Dynamic Adaptive Video Streaming on Heterogeneous TVWS and Wi-Fi networks

Luca Bedogni*, Angelo Trotta*, Marco Di Felice*, Yue Gao‡, Xingjian Zhang‡,
Qianyun Zhang‡, Fabio Malabocchia †, Luciano Bononi*
*Department of Computer Science and Engineering, University of Bologna, Italy
†Telecom Italia SpA, Turin, Italy, ‡Queen Mary University, London, UK
*Email: {lbedogni, trotta, difelice, bononi}@cs.unibo.it †, fabio.malabocchia@telecomitalia.it
‡ {yue.gao, xingjian.zhang, qianyun.zhang}@qmul.ac.uk

***Abstract*—Nowadays, people usually connect to the internet through a multitude of different devices. Video streaming takes the lion's share of the bandwidth, and represents the real challenge for the service providers and for the research community in general. At the same time, most of the connections come from indoor environments, where Wi-Fi already experiences congestion and coverage holes, which directly translate into a poor experience for the user. A possible relief comes from TV White Space (TVWS) networks, which can enhance the communication range thanks to Sub-GHz frequencies and favorable propagation characteristics, but offer slower datarates compared to other 802.11 protocols. In this paper, we show the benefits that a TVWS network can bring to the end user, and we present CABA, a Connection Aware Balancing Algorithm able to exploit multiple radio connections in favor of a better user experience. Our experimental results indicate that the TVWS network can effectively provide a wider communication range, but a load balancing middleware between the available connections on the device must be used to achieve better performance. We conclude our study by presenting real data coming from field trials in which we streamed an MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) video over TVWS and Wi-Fi. Practical quantitative results on the achievable Quality of Experience for the end user are then reported. Our results show that balancing the load between WIFI and TVWS can provide a higher playback quality (up to 15% of average quality index) in scenarios in which the WIFI is received at a low strength.**

I. Introduction

With the skyrocketing growth of mobile devices, which account for a relentless increase of bandwidth, it is mandatory to find new paradigms to support new and existing services. Often, mobile devices connect to the Internet via a cellular network like 2G, 3G or, more recently, Long Term Evolution (LTE). However, when indoors, devices often switch to WiFi, mainly because it can provide more bandwidth and costs less to the end user [1]. However, the WiFi signal is heavily affected by the distance to the Access Point (AP), and by obstacles, thus making the reception hard in some areas. In fact, a vast amount of users prefer using LTE instead of WiFi while at home, due to the aforementioned problems [2], even when this means a higher cost.

At the same time, the great majority of users requested content is video [3], hence it is even more challenging to deal with the strict timing limits it imposes. In fact it is stated in [3] that *"Mobile video traffic exceeded 50 percent of total mobile data traffic for the first time in 2012"*. Mobile data connections struggle to keep up with satisfying datarates, as [3] reports an average connection speed of 1,684 kbps in 2014 against an average of 1,387 kbps in 2013, and 46 % of the total mobile traffic was offloaded onto a fixed network through WiFi offloading or small cells [3]. With these numbers in mind, it is straightforward to note that mobile video traffic is expected to grow even more in the future, and that mobile connections are struggling to keep the same pace.

A possible solution to extend indoor coverage is to exploit TV White Space (TVWS) networks, which thanks to Sub-GHz communication frequencies and favorable propagation characteristics can extend the communication range and eventually cover a larger area. At the same time, they present slower speeds per Hz compared to WiFi networks. In this work, we will focus on TVWS as offloading technology, as it presents more similarities at the infrastructure level compared to, for instance, LTE.

To cope with a rapidly changing wireless environment, recently many protocols, either standard or proprietary, have emerged for Dynamic Video streaming, like MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [4], Apple HTTP Live Streaming (HLS), Microsoft Smooth Streaming, and Adobe HTTP Dynamic Streaming (ADS). The idea is to split a single video in multiple segments of a pre-defined length and at different bitrates. Then, a manifest for each multimedia content is created and published, and the clients asking for that video have the possibility to know and choose the possible resolutions and coding qualities. The client can play an active role in the rate adaptation algorithm and decide what to privilege, e.g. the video quality, by trying to download higher quality segments, or the video flow, thus reducing quality in favor of a smoother playback in presence of unstable network conditions.

In this paper, we focus on the MPEG-DASH protocol, but the proposed solution is independent on the application protocol used. We use a standard MPEG-DASH client, and center our analysis on the lower layers of the network stack. A complete review on the MPEG-DASH protocol is outside the scope of this paper, and thus we refer the interested readers to [5]. We remark the fact that our solution is completely transparent to any HTTP based protocol, and to any client

that implements it. Thus, it can be used without any change to already developed clients.

We leverage on the availability of multiple wireless connections on a device, and provide three novel contributions: *i)* an experimental study on MPEG-DASH performance with different Radio Access Technologies (RAT) *ii)* CABA, a Connection Aware Balancing Algorithm that estimates the networks conditions, and routes the traffic accordingly, *iii)* performance evaluation of the proposed solution, along with a testbed evaluated with real TVWS and WiFi networks. To the best of our knowledge, this is the first work that presents video streaming results on TVWS and WiFi networks.

Selecting the network directly at the application layer has several advantages for video streaming, as it access crucial information from the video streaming client. Basically, at the application layer a number of information such as the precise quality at which the video is streamed, the buffer level, and an history of previous problem with the current video are available. For instance, at the application layer we have the possibility to choose more stable connections if the buffer is low, hence exploiting robustness over speed. It also empowers the rate adaptation algorithm with additional information about the lower layers. If aggregation at the lower layer is in place, then the rate adaptation algorithms only have the ability to choose the quality of the connection based on the observed throughput. However, this might vary dramatically, depending on which connection is used. Therefore, the video client might be dazzled by it, by seeing different datarates at the application layer, which might be simply related to different connections used.

The cost of doing interface switching at the application layer is not related to the multimedia video streaming application itself, but rather to other applications and services running on the same host. Basically, lower layer systems such as MULTIPATH can better balance the different network connections of the host, as it has a global view of the system, and can thus achieve also a better fairness among applications. However, since our focus is on video streaming, which when in play is typically the most important application running on the host for the user, doing the interface switching at the application layer has the advantage of better selecting the interface for the multimedia application itself, possibly reducing the performance of other applications.

Results on a synthetic dataset showed that CABA is able to provide the best performance in all the 4 scenario considered, regardless of the performance of the WiFi connection or the TVWS.

Moreover, on a real deployment which we set up for the purpose of this study, CABA achieved a better buffer level in 3 out of the 6 scenario tested, and the same one in the other 3, with the same video quality of other connections alone, and a higher one in 1 scenario. Moreover, the TVWS connection alone achieved a good video quality streamed in 6 tested locations in which the WiFi connection was not received at all, therefore highlighting the benefits of a lower data rate connection with better transmitting range.

The rest of this paper is as follows: Section II presents related work on Dynamic Adaptive Streaming protocol and studies; the motivation behind this work is given in Section III-A; we then present our algorithm in Section IV, and evaluate it in Section V. Section VI reviews the results obtained and concludes the paper.

II. Related Work

The spectrum scarcity, along with the need to allocate new services, is the focus of several works and proposals to enhance the spectrum utilization and eventually provide a better quality of experience to the final users. Several works propose the use of cognitive wireless network techniques, in order to achieve a better spectrum utilization by leveraging on the temporal utilization of the spectrum [6].

TVWS network foresee the usage of a remote spectrum database, in charge of providing the list of available channels at the device location [7]. Another possibility is to sense the channel before transmitting data, called spectrum sensing [8]. Studies vary, but in general there is abundance of TVWS channels, particularly in rural areas. However, also in urban areas it is possible to find plenty of TVWS, thanks to the transition to digital TV [9].

In particular, the authors in [10] propose a distributed spectrum sharing mechanism based on cognitive networks techniques, focused on video streaming. They define different queues, in which they place the packets to be transmitted with different probabilities, based on the packet's importance for the playback. If the packet is an I-Frame, it gets higher priority compared to a P-Frame, which only carries the differences between two subsequent images, rather than the whole image as the I-Frame does. A similar approach is taken in [11], where the authors still leverage on the difference between the frames in order to maximize the number of I-Frames to be delivered, which might freeze the video if lost.

To overcome the poor quality of real-time data streaming, like voice-over-IP (VoIP), over a single WiFi connection in [12] is proposed a system that replicates the data stream on more then one connection link exploiting the diversity of multiple WiFi links. Furthermore, the idea to use more than one wireless technology, to possibly provide more bandwidth and/or increase the reliability of the connection, is an active topic and has already been investigate in literature [13]. Regarding multiple RAT technologies, it is possible to find a plethora of different proposal, ranging from offloading to TV White Space [14] [15] or [16], to content centric networking [17] [18], to cooperative streaming through smartphones [19] [20]. It is also important to note the work in [21], where the authors study different antenna patterns regarding TV White Space. The rationale is that most of the proposals try to forecast the network conditions, and select either the RAT that has the higher performance, or the one that is more reliable, depending on the application requirements. At system level, Multipath TCP (MPTCP) is the standard solution when using multi homed devices [22]. Basically, multiple subflows for every connection can be created, and the traffic is is spread

over those subflows according to the MPTCP scheduler. This system has been studied in multi-homing scenarios with WiFi and cellular connections [23]. In this case, only on particular scenarios, i.e. when the two used connections have similar characteristics, the MPTCP method gives good performance [23]. More recently, also specialized schedulers for MPEG-DASH have been proposed, such as the one in [24]. The main difference between [24] and the work presented in this paper is that CABA does not require any modification to the remote server, as [24] does. It also targets cellular and WiFi connections, whilst our study is more focused on in-home, AP-like networks. Moreover, doing the switching at the application level, directly in the video streaming client, offers the possibility to monitor the state of the buffer and take decisions accordingly.

