QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming

STEFANO PETRANGELI, JEROEN FAMAEY, MAXIM CLAEYS and FILIP DE TURCK,
Department of Information Technology (INTEC), Ghent University–iMinds
STEVEN LATRÉ, Department of Mathematics and Computer Science, University of Antwerp–iMinds

HTTP Adaptive Streaming (HAS) is quickly becoming the de facto standard for video streaming services. In HAS, each video is temporally segmented and stored in different quality levels. Rate adaptation heuristics, deployed at the video player, allow the most appropriate level to be dynamically requested, based on the current network conditions. It has been shown that today's heuristics underperforms when multiple clients consume video at the same time, due to fairness issues among clients. Concretely, this means that different clients negatively influence each other as they compete for shared network resources. In this article, we propose a novel rate adaptation algorithm called FINEAS (Fair In-Network Enhanced Adaptive Streaming), capable of increasing clients' Quality of Experience (QoE) and achieving fairness in a multi-client setting. A key element of this approach is an in-network system of coordination proxies in charge of facilitating fair resource sharing among clients. The strength of this approach is threefold. First, fairness is achieved without explicit communication among clients and thus no significant overhead is introduced into the network. Second, the system of coordination proxies is transparent to the clients, i.e. the clients do not need to be aware of its presence. Third, the HAS principle is maintained, as the in-network components only provide the clients with new information and suggestions, while the rate adaptation decision remains the sole responsibility of the clients themselves. We evaluate this novel approach through simulations, under highly variable bandwidth conditions and in several multi-client scenarios. We show how the proposed approach can improve fairness up to 80% compared to state-of-the-art HAS heuristics in a scenario with three networks, each containing 30 clients streaming video at the same time.

Categories and Subject Descriptors: C.2.3 [**Computer-Communication Networks**]: Network operations—*Network Management*; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*

General Terms: Algorithms, Design, Management, Performance

Additional Key Words and Phrases: Experimental Evaluation, Fairness, HTTP Adaptive Streaming, Microsoft Smooth Streaming, Mobile, Quality of Experience, Rate Adaptation

ACM Reference Format:
Stefano Petrangeli, Jeroen Famaey, Maxim Claeys, Steven Latré and Filip De Turck. 2014. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 0, 0, Article 0 (0000), 23 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Nowadays, video streaming applications are responsible for the largest portion of the traffic exchanged over the Internet. Particularly, HTTP Adaptive Streaming (HAS)

Author's address: S. Petrangeli, M. Claeys, and F. De Turck, Gaston Crommenlaan 8 (Bus 201), 9050 Ghent, Belgium; email: name.surname@intec.ugent.be; J. Famaey, and S. Latré, Middelheimlaan 1, 2020 Antwerp, Belgium; email:name.surname@uantwerpen.be

protocols have become very popular due to their flexibility, and can therefore be considered as the de facto standard for video streaming services. Microsoft's Smooth Streaming, Apple's HTTP Live Streaming, Adobe's HTTP Dynamic Streaming and MPEG Dynamic Adaptive Streaming over HTTP (DASH) are examples of available HAS technologies. In an HAS architecture, video content is stored on a server as segments of fixed duration at different quality levels. Each client can request the segment at the most appropriate quality level on the basis of the local perceived bandwidth. In this way, video playback dynamically changes according to the available resources, resulting in a smoother video streaming experience. The main disadvantage of current HAS solutions is that the heuristics used by clients to select the appropriate quality level underperforms in a multi-client scenario [Akhshabi et al. 2012a],[Li et al. 2014],[Petrangeli et al. 2014]. In a real scenario, multiple clients simultaneously request content from the HAS server. Often, clients have to share a single medium and issues concerning fairness among them appear, meaning that the presence of a client has a negative impact on the performance of others. As reported by Akhshabi et al. [2012a], fairness issues are not due to TCP dynamics, but mainly arise from the rate adaptation algorithms, as they decide on the actual rate to download. When multiple clients stream a video at the same time, a wrong bandwidth estimation can occur, due to the temporal overlap of the activity-inactivity periods of different clients. This wrong estimation subsequently affects the bit rate selection and thus the clients' Quality of Experience (QoE). This problem is aggravated by the uncoordinated nature of current HAS heuristics. This entails they are not aware of the presence of other clients nor can they adapt their behaviour to deal with it.

In this paper, we investigate the aforementioned problems arising in a multi-client setting. Particularly, we present a fair HAS client able to achieve smooth video playback, while coordinating with other clients in order to improve the fairness of the entire system. This goal is reached with the aid of an in-network-based system of *coordination proxies*, in charge of collecting measurements on the network conditions. This information is then used by the clients to refine their quality decision process and develop a fair behaviour.

The main contributions of this paper are three-fold. First, we present a new HAS heuristic called FINEAS (Fair In-Network Enhanced Adaptive Streaming) able to select the best quality to select depending on network conditions, in order to provide a smooth video streaming and improve fairness. Particularly, our heuristic is able to increase the average requested quality level compared to current HAS heuristics and avoid video freezes, while guaranteeing similar QoE to the all the clients streaming video, i.e. fairness. Second, we design an in-network-based system to help clients coordinate their behaviour, which does not require explicit client-to-client communication or a centralized decision process. Consequently, the quality level selection can still be performed locally and independently by each client, without any modification to the general HAS principle. Third, detailed simulation results are presented to characterize the gain of the proposed framework compared to state-of-the-art HAS heuristics.

