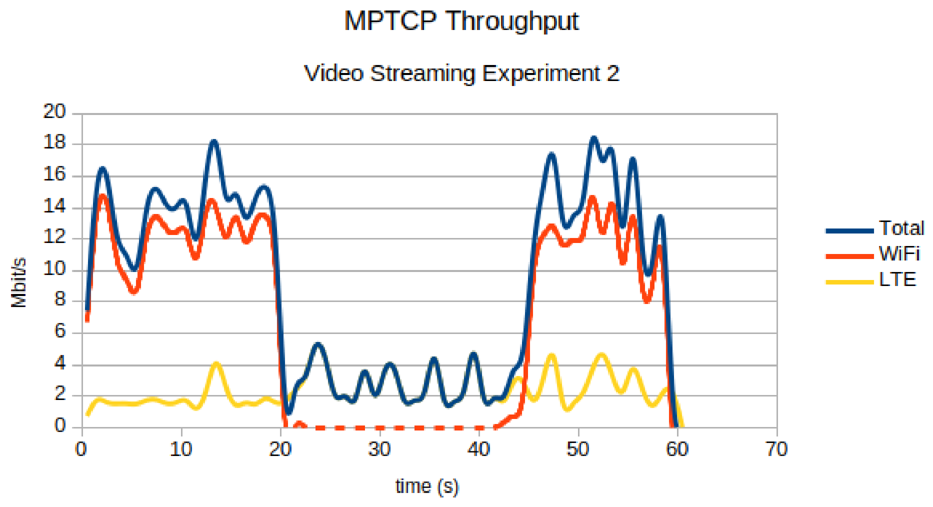

Fig. 6. Throughput of MPTCP over LTE and WiFi.

Figure 6 shows the throughput of both network interfaces during a DASH video streaming test. At the start both interface are active and after 20 seconds WiFi was turned off (to simulate a signal loss) and around 25 seconds later on it was turned back on again. The graph shows a throughput around a relatively constant 3 Mbps. For the WiFi interface the throughput is around 12 Mbps while it is active. Overall this sums up to a total throughput around 15 Mbps.

In the following is presented a graphic showing the change of the buffer size during the video streaming. In the figure 7 we can see that at the beginning we are using the lowest representation but after the few seconds we came to the representation 3 because we have WiFi and LTE working in parallel. Significant drop on the graphic is because it is a moment when WiFi was turned off, so it has to adjust to new network conditions and continue playing the lowest representation. After 20 seconds, we again turned on WiFi and we can see that higher representations are used again.

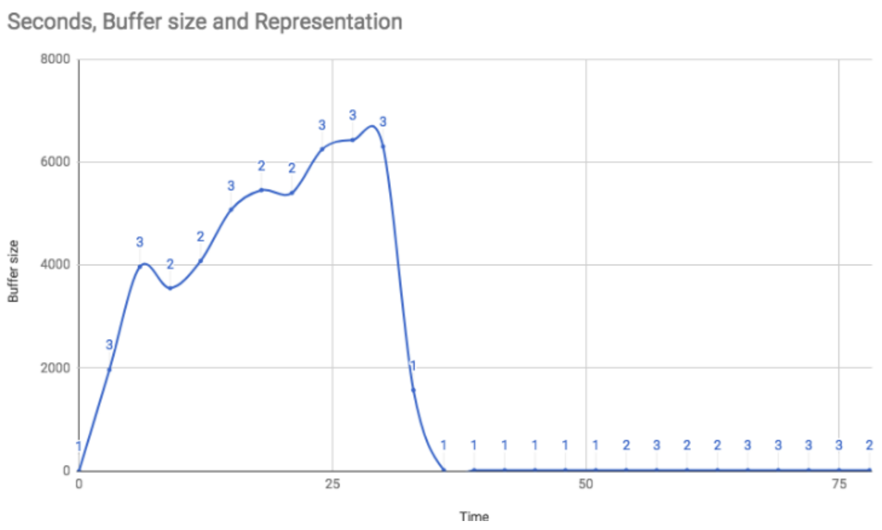

Fig. 7. Graphic representing the change of buffer size during the experiment.

4 Conclusions

The two main problems of the streaming video content are quality and buffering. For example, video that is only 1280 x 720 will never play at correct quality levels on a screen that is 1920 x 1080 px, it will be stretched. Second thing is buffering. If the users has a poor quality internet connection, and cannot download the video stream quickly enough, then the video will need to pause, wait for more data, and then start again. This makes watching a video horrible for the user. Solution for this problem is adaptive streaming.

Adaptive streaming allowed us to create a different video representations so we can stream a specific video file ensuring that the viewer always receives a video that will look good. Because adaptive streaming adapts to the speed of the users internet connection, we avoided pausing the video because of the buffering.

In our experiment we came to the conclusion that using WiFi when is available, improves the quality of experience very much. It allows user to watch video in higher resolution without pausing and waiting to buffer more data.

In addition, we would like to thank Dr. Eryk Schiller for helping us with setting everything up.

References

1. Barr, Sbastien. *"Implementation and assessment of modern host-based multipath solutions."* Universite catholique de Louvain, Louvain-la-Neuve, Belgium (2011).
2. Michalos, M. G., S. P. Kessanidis, and S. L. Nalmpantis. *"Dynamic Adaptive Streaming over HTTP."* Journal of Engineering Science & Technology Review 5.2 (2012).

3. C. Paasch, S. Barre, et al.. *Multipath TCP in the Linux Kernel.* Available from `http://www.multipath-tcp.org`.
4. Stefan Lederer, Christopher Mller and Christian Timmerer, *Dynamic Adaptive Streaming over HTTP Dataset*, In Proceedings of the ACM Multimedia Systems Conference 2012, Chapel Hill, North Carolina, February 22-24 (2012).