In the recent years, several dynamic and adaptive video streaming protocols have been proposed and effectively utilized, ranging from Apple HLS, to Microsoft Smooth Streaming, to MPEG-DASH. The basic idea is to divide the video to be streamed into different segments at different resolutions. A manifest file is then built to report all the available resolutions of the video to be streamed. To play a content, clients need first to access the manifest, and select an appropriate video quality according to the network conditions, that can change over time. Hence, clients can exploit a rate adaptation algorithm, that has to monitor the network conditions and throughput performance, and adapt the playback to the network behavior [5]. MPEG-DASH based streaming has been evaluated also in the context of vehicular networks [25], where network conditions change rapidly and can disrupt the playback.

Most of the research work on MPEG-DASH relies on the study of rate adaptation algorithms that can provide a better Quality of Experience to the end user, by switching to the most suitable video quality given the network conditions. These works include [26], where the authors present a rate adaptation algorithm that is also studied for 3D videos and [27] where the authors propose a rate adaptation algorithm with fixed-interval buffer. Basically, they keep the same resolution until quality remains within a certain threshold, and switch up or down depending on the network conditions. FDASH, a rate adaptation algorithm based on fuzzy logic, is presented and evaluated in [28].

In this work, we use a publicly available MPEG-DASH web client developed by Akamai [1], and thus we do not focus on a specific rate adaptation algorithm.

III. Motivation and System Model

The motivation of this work, presented in this section, is to show results from a preliminary study we performed to evaluate the effects of TVWS and IEEE 802.11n WiFi networks in the same environment, which is presented in Section III-A. Section III-B details the proposed system model.

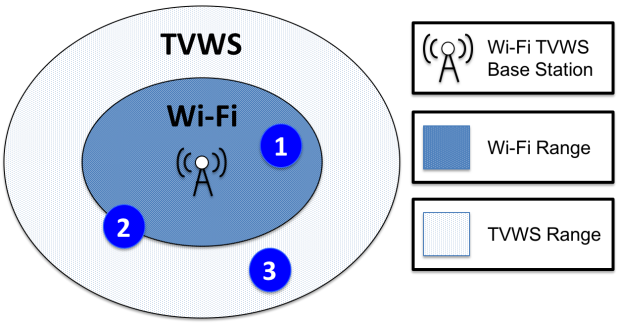


Fig. 1. The studied scenario.

A. Motivation

Figure 1 shows the studied scenario, in which we have a base station (BS) with both TVWS and WiFi transmitters. WiFi provides faster datarates, but in a smaller range, while the TVWS network has slower maximum datarates, but on a wider range. In Figure 1, node N1 is close to the transmitter, and might use WiFi to communicate, since it can offer a higher datarate compared to the TVWS network. Node N2 is instead at the edge of the WiFi transmitting range, and has to decide which radio technology can be used. WiFi might still provide good datarates, but could be suddenly shadowed by obstacles particularly if the user is moving, thus breaking the communication. Finally, node N3 is in an area in which the communication on WiFi is not feasible, and thus should use the TVWS network for its traffic. Clearly, when the receiver moves, the performance of the different RATs can change suddenly, due to shadowing and to the varying distance from the BS.

We foresee a client device offering both TVWS and 802.11n WiFi connectivity, which can thus connect to either network independently.

We present a preliminary study we have performed at the Mile End campus of the Queen Mary University London, in which we placed an access point which offered both TVWS and WiFi connectivity in a room, and moved around the rooms of the campus keeping track of the offered throughput. Figure 2 shows the results of the preliminary test we performed, measuring the offered throughput on the two technologies. Close to the access point, it is evident how the WiFi network can provide much higher datarates compared to the TVWS network. However, after few meters, the throughput of the WiFi network drops steeply, mainly due to obstacles and clearly to the distance from the access point. The TVWS network instead maintains a reasonable throughput (i.e. just short of 5 Mbps) even far from the access point. In fact, the bitrate smoothly decreases when moving farther from the access point.

We can see from Figure 2 that below 20 m the WiFi is certainly to be preferred, but farther than that, no connection is possible, and thus the TVWS network should be used to

[1] http://mediapm.edgesuite.net/dash/public/support-player/current/index.html

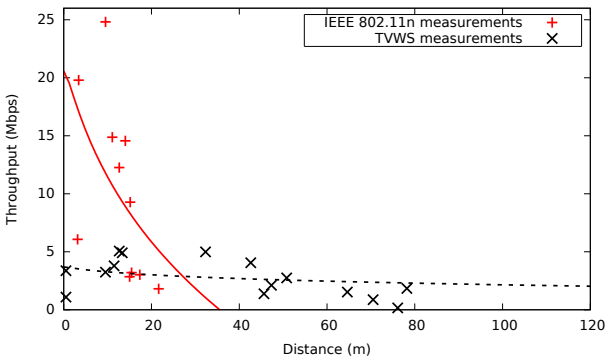


Fig. 2. The throughput difference between the IEEE 802.11n network and the TVWS network.

maintain connectivity. It is finally straightforward to note how Figure 2 closely follows the Shannon Capacity [14] [29].

It is important to note that we do not report the presence of obstacles for this preliminary study, but we will clarify it later. Certainly, although showing only the distance in the chart, the attenuation due to building structure severely degrades the IEEE 802.11n signal. The same does not happen for the TVWS network, which presents a rather stable signal regardless of possible obstacles between the transmitter and the receiver.

B. System Model

The MPEG-DASH client is fed with a manifest file, which contains the video description, and the segment URLs together with their length, resolution, codecs, the segment size and other meta information useful for the playback. The client also encompasses a rate adaptation algorithm, which chooses the quality depending on the network performance. For our test, we used the publicly available MPEG-DASH client made by Akamai[2].

We implemented our solution as a middleware which could either lay on the client device, or can be also implemented on the gateway router. Clearly, the latter solution might also add delays due to the network interface. At the same time, it does not require that the device physically have multiple connections, but performs the offloading on the gateway router.

The client issues an HTTP GET request of the desired segment, which arrives at our middleware, in charge of selecting the best interface through which the query has to be routed. We note that even if all the connections use TCP, and thus maintain a connection, different segments instantiate different connections, since they are all different HTTP GET, hence it is possible to manage each connection separately from the others.

IV. Algorithm

In this section, we describe our proposed algorithm, named CABA, which stands for Connection Aware Balancing Algorithm, and its implementation on a Raspberry Pi. We note that we based our analysis on the MPEG-DASH protocol, but that our framework is completely transparent to the HTTP-based streaming protocol used. Our algorithm is intended for *nomadic* use [30], in which the user is watching videos with occasional mobility, which however can change dramatically the throughput and thus the video quality. In Algorithm IV is described the pseudocode of the proposed CABA algorithm. We present in Section IV-A how we select the interface devoted to the segment download, while in Section IV-B we present our prefetching mechanism, implemented in CABA. In Table I are described all the notation used in this Section.

TABLE I
VARIABLES NOTATION

RI	Number of radio interfaces
N	Size of the downloaded segment history
$\Omega_k = \{\omega_1^k, \cdots, \omega_N^k\}$	Size of the last N downloaded segments
$\Delta_k = \{\delta_1^k, \cdots, \delta_N^k\}$	Download time of the last N segments
μ_k	Speed average for connection k
σ_k	Speed standard deviation for connection k
W	Size of the block windows
J	Number of the block windows ($J = \frac{N}{W}$)
$\Phi_k = \{\phi_1^k, \cdots, \phi_J^k\}$	Speed standard deviation for each block window on connection k
μ_j^k	Speed average for the block window j on connection k
R_k	Datarate index for connection k
P_k	Standard deviation index for connection k
DR_k	Expected datarate for connection k
t_k	Expected time to download a video segment on connection k
Q_k	Segment queue size on connection k
T_k	Expected time to get a video segment if requested on connection k
H	Average video segment size
L	Video segment duration (in seconds)
θ	Next video segment index to be downloaded
M	Number of video segments composing the requesting video

A. Interface selection

We assume that the device is equipped with RI radio interfaces, with $RI \geq 2$. Clearly, if $RI = 1$ there is only one possible interface devoted to the video segment download. For each radio interface k, with $1 \leq k \leq RI$, we define a vector $\Omega_k = \{\omega_1^k, \omega_2^k, \cdots, \omega_N^k\}$ containing the size of the last N downloaded segments over interface k, where ω_1^k and ω_N^k refer to the oldest and the newest segment received, respectively. Accordingly, we also define a vector $\Delta_k = \{\delta_1^k, \delta_2^k, \cdots, \delta_N^k\}$, which contains the time needed to download each segment. Then, we compute the average connection speed μ_k over the last N segments, and the standard deviation σ_k:

$$\mu_k = \frac{\sum_{\omega_i^k \in \Omega_k} \omega_i^k}{\sum_{\delta_i^k \in \Delta_k} \delta_i^k} \tag{1}$$

$$\sigma_k = \sqrt{\frac{\sum_{i=1}^{N} \left(\frac{\omega_i^k}{\delta_i^k} - \mu_k\right)^2}{N-1}} \tag{2}$$

The term μ_k indicates the average speed of the connection through interface k, while σ_k indicates its stability over time.