The remainder of this article is structured as follows. Section 2 reports related work on HAS and multi-client algorithms. Section 3 details the proposed multi-client HAS framework, both from an architectural and algorithmic point of view. In Section 4, we evaluate our solution through simulation and show its effectiveness compared to current HAS heuristics. Section 5 presents the main conclusions. The online appendix presents an analytical formulation of the general fairness problem.

2. RELATED WORK

Akhshabi et al. [2012b] present an analysis of the performance and drawbacks of some commercially available HAS heuristics, such as Microsoft Smooth Streaming, Netflix and Adobe players. It is shown that current heuristics perform quality selection suboptimally. Particularly, they fail to adapt to rapid bandwidth changes, leading to drops in the client play-out buffer and unnecessary quality switches. They also analyse the performance of two competing HAS clients sharing the same bottleneck, and point out that they are not able to develop a fair behaviour. Similar considerations are reported by Müller et al. [2012a] when testing different HAS implementations using real bandwidth traces collected on a mobile network. They also point out that the Microsoft Smooth Streaming client is able to achieve the highest average bit rate as well as a low number of quality switches.

Many HAS rate adaptation heuristics have been proposed to alleviate the problems highlighted in the previous paragraph. Zhou et al. [2012] present a control theory-based HAS client where the buffer filling level of the client is controlled. A similar approach is also studied by Tian and Liu [2012]. Adzic et al. [2011] propose to add additional information into the video segments to enhance the quality decision algorithm. The client can then decide to switch up or down depending on the effect on bit rate and the intrinsic quality of the next segment to download. Jarnikov and Ozcelebi [2010] study an algorithm based on a Markov Decision Process (MDP), which requires offline training. Xiang et al. [2012] use a similar approach for Scalable Video Coding streaming and adopt a Markov model to describe the variations of wireless channel conditions. Also the work presented by Claeys et al. [2014] is based on a MDP and Reinforcement Learning theory. In this case, the solution to the MDP is computed online by means of the Q-Learning algorithm, without any *a priori* knowledge. All these algorithms share a common drawback. Although effective in a single-client scenario, they are not explicitly designed to deal with the presence of multiple clients streaming video simultaneously over a shared network medium. This means that they can fail in these circumstances, as shown by Akhshabi et al. [2012a]. They report that unfairness is mainly caused by the quality adaptation algorithms themselves, since they are not designed to explicitly cope with a multi-client scenario. It is shown that a relevant cause of unfairness is the temporal overlap of the activity-inactivity periods of different clients, since this can lead to wrong bandwidth estimations.

Based on this consideration, most of the algorithms designed to improve fairness in HAS focus on the modification of the time interval at which clients request a new segment. Villa et al. [2012] implement a modified version of the Microsoft ISS Smooth Streaming (MSS) client, randomizing the quality selection decision interval. The results show that this technique can improve fairness, but the randomization characteristics have to be selected carefully. A similar approach is also used by Jiang et al. [2014]. In this case, the next segment download is randomized taking into account the buffer status, in order to avoid freezes. Moreover, a stateful bit rate selection is introduced to allow clients with a low quality to increase it more aggressively. The authors theoretically prove that this selection allows convergence of the clients' bit rates. In the work by Li et al. [2014], the download of a chunk is scheduled to obtain a continuous average data rate sent over the network and to maintain the buffer level close to a given threshold. The segment fetch time is used by Liu et al. [2012] to decide which quality level to request. They try to improve fairness by allowing recently joined clients to behave more aggressively. De Cicco et al. [2013] propose the ELASTIC client, which avoids activity-inactivity periods by downloading segments in order to maintain the buffer level close to a set-point. The ELASTIC client is mainly designed to obtain fairness from the network point of view but not to maximize QoE, since quality switches are not included in the rate adaptation heuristic. This results in a high number of switches, even in a fixed bandwidth scenario, and consequently low QoE. The main problem of the presented algorithms is that they are purely client-based, i.e. no coordination is envisaged among the clients. Although this aspect simplifies the design and implementation of the algorithms, it does not allow solving the fairness problem completely. The lack of coordination mechanisms entails that the fairness problem has to be solved at design time and that the solution has to be encoded in the client's heuristic. Since fairness is inherently an online problem, which depends on the number of clients, network topology and bandwidth conditions, the obtained solution is sub-optimal. As an example, Müller et al. [2012b] pointed out that the network infrastructure, as a system of caching proxies, has a negative impact on the bandwidth estimation process of the clients, and thus also on their performance in terms of QoE and fairness. This type of information, which is not known at design time and thus not available to purely client-based algorithms, affects the effectiveness of the aforementioned solutions.

A first approach to tackle this problem would be a centralized solution, where the video server decides on the quality level to return. This solution is investigated by De Cicco et al. [2011] and Kuschnig et al. [2010]. De Cicco et al. propose a control theory-based algorithm to control the length of the sender buffer, placed at the server, in order to select the most appropriate quality level. In the work presented by Kuschnig et al., the server estimates the bandwidth of the TCP connection considering the number of bytes transmitted in a time unit. This estimation is performed on a per-Group Of Pictures basis. Even if a centralized approach would facilitate the computation of an optimal global fair policy, it is not scalable if the number of clients to serve increases. Moreover, a centralized approach alters the classical HAS principle.