[2]http://mediapm.edgesuite.net/dash/public/support-player/current/index.html

Algorithm 1 CABA Algorithm

1: **function** CABA(θ)
2: **for** $k = 1$ **to** RI **do**
3: **if** $DR_k > 0$ **then**
4: $t_k = \frac{H}{DR_k}$
5: $T_k = t_k \cdot Q_k$
6: **else**
7: $T_k = \infty$
8: **end if**
9: **end for**
10: $min = \text{argmin}_{1 \leq k \leq RI}(T_k)$
11: $Q_{min} \leftarrow Q_{min} \cup \{\theta\}$
12: PREFETCH($\theta, C \setminus \{I_{min}\}$)
13: **end function**
14: **function** PREFETCH(θ, C')
15: **for** I_k **in** C' **do**
16: **if** $|Q_k| = 0$ **then**
17: $E_k = \frac{H}{DR_k}$
18: $F = \theta + \lceil \frac{\lceil E_k \rceil}{L} \rceil$
19: $Q_k \leftarrow Q_k \cup \{F\}$
20: **end if**
21: **end for**
22: **end function**
23: **function** UPDATEINTERFACES($\omega^k_{new}, \delta^k_{new}, \Omega_k, \Delta_k$)
24: $\Omega_k \leftarrow \Omega_k \cup \omega^k_{new}$
25: **if** $|\Omega_k| > N$ **then**
26: $\Omega_k \leftarrow \Omega_k \setminus \{\omega^k_{oldest}\}$
27: **end if**
28: $\Delta_k \leftarrow \Delta_k \cup \delta^k_{new}$
29: **if** $|\Delta_k| > N$ **then**
30: $\Delta_k \leftarrow \Delta_k \setminus \{\delta^k_{oldest}\}$
31: **end if**
32: $J = \frac{N}{W}$
33: **for** $j = 1$ **to** J **do**
34: $\mu^k_j = \frac{\sum_{i=(j-1)\cdot W+1}^{j\cdot W} \omega^k_i}{\sum_{i=(j-1)\cdot W+1}^{j\cdot W} \delta^k_i}$
35: $\phi^k_j = \sqrt{\frac{\sum_{i=(j-1)\cdot W+1}^{j\cdot W} \left(\frac{\omega^k_i}{\delta^k_i} - \mu^k_j\right)^2}{W-1}}$
36: **end for**
37: $R_k = \frac{\sum_{i=1}^{J} \left(\frac{i}{J}\right) \cdot \mu^k_i}{\sum_{i=1}^{J} \left(\frac{i}{J}\right)}$
38: $P_k = \frac{\sum_{i=1}^{J} \left(\frac{i}{J}\right) \cdot \phi^k_i}{\sum_{i=1}^{J} \left(\frac{i}{J}\right)}$
39: $DR_k = max(R_k - \sqrt{P_k}, 0)$
40: **end function**

Each time a new segment is downloaded from the radio interface k, with $1 \leq k \leq RI$, the vectors Ω_k and Δ_k are updated accordingly (function *UpdateInterfaces()* in Algorithm IV, lines 23-40). We then define a parameter $W \leq N$, and create a vector $\Phi = \{\phi_1, \phi_2, \cdots, \phi_J\}$ of size $J = \frac{N}{W}$ for each interface k where each item is defined as:

$$\phi^k_j = \sqrt{\frac{\sum_{i=(j-1)\cdot W+1}^{j\cdot W} \left(\frac{\omega^k_i}{\delta^k_i} - \mu^k_j\right)^2}{W-1}} \qquad j = 1..J \tag{3}$$

with

$$\mu^k_j = \frac{\sum_{i=(j-1)\cdot W+1}^{j\cdot W} \omega^k_i}{\sum_{i=(j-1)\cdot W+1}^{j\cdot W} \delta^k_i} \qquad j = 1..J \tag{4}$$

which basically computes the standard deviation for each of the J windows of size W. Clearly, we assume N multiple of W.

We then compute the following:

$$R_k = \frac{\sum_{i=1}^{J} \left(\frac{i}{J}\right) \cdot \mu^k_i}{\sum_{i=1}^{J} \left(\frac{i}{J}\right)} \tag{5}$$

$$P_k = \frac{\sum_{i=1}^{J} \left(\frac{i}{J}\right) \cdot \phi^k_i}{\sum_{i=1}^{J} \left(\frac{i}{J}\right)} \tag{6}$$

which highlights the recent performance of interface k. Clearly, R_k represents the datarate and P_k represents the standard deviation of interface k. Clearly, recent windows (i.e. $i \approx J$) are considered more than the others.

The expected datarate for the given interface k is hence forecast as:

$$DR_k = max(R_k - \sqrt{P_k}, 0) \tag{7}$$

Clearly, a large P_k means that the connection is experiencing bad conditions, and in case $P_k \geq R_k$, the connection is marked as unreliable with expected datarate put to 0. Otherwise, the connection is more stable, and thus the datarate DR_k will not be penalized too much.

With the method described above, the CABA algorithm constantly update the radio interfaces status and can hence exploit this awareness for making the best choice for the interface to use (function *CABA()* in Algorithm IV, lines 1-12).

In Figure 3 is depicted the flow diagram describing the CABA algorithm. After receiving the video request of index θ from the video client, the CABA algorithm estimates the expected download time T_k needed to get the video segment θ on each connection k, with $1 \leq k \leq RI$. Then the algorithm chooses the connection k that minimize the download time T_k. In more details, from the video manifest, we can get all the different video qualities along with the average segment size H. Thus the expected time to download a segment is defined as $t_k = \frac{H}{DR_k}$. Then, according to the size of the segment queue Q_k, we compute the expected time to deliver the next segment as:

$$T_k = t_k \cdot Q_k \tag{8}$$

Finally, the interface with the lowest T_k is selected to deliver the requested segment. If all the interfaces have $DR_k = 0$, we break the tie by picking the one that with the highest μ_k, that is the interface with the highest datarate, thus not considering its variance (this check should be done at line 10 of Algorithm IV, but is not inserted for space reasons).

B. Prefetching

In this section we detail our prefetching algorithm (line 14-22 of Algorithm IV), capable to leverage the multiple RAT installed on the device to offload part of the content and download it in advance. In the second part of Figure 3 is depicted the flow diagram of the prefetching method. After choosing the connection on which request the video segment θ, the CABA algorithm calls the *Prefetch* method. This method consists in requesting future video segments in advance of

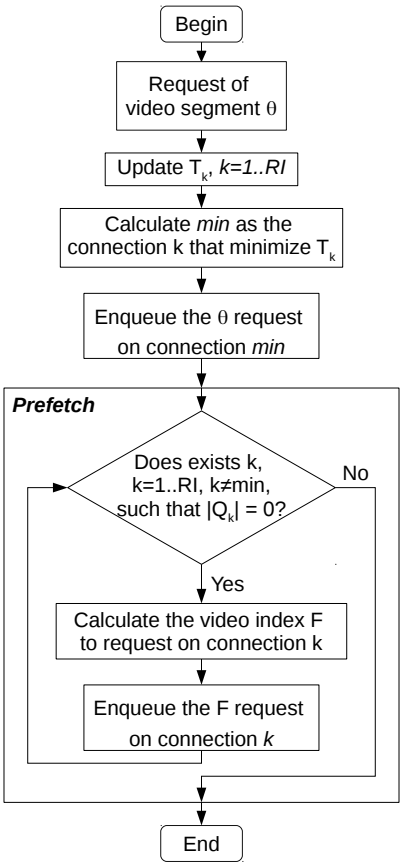


Fig. 3. Flow diagram describing the CABA algorithm

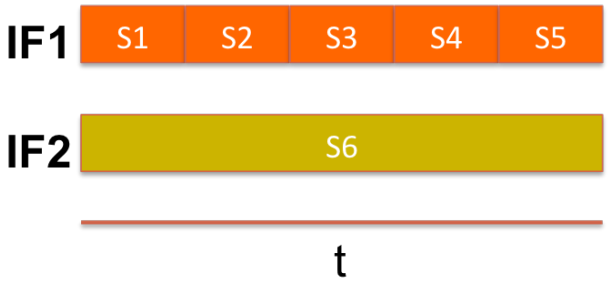


Fig. 4. The prefetching mechanism. IF1 offers a higher datarate, while IF2 is used to prefetch future segments to be played by the client.

the client requests and it is executed only on free connections having no pending requests. We picture this behavior in Figure 4. The IF1 can provide faster datarate compared to IF2, and thus it is selected as preferred interface to download DASH segments according to the client's requests. IF2 is slower, and thus it is used to prefetch segments the clients will ask for in the future. Selecting the segment size to download on the second interface is not trivial: downloading a too early segment might result in the client querying for it too soon, and thus having to download it on IF1 wasting the downloaded data on IF2. On the opposite, asking for a too late segment delays the time in which it will be asked by the client, thus reducing the benefits of the prefetching.

Let $C = \{I_1, \cdots, I_{RI}\}$ be the set of connections available on the device. From previous measurements, we also have defined $DR = \{DR_1, DR_2, \cdots, DR_{RI}\}$ as the set of average datarates for each connection. From the manifest, we know the average segment size H and the segment duration in seconds L, and we can also define a set $\Theta = \{1, 2, \cdots, M\}$ containing the IDs of all the segments of the video.

We assume that the client requests the segment of index θ at time T $(1 \leq \theta \leq M)$, and as we already stated we select the best available connection to download the segment, more precisely $i = argmin_i(T_i)$. Thus, the prefetching mechanism has to choose between the set $C' = C - I_i$ the candidate interface to download a future segment.

Hence

$$\forall I_k \in C' : E_k = \left(\frac{H}{DR_k} \right) \tag{9}$$

We note that we prefetch the segment with the same quality of the last requested segment. As before, should any ties between the remaining interfaces happen, we will pick the one with the highest datarate. We can then compute the segment to be downloaded F so that its download would finish before the client could request it as

$$F = \theta + \left\lceil \frac{\lceil E_k \rceil}{L} \right\rceil \tag{10}$$

We then proceed to effectively start the prefetch of the segment F on each free interface I_k.

V. Performance Evaluation

A. Analytic Study

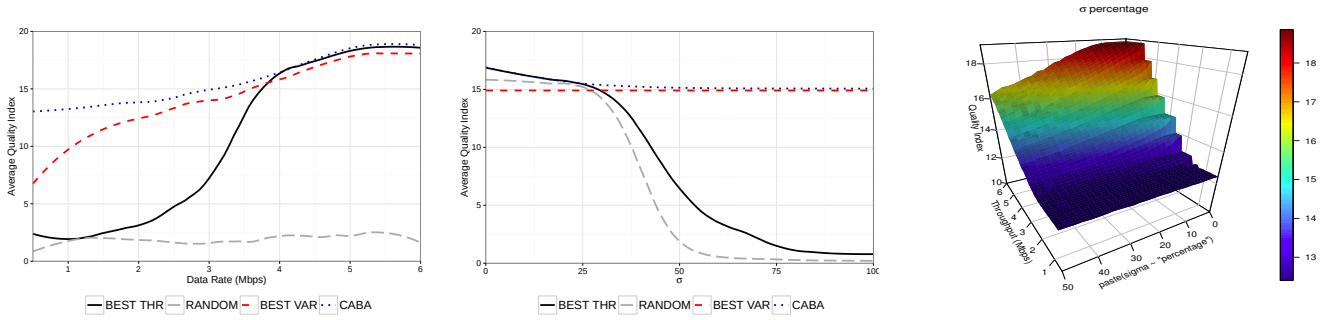
In this section we analyze the performance of CABA against other interface selection proposal. More in detail, we have implemented a simulator able to evaluate different interface selection algorithms, basing our analysis on the Big Buck Bunny video from [31]. We compare our proposal against 3 other interface selection algorithms, defined as follows:

- **BEST THROUGHPUT**: in this algorithm, each time a new packet has to be downloaded, we select the interface with the highest throughput.
- **BEST VARIANCE**: this algorithm selects the most stable interface.
- **RANDOM**: with this algorithm, at each step we randomly select one of the available interfaces

To perform a fair analysis, for this study we deactivate the prefetching algorithm explained in Section IV-B. The BEST THROUGHPUT and BEST VARIANCE algorithms only exploits in turn one of the features of CABA, since the BEST THROUGHPUT does not consider the stability of the connection, and the BEST VARIANCE does not exploit higher throughput connections, if unstable.

All the tests are performed assuming two different interfaces. The throughput of each interface i is computed assuming a Normal distribution with mean μ_i and standard deviation σ_i.

Figure 5(a) shows the performance of the four algorithms we studied by setting interface 1 with $\mu_1 = 3.5Mbps$ and



(a) (b) (c)

Fig. 5. Results of the analytic study.

$\sigma_1 = 1.8Mbps$, thus representing an unstable connection with fair datarate. Interface 2 is instead set as more stable, with $\sigma_2 = 0.15Mbps$ and varying throughput. The figure clearly shows the benefits of CABA, which achieves the best performance regardless of the performance of interface 2. The RANDOM algorithm performs the worst, as it is straightforward to imagine, due to inefficiencies and too high variations in the throughputs, which also make difficult for the rate adaptation algorithm to understand the preferable quality of the video to be streamed.

Figure 5(b) details a similar analysis, in which we set $\mu_2 = 4Mbps$, and we vary σ_2. As $\mu_2 > \mu_1$, the `BEST THROUGHPUT` algorithm prefers to chose interface 2 when it is stable (i.e. low σ_2). However, as σ_2 increases, the choice of the best interface is more difficult, and both the rate adaptation algorithm as well as the interface selection algorithm struggle, delivering poor performance. The `BEST VARIANCE` solution leverages the stability of the connection, thus selecting the most stable connection among the two available, although not exploiting the higher throughput of interface 2 when σ_2 is low. CABA instead exploits this behavior by preferring interface 2 when it is stable (i.e. the left part of Figure 5(b)), and selecting the most stable one when interface 2 varies too much.

Figure 5(c) shows a 3D analysis keeping interface 2 with $\mu_2 = 2Mbps$ and $\sigma_2 = 0.4Mbps$, and varying both μ_1 and σ_1, where σ_1 is plotted as the percentage with respect to μ_1. Clearly, the analysis is performed only for the CABA algorithm. The chart confirms that both the throughput and its stability play a dominant role in the quality of the video in play.

B. Multipath comparison

Multipath TCP (MPTCP) is the de-facto standard solution when exploiting multi-homed devices, as it offers a low layer load balance solution which can span among different applications and services. In literature, it is possible to find many studies on MPTCP [23] [32], also related to video streaming [24]. Clearly, results vary, but in general MPTCP offers good performance when the connections employed show similar characteristics, and lower when the two connections differ too much.

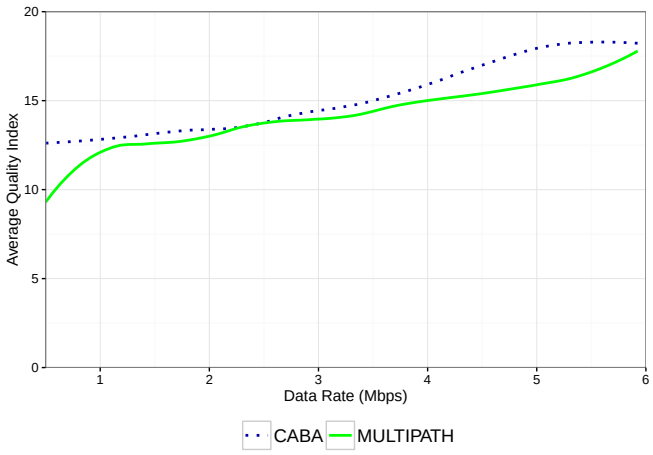


Fig. 6. Results from the MPTCP comparison.

Figure 6 shows a comparison between CABA and MPTCP. To do that, we set up a testing environment in which we installed MPTCP both on a server, containing all the video segments of Big Buck Bunny, and on the client, running MPTCP version 0.91.3 with *fullmesh* as path manager and *default* as scheduler. We compare it with CABA, for which we disabled MPTCP. The throughput of the two networks is configured the same way as for the tests of Figure 5(a).

At first, we can see that although CABA follows a similar trend as in Figure 5(a), the average value is a bit lower, since in this test we dealt with real networks, which thus can carry latency and throughput variance, which we did not had for the tests of Figure 5(a).

MPTCP always offers lower performance than CABA, except around 2.5 Mbps in which the two algorithms perform similarly. This is because the two connections perform similarly, hence MPTCP can better distribute the traffic between the two. This also confirms previous studies such as [23] [32], which highlighted the benefits of MPTCP exploiting similar connections, but also showed its limits with heterogeneous connections. In fact, when one of the connections offer low throughput (i.e. leftmost part of Figure 6), still MPTCP tries

to download some bytes through it, thus penalizing the stream. Also, when the throughput of such connection increases (i.e. rightmost part of Figure 6), again MPTCP tries to download some bytes through the other connection. However, the average QI is much more, as the total throughput is higher.

C. Emulation Analysis

In this section we describe a preliminary analysis we performed, based on real data gathered through an official Ofcom testbed.

In 2014, several partners conducted a measurement campaign to test the effectiveness and the differences between IEEE 802.11af and IEEE 802.11n networks for in-home video streaming [33]. The report consists of measurements obtained in 4 different houses in the city of Glasgow, Scotland. The focus of the test was on video streaming, and the report shows the performance of both networks in different locations for each house. The full report is available at [33]. The rationale is that the IEEE 802.11n network offered higher throughput when close to the access point, delivering slightly more than 30 Mbps of TCP throughput, while the IEEE 802.11af network achieved a maximum of 10 Mbps. Clearly, UDP throughput is higher, reaching around 40 Mbps for the 802.11n network and 15 Mbps for the 802.11af network. The conclusion of the report states that the throughput of the 802.11af network is expected to increase in the future, as the driver of the devices is still being optimized. Moreover, the 802.11n network was configured to work at 20 MHz both for the 2.4 GHz and for the 5 GHz band, while the 802.11af network worked at 8 MHz, although in the future it is expected to be able to bond different channels together to increase the bandwidth.

To emulate the two networks behavior, we use the well known Traffic Control (TC) utility available in GNU/Linux distributions, which allows to set the connection speed of any interface. We used two of the available interfaces of our test computer, one WiFi and an Ethernet connection. Both are connected through two different high speed networks to the Internet.

For this evaluation, the video player is run on the Raspberry Pi, which also runs the algorithm presented in Section IV. We played the video Big Buck Bunny from the dataset presented in [34].

For each of the four houses, we select a location and test the effectiveness of our proposal. Finally, we also evaluate a dynamic scenario in house 1, in which a client virtually moves around the house, thus experiencing different throughputs. We note that these test are obtained with a single run, and thus might present throughput variations due to the real network congestion. Thus, they should be only considered as qualitative and not quantitative results.

The summary of the throughputs experienced in the different locations is shown in Table II. It is possible to see how the 802.11n network is not connected in one of the location, and in another one presents low performance. In contrast, the TVWS network is always connected and presents a more stable throughput.