In order to solve the scalability issue, we adopt in this paper an in-network approach. In in-network algorithms, intermediary nodes are placed in the network to collect information regarding the available bandwidth and influence the behaviour of HAS clients. An example of in-network adaptation is given by Bouten et al. [2012]. A system of network proxies periodically solves an optimization problem to determine the maximum quality the clients can download, based on the current network status and an objective function. Houdaille and Gouache [2012] introduce a bandwidth manager inside the home gateway to manage the flows belonging to the different clients. Based on clients' characteristics and network conditions, the bandwidth manager determines the target bit rate for each client, in order to fairly share the available bandwidth. The bandwidth manager then applies traffic shaping techniques to limit the bandwidth of each client and drive it to request the target bit rate. A similar approach is investigated by Georgopoulos et al. [2013]. A centralized OpenFlow controller is used to allocate the bandwidth for each streaming device, in order to obtain fairness from a QoE point of view. The authors present a model to correlate video bit rate with video quality, which is used by the controller in the bandwidth allocation process. Also Ma and Bartos [2011] implemented a bandwidth manager to determine the quality levels to assign to the clients, based on the available bandwidth and their subscription level. Mok et al. [2012] introduce a measurement proxy between the clients and the video streaming server, which estimates the available bandwidth and decides the highest possible quality level the clients can support. Based on this information and their buffer status, the clients decide the most appropriate quality level to request. El Essaili et al. [2013] propose a QoE optimizer for wireless networks that computes the optimal rate for the streaming clients, based on the wireless channel conditions. This value is then used by a QoE proxy, in charge of intercepting and rewriting clients' requests to match the requested quality level with the optimal rate. The principal disadvantage of the aforementioned algorithms is the active role of the in-network elements in the quality decision process. This aspect entails an alteration of the classical HAS principle, as the network *de-facto* decides which quality level the clients can download. Even if different clients will select quality differently, at steady-state all clients will converge to the quality level enforced by the network. Moreover, these solutions present robustness issues in case of fault or malfunctioning of the in-network elements.

Based on these considerations, we developed a new client-based rate adaptation heuristic to optimize the QoE delivered to the clients, together with an in-network coordination mechanism to improve fairness, which can operate in face of multiple bottleneck links. This approach presents two advantages. First, the in-network computation can be kept very simple and consequently not computationally demanding, since the quality decision algorithm still runs at the client. Second, it is more robust in case of fault or malfunctioning of the network equipment, as the client can continue to operate (at a sub-optimal level) without the in-network system. The concept of an in-network system to help clients obtaining fairness has been firstly introduced in our previous work [Petrangeli et al. 2014]. The difference with respect to the work presented in this paper is two-fold. First, the in-network system presented by Petrangeli et al. [2014] is composed of a single centralized node, which collects and aggregates statistics on clients' QoE performance. Even though the presented approach showed promising results, further experiments have shown unsatisfactory performance in complex scenarios, mainly due to: (i) the centralized nature of the solution, which is not able to handle the presence of multiple bottlenecks and (ii) the aggregation of QoE statistics, which is not trivial if clients are located in different sub-networks. In this work instead, we propose a distributed in-network system, able to cope with the presence of multiple bottlenecks, computing an estimate of the clients' fair bandwidth share on bottleneck links. This information is provided to the clients, which use it in their heuristics to achieve fairness. Second, we design a completely new rate adaptation heuristic, called FINEAS, to maximize client's QoE and exploit the information provided by the in-network system to achieve fairness.

3. PROPOSED FAIR HAS FRAMEWORK

The problem we propose to investigate in this paper is two-fold. First, clients have to obtain the highest possible video quality. Second, they have to show similar performance if they share bottleneck links, i.e. fairness. Based on this consideration, all the clients sharing the same bottlenecks should act fairly, even if they belong to different networks. An analytical formulation of the general fairness problem is presented in the online appendix. In order to maximize the QoE delivered to the clients and achieve fairness, we present the FINEAS (Fair In-Network Enhanced Adaptive Streaming) heuristic. The FINEAS heuristic runs at the clients and performs the quality level selection based on three inputs: the local perceived bandwidth, the video player buffer status and the so-called *fairness signal*. The fairness signal is an additional measure introduced to achieve fairness, obtained when the client downloads a segment. The fairness signal is computed by a system of network nodes, called *coordination proxies*, and represents an estimate of the fair bandwidth share of all the clients streaming video. In the remainder of this section we provide an architectural overview of the proposed framework (Section 3.1) and detail the implementation of the FINEAS heuristic (Section 3.2) and of the fairness signal computation (Section 3.3).

3.1. Architectural Overview

As introduced previously, the system of coordination proxies is in charge of helping clients achieving fairness, by computing an estimate of the fair bandwidth share for all the clients streaming video, even if they belong to different networks. In order to maintain scalability, the computation of the fairness signal is performed periodically and in a hierarchical way by the coordination proxies. A generic coordination proxy P receives an estimate of the fairness signal from its parent node and computes a new estimate of the fairness signal for each of its child proxies. This estimate is computed by monitoring the available bandwidth for HAS traffic on the links connecting proxy P to its child nodes. A root proxy triggers the computation of the fairness signal. At