TABLE II
THROUGHPUTS

House	Location	Throughput 802.11af (Mbps)	Throughput 802.11n (Mbps)
1	3	8.93	11
2	1	10	33
3	1	4.83	0.6
4	3	6.6	-

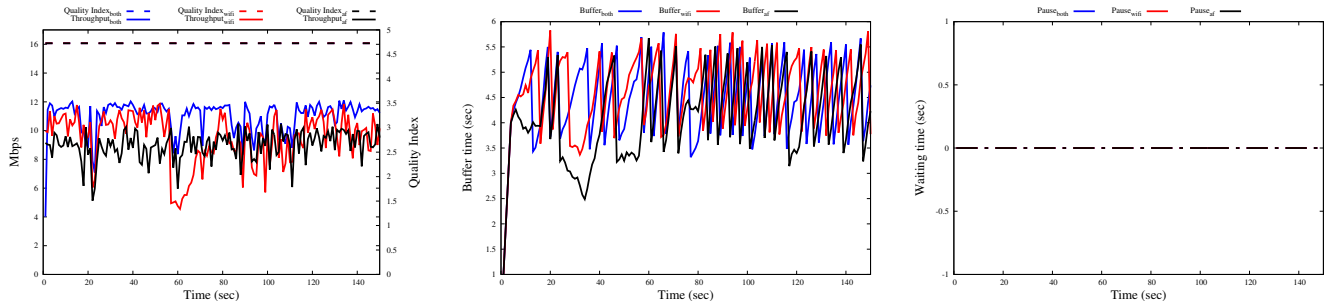
The first test was performed in location 3 in house number 1. Both throughputs are similar, and as it is possible to be seen in the video, both networks perform roughly the same. To better evaluate the benefits of the different proposals, we study the Quality Index (QI) instead of the video bitrate. We do this in order to be consistent with effective quality played, as the bitrate of a specific video segments may vary from the previous one even if the resolution and frame rates are the same, due to video compression techniques. Figure 7(a) shows the throughput and the quality index, which is always at top since the throughput is enough for both networks to play it at the maximum quality. Figure 7(b) shows instead the pause times, while Figure 7(c) shows the buffer size. Pause is never experienced, and the buffer level is always near the maximum. In this scenario, our proposed algorithm tops the others in terms of throughput, as it account for small variations on the throughputs of the two connections, and routes traffic accordingly.

The test performed on throughputs of house 2 are similar, as both throughputs are high and thus Figures 8(a)-8(c) clearly shows that both throughputs are basically the same, as well as the other investigated metrics.

House 3 presented different throughputs, as the 802.11n connection offer a low throughput in location 1. Figures 9(a)-9(c) report the results of the test, which clearly shows that the 802.11n is not able to support a high quality streaming. The 802.11af network delivers a good quality video, in line with our algorithm which clearly selects it to stream the content, as the plotted lines are quite similar. The buffer level present a chart in which the 802.11n network have a bigger buffer at the end of the test. It is worth to note that this happens since the much lower quality played translates into smaller packets, which are downloaded by the client to fill the buffer. The 802.11af network and our algorithm instead, although playing a high quality video, have a low buffer level, since the maximum throughput is more or less the same of the higher quality of the played video.

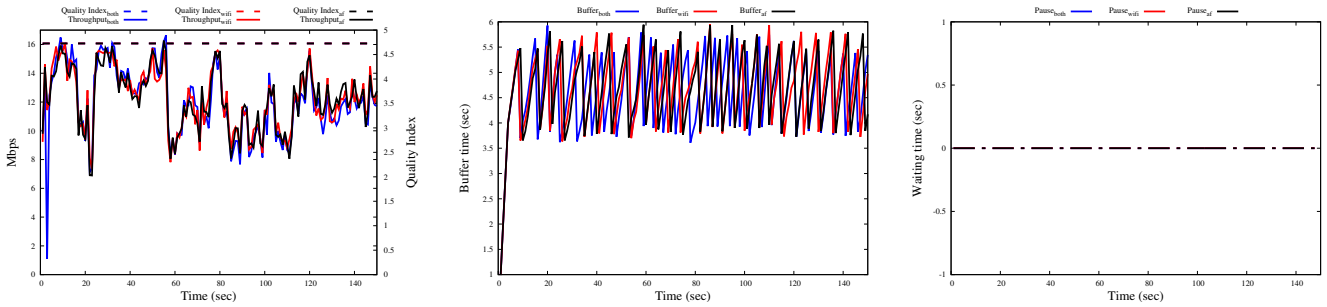
The last static test was performed in location 3 of house 4. Here, no 802.11n throughput is reported. Therefore, this test is just to show the ability for our algorithm to work even if one of the available connections does not work or could not be used to establish a connection. Still, the performance of the played video are quite good, and the quality is near the maximum one available. The results of this test are shown in Figures 10(a)-10(c).

The last test was performed using the throughputs experienced in house 1, but with the client virtually moving from



(a) (b) (c)

Fig. 7. Test in location 3 in house number 1



(a) (b) (c)

Fig. 8. Test in location 1 in house number 2.

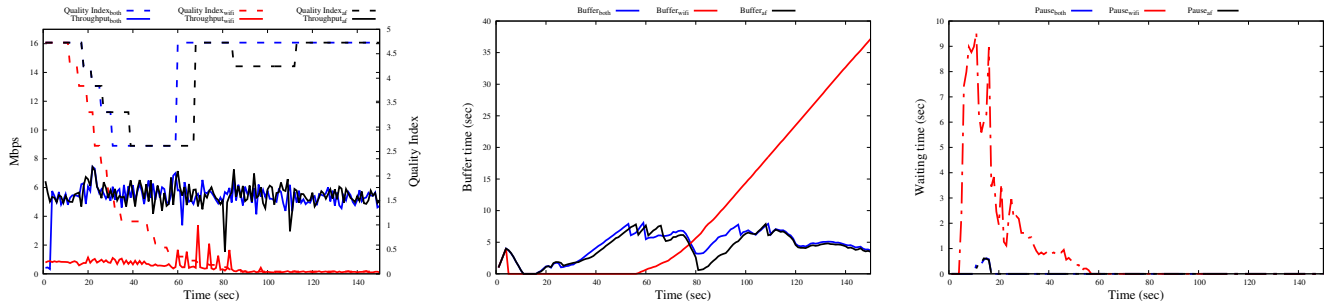
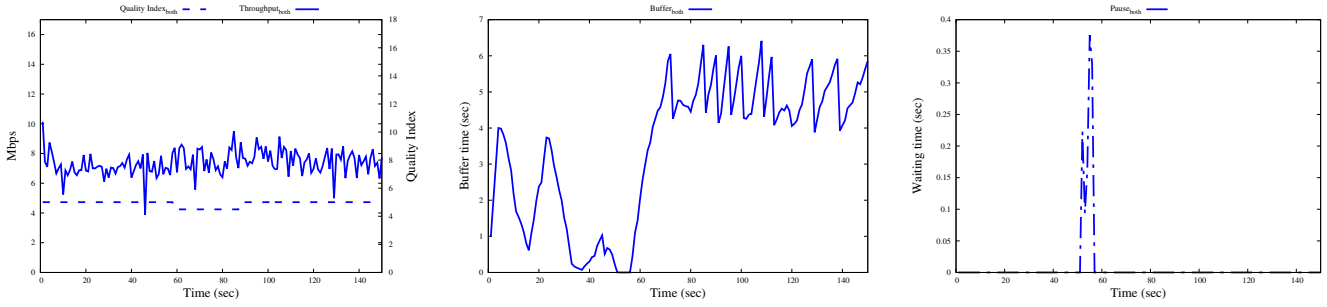


Fig. 9. Test in location 1 in house number 3.



(a) (b) (c)

Fig. 10. Test in location 3 in house number 4.

one point to another. The test starts in location 2, where we experience 10.1 Mbps for the 802.11af network and 24 Mbps for the 802.11n network. We then move after 50 seconds to location 3 (for which the throughputs have been presented in Table II) and finally after 50 seconds we eventually move to location 4, where the throughput of the 802.11af network is reported as 5 Mbps, while the 802.11n network as 1.46 Mbps, thus changing the network which achieves the best throughput. After 50 seconds we move back to location 3, and after further 50 seconds we finally move to location 2, where we keep running the test for 50 more seconds.

Figure 11 reports the results, where it is evident the drop in throughput after the first two locations (at 100 seconds). Here, the 802.11af network still delivers good performance, as well as our algorithm. The 802.11n network instead offers lower throughput, and the video quality is affected as it is played to a lower quality. The throughput increases at 150 seconds and so does the quality, but the WiFi continues to struggle due to the instability of the connection, which makes the client to lower the quality of the playback in order not to introduce pauses in the streaming.

The tests just presented confirm the increased benefits of offloading part of the video streaming data to the secondary TVWS network, which can still deliver good performance when the primary network fail.

D. Testbed

In this section we describe the testbed configuration we used in our experiments.

Figure 12 shows our testbed setup, with a Raspberry Pi on the right, in which we implemented our algorithm, and the Carlson Wireless TVWS device on the left, connected through an Ethernet cable. The Raspberry Pi has two IEEE 802.11n interfaces installed, tuned to channel 6 and 11. The interface on channel 6 is used as an AP to which clients can connect, while the interface on channel 11 is used to connect to the AP placed next to the TVWS receiver device, which offers internet connectivity. We specifically choose these two channels as they were the least interfered in the scenario in which we ran the tests.

E. Testbed Results

In this section we present the results obtained in the test we performed at the Mile End campus of the Queen Mary University of London.

We placed our combined AP at the third floor of the EECS building of the QMUL Mile End Campus, in room 358. We then moved through the campus, in the same building and in others nearby, playing the Big Buck Bunny video from [34]. The client runs in a laptop, connected through WiFi to the Raspberry Pi, which could choose the interface through which getting the video segments between TVWS and WiFi.

For our tests, we present results about 12 different rooms, and averaged over all the tests. Over all the locations tested, in 6 of them we received both WiFi as well as TVWS signals. For those rooms, we present results about raw measurements and using CABA. In other 6 rooms, we only received TVWS signal, and thus we only show results for this technology. The rooms 353, 358 and 354 are at the same floor of the AP (i.e. the third), while rooms 250, 251 and 254 are one floor below it (i.e. at the second floor). Rooms in which we did not receive any WiFi signal are located farther, in other buildings, and present distances between 40 meters and 100 meters.

TABLE III
RAW DATARATES OBTAINED IN THE TESTED LOCATIONS

Testing Location	WiFi Datarate (Mbps)	TVWS Datarate (Mbps)
353	1.35	2.6
358	21	3.7
251	6.05	4.48
254	5.82	4.24
354	25	1.5
250	5.30	4.74
br3.1	0	2.66
ppground	0	2.67
ppgroundcorridor	0	2.39
ITL meeting room	0	6.08
cshub	0	2.79
firstqueens	0	1.33

In Table III we present the maximum datarate achieved in all the locations tested. As it is straightforward to note, WiFi offers the best datarate in 5 out of the 6 locations in which we received it. On the inverse, TVWS brings a more reliable connection, but at the cost of a slower speed. It is worth to note a couple of issues: at first, the device we used for TVWS is quite new, compared to a heavily tested standard installed on the WiFi access point we used during the test. Thus, it is expected that future improvements on the TVWS device and chip manufacturing would also improve its performance. At second, IEEE 802.11af offers to bond together up to 4 channels, thus increasing the bandwidth and eventually the datarate. For our tests, we only used a single channel to transmit.

Starting from around 25 meters away from the access point, we only received TVWS signal. Thus, for the remaining 6 rooms we only report measurements for this technology.

To summarize this data, it is worth to mention the fact that during the test, once we achieved TVWS connectivity, the communication link was very stable, without any drop in datarate and channel quality. On the inverse, WiFi suffered from more interference and its channel quality variance had a strong impact on the results we are going to show.

F. Playback Quality

The playback quality tells the resolution and frame rate at which the video is played. Clearly, a higher playback quality eventually means for the user a better Quality of Experience. We present these results for 3 minutes of playback of the Big Buck Bunny video [34], after an initial transient of 50 seconds to account for connection establishment which would deviate the data if included.

Figures 13(a)-13(f) present the average playback quality achieved in the 6 locations where we both had TVWS and

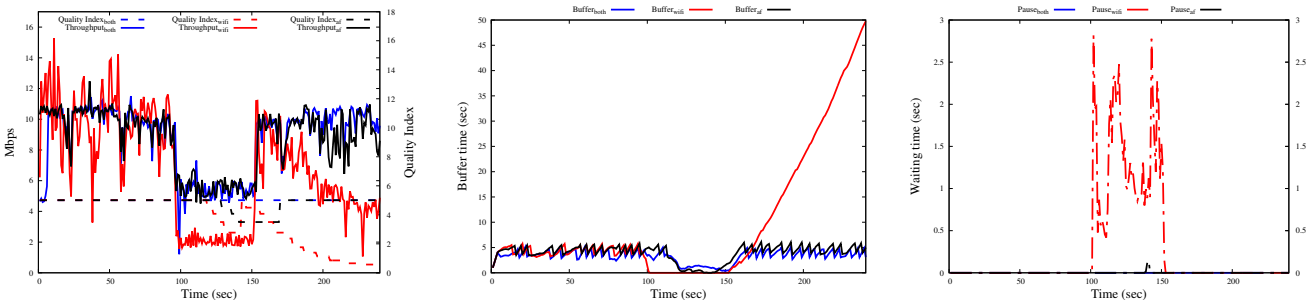


Fig. 11. Test result for the simulated dynamic scenario.

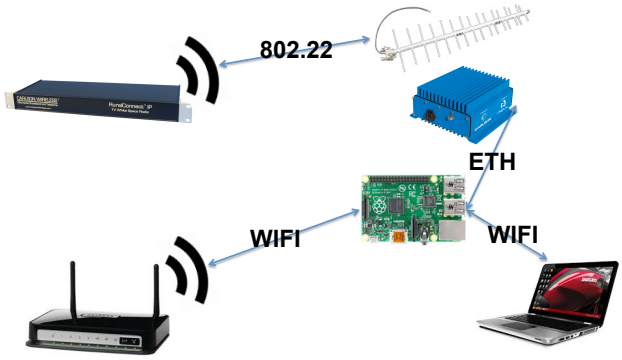


Fig. 12. The device configuration we used in our experiments.

WiFi connectivity. In all but room 353, the WiFi network offered the best performance. In fact, it was the room where we received the worse WiFi performance, even if it is not far from the AP location. The reason falls probably into a huge shadowing by obstacles. Regardless of the room, CABA always shows top performance. Clearly, it cannot be higher than the WiFi network if it already offers the best possible video quality achievable, which is around 4.7 Mbit/s for a 1920x1080 Full HD video with 24 as framerate [34].

In Figure 14 we show instead the playback quality in the 6 locations in which we only received the TVWS link. In 3 out of 6 rooms the quality is below 1.5 Mbps, but the video is still playable. In the other 3 rooms the quality almost exceeds 3 Mbps, which confirms how TVWS are able to fill possible coverage holes thanks to a wider communication range and a better obstacle resilience.

Figure 15 present the usage ratio of each interface in the 6 locations where we had both TVWS and WiFi connectivity. Clearly, we don't show results for TVWS or WiFi alone, as there is only one interface to use and thus there cannot be any offloading to the secondary interface. CABA balances the load, and offloads some of the content to the worst connection to prefetch segment to be played at a later time. In all the charts, after an initial setup phase, in which the algorithm has to understand the performance of the available connections, it stabilizes depending on the performance of the connections.

Finally, Figure 16 presents the average playback quality in all the locations. We note that we only averaged the WiFi playback qualities for the 6 locations in which we received it. CABA is instead averaged over all the 12 locations, as well as TVWS. ONLY-TVWS is instead the average of the 6 locations where we had no 802.11n connectivity but only TVWS. Regardless of the position, room or network condition, CABA is able to deliver the best results. WiFi still offers similar results, but we remark that in 6 out of 12 locations we could not get any signal, and thus we were not able to play the video. TVWS enhance the range, and still offers decent performance. As we already noted, we are comparing a well tested IEEE 802.11n AP against a prototype, which presents big margins of improvements and will certainly offer better results in the future.

G. Buffer Time

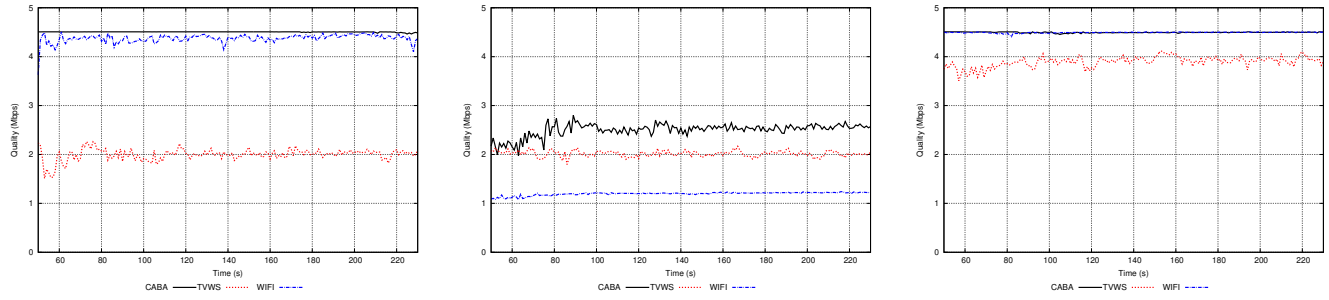
In this section we analyze the buffer time in the 6 different locations in which we received both TVWS signal as well as 802.11n. The buffer time tells how much the playback is considered stable by the client, which uses this information to decide the playback quality.

By looking at Figures 17(a)-17(f), we can find a similar pattern compared to Figures 13(a)-13(f). This is because a higher buffer time allows the client to increase the playback quality, while a lower buffer times forces it to be prudent and favor lower quality fragments, since they are smaller and can thus be downloaded in less time. We note that we only show this for the 6 locations in which we received both TVWS and 802.11n signal, because in the other rooms we cannot compare to anything since we only receive TVWS.

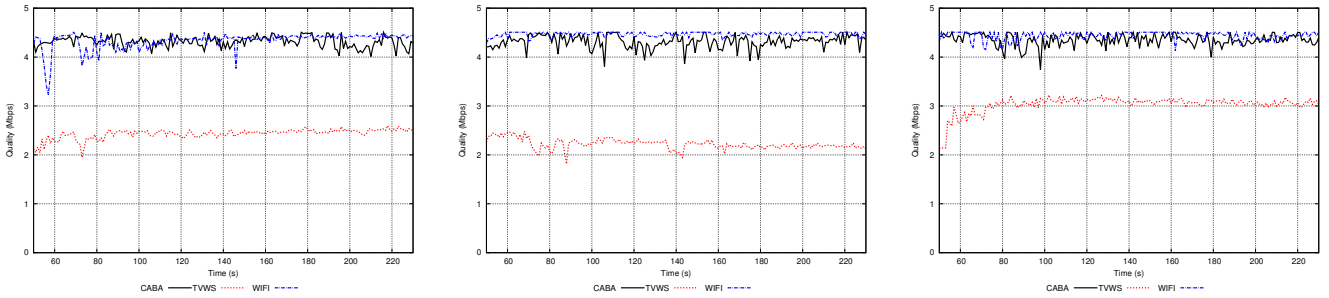
H. Pause Time

The last analysis we perform is about the pause times. When playing a video, users would allow for a change in the playback quality, but cannot bear to have any pause in the video playback. Hence, the pause time represent how many seconds the video stopped during our test. Again, we note that we only show this for the 6 locations in which we received both TVWS as well as 802.11n.