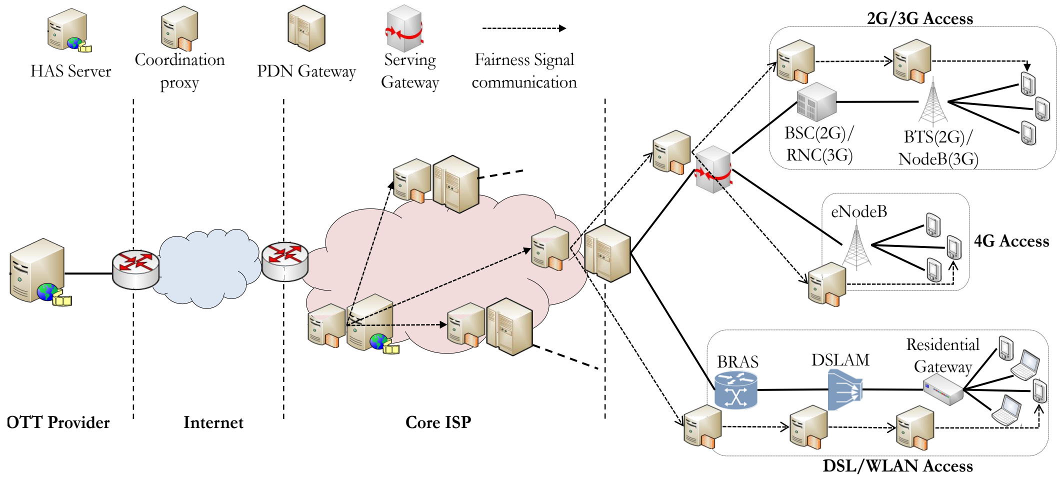


Fig. 1: Schematic representation of a communication network. The possible locations of the coordination proxies are shown. If the streaming service is offered by an Over-The-Top (OTT) provider, the proxy located at the HAS Server in the core ISP network has to be moved to the router connecting the core ISP network with the Internet.

initialization, the coordination proxies start monitoring the bandwidth for HAS traffic. After T_{fair} seconds, given T_{fair} is the fairness signal computation period, the root proxy forwards its first estimate of the signal and triggers its computation for the entire system. In order to limit overhead, the calculated fairness signal can be added as an HTTP header field and returned to the clients when delivering the next segment to play. Particularly, the clients translate the fairness signal into a *reference* quality level, representing the theoretical quality level to request in order to obtain perfect fairness among the clients. This reference gives an indication on the best quality level to achieve fairness, rather than determining the actual quality to be requested. The reason for this behaviour is two-fold. First, directly requesting the reference quality level would be optimal from the fairness point of view but not from the QoE point of view, because of the frequent switches that would occur. Second, directly requesting the reference quality level would alter the classical HAS principle, as the decision on the quality level to download would no longer be carried out by the clients.

The main advantage of this hybrid approach is three-fold. First, no communication is needed among clients and consequently no significant overhead is introduced. Moreover, no client-to-proxy communication is required. The proxies are *transparent* to the clients, as the clients only need to know how to access the fairness signal but not how it is created. Second, the computation and delivery of the fairness signal do not negatively affect the behaviour of existing clients. Third, the approach is robust toward proxy failure, as the clients can also operate without the fairness signal.

As far as the coordination proxies positioning is concerned, the proxies should be located at the main aggregation points of the network, in order to monitor the links where a bottleneck can occur. Potential bottlenecks can be identified by analysing the underlying network architecture or at runtime by monitoring link conditions (e.g., if the traffic exceeds a certain percentage of the link capacity, a coordination proxy can automatically become active). Since network operators have full control of their delivery infrastructure, they can easily identify which are the most sensible paths in their networks where a bottleneck could occur. This way, they can perform an initial placement of the coordination proxies on network nodes. Note that this assumption does not impact the flexibility of our solution, since in a real scenario the network architecture is given and does not significantly change over time. Furthermore, as coordination proxy functionalities can be implemented via software, proxies can be flexibly relocated in case network conditions consistently change over time. Moreover, coordination proxies can be placed liberally on network nodes, without negatively impacting the fairness signal computation even if a bottleneck does not occur. In this case, a proxy only receives the fairness signal from its parent node and forwards it to its child proxies, without performing any operation on it. In other words, if a bottleneck does not occur, the considered proxy becomes transparent with respect to the computation of the fairness signal. As an example of coordination proxies placement, Figure 1 depicts a schematic representation of a communication network architecture based on the Evolved Packet Core (EPC), as described by Release 11 of the 3rd Generation Partnership Project (3GPP) [3GPP 2013]. The EPC represents a unified core network for the convergence of both 3GPP (as 2G, 3G and 4G) and non-3GPP (as WLAN or DSL) access networks, but it is worth noting that our approach can be applied to other network topologies as well. Depending on whether the streaming service is offered by the network operator or an Over-The-Top (OTT) provider, the HAS server can be located inside the core ISP network or in a data center connected to the core ISP network through the Internet, respectively. In the latter case, a coordination proxy is located at the edge between the Internet and the core ISP network. In 2G and 3G access networks a base station and a base station controller manage the radio interface. In the 4G case these functionalities are carried out by the same node. A DSLAM concentrates the traffic coming from WLAN and DSL access networks and forwards it to the BRAS. A PDN Gateway aggregates the traffic belonging to 3GPP and non-3GPP access networks and directs it into the core ISP network. The aforementioned nodes can provide coordination proxy functionalities, since they aggregate traffic at different network levels and can detect the presence of congested links. It is worth noting that the proposed system can be extended to take into account the presence of caches. In this case, the proxies have to compute a separate fairness signal for each of the considered delivery nodes (e.g., a generic HAS server or a cache). Consequently, our approach can be combined with any existing HAS-optimised caching scheme.

An important aspect of our framework is how the bandwidth estimation process is carried out by the proxies. In this paper, we use an implementation based on sFlow, which performs bandwidth estimation through an operation of packet sampling [Schmidt et al. 2013]. Proxies monitor the bandwidth for two groups of flows: HAS traffic and non-HAS traffic (i.e. the traffic generated by non-HAS clients is grouped into the same flow). When a packet is sampled, the packet header is extracted and analysed in order to match it to a specific flow. A packet belongs to the HAS flow if the IP source address matches that of the HAS server and the TCP source port is equal to 80 (the standard port for HTTP connections). Otherwise, the packet is considered part of the non-HAS traffic flow.

3.2. Client-side Rate Adaptation

The FINEAS heuristic is executed by the client when a new segment has been downloaded, and before the request of a new one. The goal of the client is to select the most appropriate quality level in order to maximize QoE and achieve fairness. A detailed description of the heuristic is provided in Algorithm 1. We indicate all the parameters belonging to our heuristic with the symbol *. In Section 4, an extensive evaluation of the impact of these parameters on the algorithm's performance is reported. The FINEAS heuristic can be subdivided into four main parts we describe in the following.

Input collection (lines 1-15). First, the client obtains the local perceived bandwidth, the video player buffer filling level *bufferFillingLevel* and the buffer target

ALGORITHM 1: FINEAS client heuristic. The symbol * indicates the parameters belonging to the heuristic.

```
Data: videoBitRate, vector containing the bit rate of each quality level of the video
maxAvailableQuality, highest available quality level of the video
segmentDuration, duration of the video segments (in seconds)
bufferSize, size of the video player buffer, expressing the maximum amount of video that can be stored (in seconds)
qualityWindow*, time window stating how many quality level samples should be kept in memory (in seconds)
bufferMin*, panic threshold of the buffer level (in seconds)
bufferPercentage*, target percentage of the buffer size
α*, global utility function weight
Result: nextQL, the next quality level to request
$bandwidth \leftarrow$ getLocalPerceivedBandwidth();
$bufferFillingLevel \leftarrow$ getCurrentBufferFillingLevel();
$bufferTarget = bufferSize \times bufferPercentage$;
$qualityHistoryVector \leftarrow$ updateQualityHistoryVector($currentTime$,$qualityWindow$);
$averageQuality \leftarrow$ computeAverageQuality($qualityHistoryVector$);
$fairnessSignal \leftarrow$ getFairnessSignal();
$fairnessQuality = 1$;
for $ql \leftarrow 1$ to $maxAvailableQuality - 1$ do
    if $videoBitRate(ql) \leq fairnessSignal < videoBitRate(ql + 1)$ then
        $fairnessQuality = \frac{fairnessSignal + ql \times videoBitRate(ql+1) - (ql+1) \times videoBitRate(ql)}{videoBitRate(ql+1) - videoBitRate(ql)}$;
    end
end
if $fairnessSignal \geq videoBitRate(maxAvailableQuality)$ then
    $fairnessQuality = maxAvailableQuality$;
end
if $bufferFillingLevel \leq bufferMin$ then
    $nextQL = 1$;
else
    $maxDownloadableQuality = maxAvailableQuality$;
    for $ql \leftarrow 1$ to $maxAvailableQuality$ do
        $estimatedDownloadTime = (videoBitRate(ql) \times segmentDuration) / bandwidth$;
        $estimatedBufferLevel = bufferFillingLevel - estimatedDownloadTime + segmentDuration$;
        if $estimatedBufferLevel \leq bufferMin$ then
            $maxDownloadableQuality = ql - 1$;
            break;
        end
    end
    $globalUtilityMax = -1000$;
    for $ql \leftarrow 1$ to $maxDonwloadableQuality$ do
        $estimatedDownloadTime = (videoBitRate(ql) \times segmentDuration) / bandwidth$;
        $estimatedBufferLevel = bufferFillingLevel - estimatedDownloadTime + segmentDuration$;
        $qoeUtility = - \mid ql - maxDownloadableQuality \mid - \mid ql - averageQuality \mid$
        $- \mid estimatedBufferLevel - bufferTarget \mid$;
        $fairnessUtility = - \mid ql - fairnessQuality \mid$;
        $globalUtility = (1 - \alpha) \times fairnessUtility + \alpha \times qoeUtility$;
        if $globalUtility \geq globalUtilityMax$ then
            $globalUtilityMax = globalUtility$;
            $nextQL = ql$;
        end
    end
end
return $nextQL$;
```

bufferTarget (lines 1-3). The perceived bandwidth is the bandwidth measured during the download of a segment and is computed as the ratio between the segment size and the segment download time. *bufferFillingLevel* is the amount of video currently stored in the video player buffer, expressed in seconds. It depends on the number of segments currently stored in the buffer and their duration. *bufferTarget* represents the target for the buffer filling level, expressed in seconds. The client requests the next segment in order to maintain the resulting buffer filling level close to *bufferTarget*. This value is set by the parameter *bufferPercentage*, which is expressed as a percentage of the total buffer size. As an example, given *bufferSize* and *bufferPercent-*

age are equal to 10 seconds and 80% respectively, the desired buffer target would be equal to $10 \times 0.8 = 8$ seconds. Next, the client updates the vector *qualityHistoryVector*, which contains the quality levels requested in the last *qualityWindow* seconds, and computes the average requested quality over this time window (lines 4-5). The fairness signal is then extracted from the HTTP header of the downloaded segment (line 6). The fairness signal represents the fair bandwidth share per client as computed by the coordination proxies. A client should request the quality level whose bit rate is closest to the fairness signal, in order to fairly share resources with the other clients. Consequently, a reference quality level *fairnessQuality* is obtained starting from the *fairnessSignal* and the bit rates of the video (lines 7-15). *fairnessQuality* is then used as the target quality level to request in order to obtain perfect fairness among clients. Note that this mapping produces a real value in the interval [*1;maxAvailableQuality*], since in general there is not an exact match between the fairness signal and the bit rates of the video. The client identifies the nearest lower (*ql*) and nearest higher (*ql+1*) bit rates compared to *fairnessSignal* (line 9). *fairnessQuality* is computed on line 10. This operation corresponds to finding the fitting line passing through the points *a=(ql,videoBitRate(ql))* and *b=(ql+1,videoBitRate(ql+1))*, and using the obtained equation to find the coordinate of point *c=(fairnessQuality, fairnessSignal)*, where the unknown variable is *fairnessQuality*. Particularly, by substituting points a and b into the equation *y=mx+k* and solving it with respect to *m* and *k*, the fitting line can be found. *m* and *k* are equal to $videoBitRate(ql{+}1){-}videoBitRate(ql)$ and $(ql+1) \times videoBitRate(ql) - ql \times videoBitRate(ql+1)$, respectively. By substituting the coordinate of point *c* into the aforementioned equation and solving it with respect to *fairnessQuality*, the formula expressed on line 10 is obtained. If *fairnessSignal* is greater than the highest available bit rate, *fairnessQuality* is assigned to the highest available quality level (lines 13-14).