Figures 18(a)-18(f) show the results, and the most important thing to note is that again, regardless of the room, CABA offers the best performance, by achieving an average of 0 seconds of pause. That means that the video would never stop, and thus the QoE for the users would only depend on the playback



(a) 358 (b) 353 (c) 354



(d) 251 (e) 254 (f) 250

Fig. 13. Figures 13(a)-13(f) present the average playback quality in the 6 tested locations in which we received both TVWS and 802.11n WiFi.

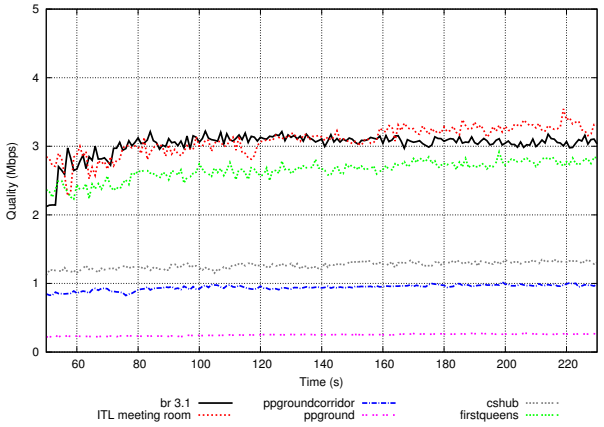


Fig. 14. Average playback quality in the 6 locations where we received only TVWS.

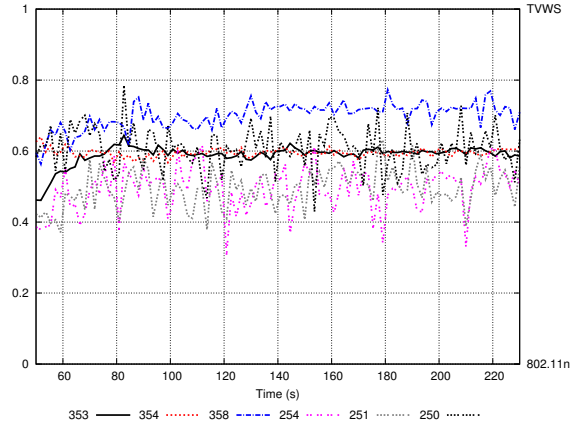


Fig. 15. Load balance between TVWS and WiFi in the 6 locations where we received both TVWS and 802.11n WiFi

quality, which we already showed it is the best one among the tested connections. In room 354 all the considered Radio Interfaces proceed without interruption, since we achieved 0 seconds of pause also for TVWS and for WiFi. In the other 5 rooms, we obtain a small amount of pause for both TVWS and for IEEE 802.11n. While for TVWS this is certainly due to the lower datarate offered, for WiFi again it is because of the shadowing and possible congestion on the channel.

I. CABA Efficiency

Table IV shows the ratio of prefetched packets played by the client. The rationale is that if the algorithms prefetches a too early segment, that segment might be requested by the client before the algorithm finished to prefetch it. Thus, the segment is downloaded also on the primary interface, wasting the time in which the other interface has been used to prefetch the data. In this table we show the ratio Pl/Pr, in which Pl is the number of prefetched segments played by the client, versus Pr which is instead the total of prefetched segments. Clearly, 0 means that all the packets prefetched were not actually played by the client, while 1 means that all of them were used to play the video. Therefore, the remaining percentage is bandwidth wasted to download segments which are not used by the client, but downloaded again over the primary interface.

CABA achieves an efficiency of more than 80 % in all the 6 locations where we received both TVWS as well as 802.11n. When computing the expected time of download, clearly we

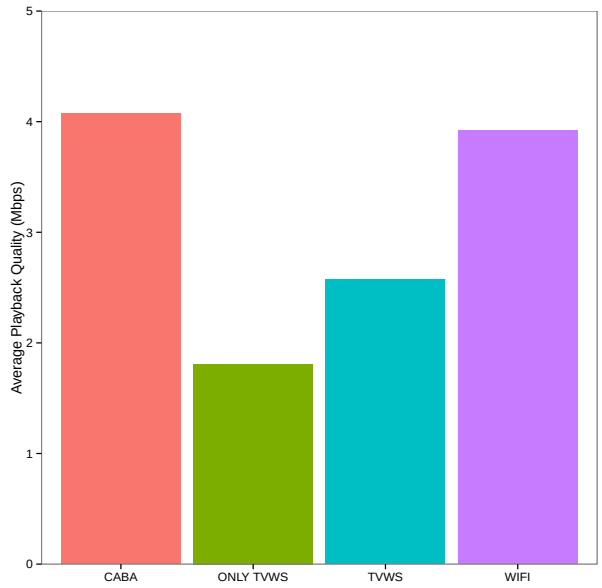


Fig. 16. Average playback quality regardless of the location.

TABLE IV
CABA EFFICIENCY

Testing Location	Efficiency (%)
353	91.14
358	85.28
251	82.33
254	86.56
354	85.28
250	85.49

do not know a priori the size of the segment to be downloaded. Thus, the expected time might be smaller than the actual time needed to download it. We leave the study on an adaptive method to raise the efficiency of CABA as future work.

VI. CONCLUSION

In this paper we presented CABA, a Connection Aware Balancing Algorithm focused on Dynamic Adaptive Video Streaming, able to balance the load and route the traffic through the best possible interface given the network conditions. We have given both the technical details as well as performance evaluation of the proposal. We based our analysis on adaptive video streaming with the well known MPEG-DASH protocol, but we remark that our proposal is general and compliant with other HTTP-based adaptive dynamic streaming protocols.

We presented the results of field tests performed at the QMUL Mile End campus, in which we ran a multi interface AP with TVWS and IEEE 802.11n. The performance evaluation shows how CABA can effectively improve the Quality of Experience of the user watching the video, by distributing the segment downloads through the available RATs. Moreover, TVWS provide a larger communication range, thanks to the lower frequency used and thanks to a better obstacle penetration.

Future work on this topic include the evaluation of a dynamic scenario, with a user moving while playing the video. We also plan to analyze the best segment size based on the network conditions, as well as increasing the number of scenario in which we run our tests.

REFERENCES

[1] M. Bennis, M. Simsek, A. Czylwik, W. Saad, S. Valentin, and M. Debbah, "When cellular meets WiFi in wireless small cell networks," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 44–50, jun 2013.

[2] GWS, "Report on Mobile Broadband at home," Tech. Rep., 2015.

[3] T. Cisco, "Cisco Visual Networking Index : Global Mobile Data Traffic Forecast Update , 2014 - 2019," *Growth Lakeland*, vol. 2011, no. 4, pp. 2010–2015, 2011.

[4] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, 2011, pp. 133–144.

[5] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE Multimedia*, vol. 18, no. 4, pp. 62–67, apr 2011.

[6] Y. Liang, K. Chen, G. Li, and P. Mähönen, "Cognitive Radio Networking and Communications: An Overview," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 7, pp. 3386–3407, sep 2011.

[7] J. van de Beek, J. Riihijärvi, A. Achtzehn, and P. Mähönen, "TV White Space in Europe," *IEEE Transactions on Mobile Computing*, vol. 11, no. 2, pp. 178–188, feb 2012.

[8] Z. Qin, Y. Gao, and C. G. Parini, "Data-Assisted Low Complexity Compressive Spectrum Sensing on Real-Time Signals Under Sub-Nyquist Rate," *IEEE Transactions on Wireless Communications*, vol. 15, no. 2, pp. 1174–1185, feb 2016.

[9] L. Bedogni, A. Achtzehn, M. Petrova, and P. Mähönen, "Smart meters with TV gray spaces connectivity: A feasibility study for two reference network topologies," in *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, jun 2014, pp. 537–545.

[10] L. Ding, S. Pudlewski, T. Melodia, S. Batalama, J. D. Matyjas, and M. J. Medley, "Distributed Spectrum Sharing for Video Streaming in Cognitive Radio Ad Hoc Networks," in *Spectrum*, 2010, vol. 28, pp. 855–867.

[11] L. Bononi, M. Di Felice, A. Molinaro, and S. Pizzi, "A Cross-Layer Architecture for Robust Video Streaming over Multi-Radio Multi-Channel Wireless Mesh Networks," *Mobiwac09: Proceedings of the Seventh Acm International Symposium on Mobility Management and Wireless Access*, pp. 75–82, 2009.

[12] R. Kateja, N. Baranasuriya, V. Navda, and V. N. Padmanabhan, "DiversiFi: Robust Multi-Link Interactive Streaming," in *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies - CoNEXT '15*. New York, New York, USA: ACM Press, 2015, pp. 1–13.