Buffer status control (lines 16-17). If the buffer level *bufferFillingLevel* is below the panic threshold *bufferMin*, the lowest quality level is directly requested. This way, the client tries to timely react to a situation where the risk of video freezes is high.

Highest downloadable quality level identification (lines 19-27). Otherwise, the heuristic identifies the highest quality level *maxDownloadableQuality* that can actually be downloaded. For each of the available quality levels, the client computes an estimate of the download time (line 21). We refer here to an estimate as the computation is performed using the local perceived bandwidth, which could change in the future. The download time is used to compute an estimate of the buffer level when the download of the next segment to request will be completed, called *estimatedBufferLevel* (line 22). The client identifies the first quality level *ql* such that *estimatedBufferLevel* is lower than the panic threshold *bufferMin* (line 23). This means that if *ql* or a higher quality level is requested, the buffer level is likely to be below the panic threshold at the end of the segment download. In order to avoid a possible freeze or the request of the lowest quality level at the next step, the highest quality level that can be downloaded (i.e. *maxDownloadableQuality*) is then assigned to *ql*-1 (line 24).

Final quality level selection (lines 28-39). The client finally requests the quality level maximizing a global utility, given by the weighted sum of a QoE and fairness term. The QoE term (line 32) is an estimation of the actual QoE as experienced by the user, comprising (i) segment quality, (ii) quality switching and (iii) freezes. Segment quality is maximized by requesting the highest downloadable quality level (i.e. $-|$*ql–maxDownloadableQuality*$|$), switching is minimized by reducing the deviation from the average requested quality within the quality window (i.e. $-|$*ql–averageQuality*$|$), and freezes are minimized by keeping the buffer level as close to the buffer target level as possible (i.e. $-|$*estimatedBufferLevel–bufferTarget*$|$). The fairness term is maximized by requesting the quality level closest to the *fairnessQuality* (line 33). This way, the

ALGORITHM 2: In-network computation performed at the coordination proxy P

```
Data: childProxies, vector containing the child coordination proxies of proxy $P$
controlledClients, vector containing the number of HAS clients controlled by each proxy in childProxies
Result: fairnessSignal, vector containing the fairness signals to forward to the proxies in childProxies
1  $parentFairnessSignal \leftarrow$ getParentFairnessSignal();
2  $unusedBandwidth = 0,\ entitledClients = 0$;
3  for i in childProxies do
4      $estimatedBandwidth \leftarrow$ getEstimatedBandwidthForHASTrafficTowardsChildProxy($i$);
5      $maxFairnessSignal(i) = estimatedBandwidth/controlledClients(i)$;
6      if $maxFairnessSignal(i) \leq parentFairnessSignal$ then
7          $unusedBandwidth{+}{=}(parentFairnessSignal - maxFairnessSignal(i)) \times controlledClients(i)$;
8      else
9          $entitledClients{+}{=}controlledClients(i)$;
10     end
11 end
12 $childProxies \leftarrow$ sortIncreasing($childProxies, maxFairnessSignal$); ##sort proxies in childProxies in increasing
   order of associated maxFairnessSignal##
13 for i in childProxies do
14     if $maxFairnessSignal(i) \leq parentFairnessSignal$ then
15         $fairnessSignal(i) = maxFairnessSignal(i)$;
16     else
17         $bandwidthPerClient = unusedBandwidth/entitledClients$;
18         $fairnessSignal(i) = \min(parentFairnessSignal + bandwidthPerClient, maxFairnessSignal(i))$;
19         $unusedBandwidth{-}{=}(fairnessSignal(i) - parentFairnessSignal) \times controlledClients(i)$;
20         $entitledClients{-}{=}controlledClients(i)$;
21     end
22 end
23 return $fairnessSignal$;
```

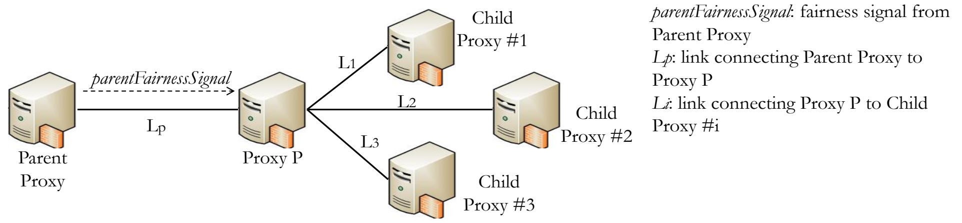

Parent Proxy
parentFairnessSignal
Lp
Proxy P
L1
L2
L3
Child Proxy #1
Child Proxy #2
Child Proxy #3
parentFairnessSignal: fairness signal from Parent Proxy
Lp: link connecting Parent Proxy to Proxy P
Li: link connecting Proxy P to Child Proxy #i

Fig. 2: Schematic representation of the coordination proxy architecture

client is driven to request the best quality level to fairly share bandwidth with the other clients. For each quality from 1 to *maxDownloadableQuality*, the global utility is computed as the linear combination of the QoE and fairness terms (line 34). The selected quality level is the quality that maximizes this global utility (lines 35-38). A parameter α weighs the importance of the QoE and fairness components on the global utility. For $\alpha \rightarrow 1$, the heuristic tends to only maximize the QoE of the client, without taking into account fairness. For $\alpha \rightarrow 0$, only fairness is optimized. The influence of this parameter is investigated in Section 4.3.

The computational complexity of our heuristic can be estimated as $O(n)$, with n the number of quality levels, given the complexity of the loops on lines 8, 20 and 29 is $O(n)$. The algorithm has been implemented on a Dell Latitude E5530 running Ubuntu 12.04 LTS 64-bit, Intel Core i5-3320M CPU @ 2.60GHz processor and 8 GB of memory to evaluate scalability. Even with 10000 available quality levels, the quality selection can still be performed in less than 6 milliseconds.

3.3. Fairness Signal Computation

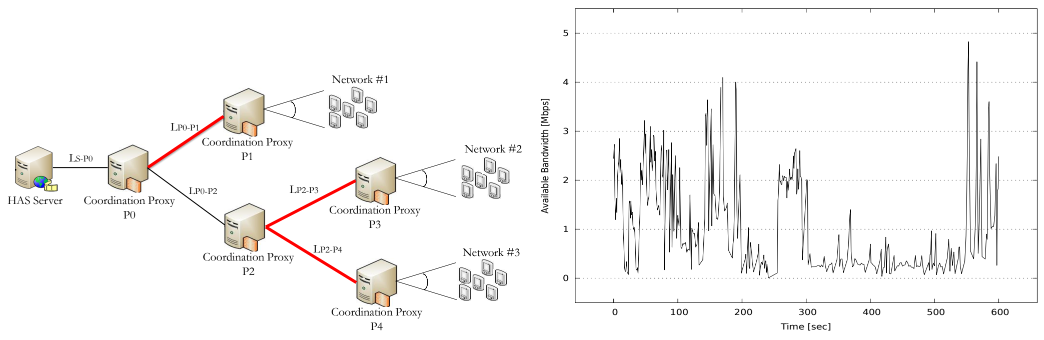
The hierarchical system of coordination proxies has the fundamental role to help the clients in achieving fairness, by periodically computing the fairness signal. This value represents the theoretical fair bandwidth allocation of the clients belonging to a certain network and is computed taking into account all the clients streaming video (even those belonging to other networks). Algorithm 2 describes the operations performed by a generic coordination proxy P to compute its estimate of the fairness signal.

As depicted in Figure 2, proxy P is connected to a parent proxy and to a set of child proxies, each controlling a certain number of clients as indicated in the vector *controlledClients*. From the parent node, P receives the fairness signal *parentFairnessSignal* (line 1). This value represents the fair bandwidth share for each client controlled by P, as computed by the parent node. Proxy P aims to fairly redistribute the bandwidth represented by *parentFairnessSignal* to its child proxies. In case P is the root of the system, *parentFairnessSignal* has to be assigned to infinite, as we assume there are no bottlenecks before the root proxy.

For each child proxy i, P obtains the estimated bandwidth for HAS traffic on the link connecting P to proxy i (link L_i in Figure 2). Next, it computes the fair bandwidth share on this link, i.e. the ratio between the estimated bandwidth on L_i and the number of clients controlled by proxy i (line 5). This value, indicated as *maxFairnessSignal(i)*, represents the maximum fairness signal achievable for proxy i. Proxy P then checks whether L_i or L_P, the link connecting the parent proxy to P, is the actual bottleneck for proxy i. Particularly, if the fair bandwidth per client on link L_i (i.e. *maxFairnessSignal(i)*) is smaller than that on link L_p (i.e. *parentFairnessSignal*), the bottleneck is represented by link L_i. Otherwise, if *maxFairnessSignal(i)>parentFairnessSignal*, L_p acts as bottleneck for proxy i. When L_i represents the bottleneck (line 6), the clients controlled by proxy i are not able to use all the bandwidth indicated by *parentFairnessSignal*. Particularly, the unused bandwidth is equal to the difference between *parentFairnessSignal* and *maxFairnessSignal(i)*, multiplied by the number of clients controlled by proxy i (line 7). The leftover bandwidth *unusedBandwidth* can be redistributed to the clients whose bottleneck is represented by link L_p. When L_p is the bottleneck, the number of clients entitled to accept part of the exceeding bandwidth is incremented by the number of clients controlled by proxy i (line 9).

The final fairness signal for each child proxy is computed on lines 13-22. Here, proxy P redistributes the exceeding bandwidth to all the entitled clients. If L_i is the bottleneck for child proxy i, i.e. no extra bandwidth can be assigned, the final fairness signal is assigned to *maxFairnessSignal(i)* (lines 14-15). Otherwise, the surplus bandwidth per client *bandwidthPerClient* is computed as the ratio between the unused bandwidth and the number of entitled clients (line 17). Next, the final fairness signal for proxy i is assigned as the sum between the parent fairness signal and the surplus bandwidth per client (line 18). The *min* operation performed on line 18 avoids that *fairnessSignal(i)* exceeds *maxFairnessSignal(i)*. The amount of unused bandwidth is then decremented by the amount assigned at the previous step (line 19). Also, the amount of entitled clients is decremented by the number of clients controlled by proxy i (line 20).