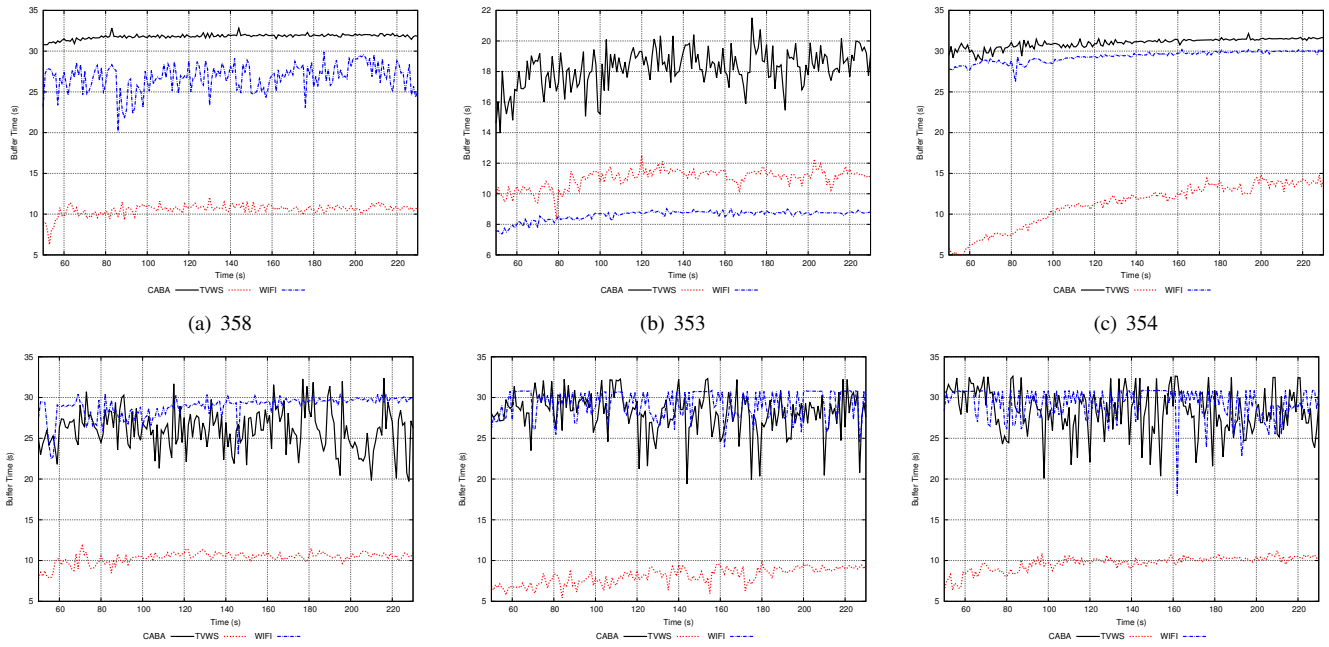
[13] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," *Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys)*, p. 209, 2010.

[14] L. Bedogni, M. Di Felice, F. Malabocchia, and L. Bononi, "Cognitive Modulation and Coding scheme Adaptation for 802.11n and 802.11af networks," in *Globecom 2014 Workshop - Telecommunications Standards - From Research to Standards (GC14 WS - TCS)*, 2014.

[15] L. Bedogni, A. Trotta, and M. Di Felice, "On 3-dimensional spectrum sharing for TV white and Gray Space networks," in *2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, jun 2015, pp. 1–8.

[16] S. Dimatteo, P. Hui, B. Han, and V. O. K. Li, "Cellular traffic offloading through WiFi networks," in *Proceedings - 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems, MASS 2011*, 2011, pp. 192–201.

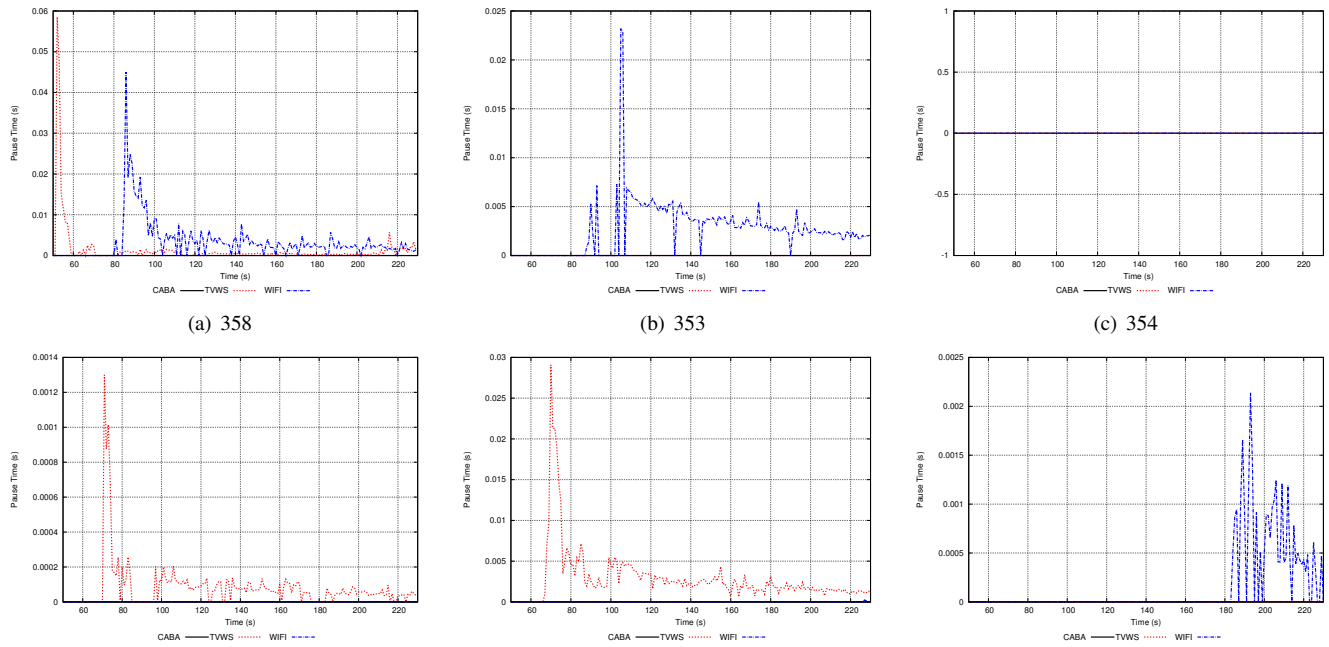
[17] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, and A. Bragagnini, "Offloading cellular networks with information-centric networking: The case of video streaming," in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2012 - Digital Proceedings*, 2012.



(a) 358 (b) 353 (c) 354

(d) 251 (e) 254 (f) 250

Fig. 17. Buffer times for the 6 tested locations in which we received both TVWS as well as IEEE 802.11n.



(a) 358 (b) 353 (c) 354

(d) 251 (e) 254 (f) 250

Fig. 18. Pause times for the 6 tested locations in which we received both TVWS as well as IEEE 802.11n.

[18] C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj, "Rate adaptation for dynamic adaptive streaming over HTTP in content distribution network," *Signal Processing: Image Communication*, vol. 27, no. 4, pp. 288–311, apr 2012.

[19] L. Keller, C. Fragouli, and U. C. Irvine, "MicroCast : Cooperative Video Streaming on Smartphones Categories and Subject Descriptors," *MobiSys 2012*, pp. 57–69, 2012.

[20] F. Malabocchia, R. Corgiolu, M. Martina, A. Detti, B. Ricci, and N. Blefari-Melazzi, "Using Information Centric Networking for Mobile Devices Cooperation at the Network Edge," in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. IEEE, may 2015, pp. 1–6.

[21] Q. Zhang, X. Zhang, Y. Gao, O. Holland, M. Dohler, and J. M. Chareau, "TV White Space Network Provisioning with Directional and Omni-directional Terminal Antennas," in *Proc. of IEEE VTC-Spring*, 2016.

[22] C. Paasch and O. Bonaventure, "Multipath TCP," *Communications of the ACM*, vol. 57, no. 4, pp. 51–57, 2014.

[23] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan, "WiFi , LTE , or Both ? Measuring Multi-Homed Wireless Internet Performance," *Imc*, pp. 181–194, 2014.

[24] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "MP-DASH," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies - CoNEXT '16*. New York, New York, USA: ACM Press, 2016, pp. 129–143.

[25] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments," *Proceedings of the 4th Workshop on Mobile Video - MoVid '12*, p. 37, 2012.

[26] B. Oztas, M. T. Pourazad, P. Nasiopoulos, I. Sodagar, and V. C. M. Leung, "A rate adaptation approach for streaming multiview plus depth content," in *2014 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, feb 2014, pp. 1006–1010.

[27] Y. Cao, X. You, J. Wang, and L. Song, "A QoE friendly rate adaptation method for DASH," in *2014 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*. IEEE, jun 2014, pp. 1–6.

[28] D. J. Vergados, A. Michalas, A. Sgora, and D. D. Vergados, "A fuzzy controller for rate adaptation in MPEG-DASH clients," in *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*. IEEE, sep 2014, pp. 2008–2012.

[29] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Prentice Hall PTR, 2001.

[30] ITU-R, "F.1399 : Vocabulary of terms for wireless access," Tech. Rep., 2001.

[31] S. Lederer, C. Mueller, C. Timmerer, C. Concolato, J. Le Feuvre, and K. Fliegel, "Distributed DASH dataset," in *Proceedings of the 4th ACM Multimedia Systems Conference on - MMSys '13*. New York, New York, USA: ACM Press, 2013, pp. 131–135.

[32] S. Ferlin, T. Dreibholz, and O. Alay, "Multi-path transport over heterogeneous wireless networks: Does it really pay off?" pp. 4807–4813, 2014.

[33] C. f. W. S. Communications, "In-Home TVWS Measurements Using IEEE 802.11af - http://www.wirelesswhitespace.org/projects/in-home-tvws-measurements-using-ieee-802-11af/," Tech. Rep., 2014.

[34] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *Proceedings of the 3rd Multimedia Systems Conference on - MMSys '12*. New York, New York, USA: ACM Press, 2012, p. 89.