As an example, we consider the scenario in Figure 2, where proxy P is connected to three child proxies. We assume each child proxy controls 10 HAS clients (i.e. *controlledClients(i)*=10 for i=1,2,3), and the estimated bandwidth for HAS traffic on links L_1, L_2 and L_3 is 10, 20 and 35 Mbps, respectively. *parentFairnessSignal* is 2 Mbps. In this case, the variable *maxFairnessSignal(i)* for each proxy is equal to 1, 2 and 3.5 Mbps, respectively. This entails that L_1 represents the main bottleneck for proxy #1, as *maxFairnessSignal(1)*$\leq$*parentFairnessSignal*. The unused bandwidth is computed as on line 7 and is equal to *unusedBandwidth*=10 Mbps. This is not the case for proxy #3, as *maxFairnessSignal(3)* is greater than *parentFairnessSignal*. Consequently, the number of entitled clients is incremented by 10 clients. The fairness signal for proxy #1 and #2 is assigned directly to 1 and 2 Mbps, respectively, as both *maxFairnessSignal(1)* and *maxFairnessSignal(2)* are smaller or equal to *parentFairnessSignal*. On the contrary, proxy P can assign the exceeding bandwidth to the clients controlled by proxy



(a) (b)

Fig. 3: Simulated topology (a) and an extract of the used bandwidth pattern (b).

#3. The variable *bandwidthPerClient* is equal to 1 Mbps and the final fairness signal is equal to *parentFairnessSignal*+*bandwidthPerClient*=3 Mbps, as shown on line 18.

The rationale behind the sort operation performed on line 12 is to reduce the computational complexity of the heuristic. The complexity of the two *for* loops on lines 3 and 13 is $O(n)$, while that of the sort operation is $O\left(n \times log(n)\right)$, with n the length of vector *childProxies*. Consequently, the total computational complexity is $O\left(n \times log(n)\right)$, which is known to scale well even for large values of n. If the vector *childProxies* is not sorted, the computation on lines 13-22 has to be performed using two nested *for* loops, for a total computational complexity of $O(n^2)$. In order to evaluate complexity, Algorithm 2 has been implemented on a Dell Latitude E5530 running Ubuntu 12.04 LTS 64-bit, Intel Core i5-3320M CPU @ 2.60GHz processor and 8 GB of memory. We incremented the number of child coordination proxies from 2 to 10000 and measured the time needed to compute the fairness signal. As expected, the computation time increases logarithmically with the number of coordination proxies. Even for 10000 child proxies, the fairness signal computation is carried out in less than 0.032 seconds on a general purpose device. Moreover, it is worth noting that the fairness signal computation is decoupled from the actual segment delivery to the clients. When a proxy receives a segment to forward, it just has to embed its current estimate of the fairness signal into an HTTP header. The actual update of the fairness signal can be considered as an independent and parallel process occurring periodically within the system of coordination proxies.

The fairness signal computation period, i.e. the communication interval among the proxies, is an important parameter of our algorithm. A small period entails that the fairness signal closely follows bandwidth variations, at the price of an increased computational and communication overhead. On the contrary, a high period reduces overhead but negatively affects the significance of the fairness signal. The impact of this parameter on the performance of our solution is evaluated in Section 4.

4. PERFORMANCE EVALUATION

4.1. Experimental Setup

A NS-3-based simulation framework has been used to evaluate our multi-client framework [Bouten et al. 2012]. The video streamed is *Big Buck Bunny*, composed by 299 segments, each 2 seconds long and encoded at 7 different quality levels: 300, 427, 608, 806, 1233, 1636, 2436 kbps. The Network Simulation Cradle[1] has been enabled for all

[1]http://research.wand.net.nz/software/nsc.php

the results, in order to provide a realistic implementation of the TCP protocol. Unless otherwise stated, the buffer size for each client is equal to 5 segments, or 10 seconds.

The simulated network topology is shown in Fig. 3a, where the position of the coordination proxies is reported. The root proxy is co-located with the HAS Server (not shown in Fig. 3a). In order to give an extensive evaluation of the FINEAS heuristic, we simulate 50 episodes of the video trace and average the results over the 50 runs. The capacity on links $L_{S\text{-}P0}$ and $L_{P0\text{-}P2}$ is kept fixed for all the experiments. A cross traffic generator introduces UDP traffic on links $L_{P0\text{-}P1}$, $L_{P2\text{-}P3}$ and $L_{P2\text{-}P4}$ (highlighted links in Figure 3a), in order to vary the available bandwidth for HAS traffic. A different cross traffic pattern is used for each episode, which varies each second and is scaled with respect to the number of clients. An episode is defined as a single simulation run. As far as the cross traffic pattern is concerned, we use an open-source dataset collected on a real 3G/HSDPA network [Riiser et al. 2012]. The available bandwidth for one client fluctuates between 202 bps and 6335 kbps, with an average of 2087 kbps and a standard deviation of 1314 kbps. An example of the bandwidth pattern is shown in Figure 3b. Regarding the bandwidth estimation at the coordination proxies, the sFlow packet sampling rate has been set to 1 packet sampled every 1000 for all the experiments. As far as the fairness signal is concerned, it is added as an HTTP header field by the proxies and returned to the clients when delivering the next segment.

In order to provide an extensive benchmark of the FINEAS algorithm, we compare our results to those obtained using four other HAS clients. Particularly, we choose a proprietary HAS client, the MSS[2] client, the Q-Learning-based client described by Claeys et al. [2014] and the client developed by Miller et al. [2012]. We also studied the performance of the FESTIVE algorithm, one of the first algorithms developed to explicitly deal with a multi-client scenario [Jiang et al. 2014]. As far as the performance evaluation is concerned, fairness is computed as the standard deviation of clients' QoE. Unless otherwise stated, all the simulated clients start streaming the video at the same time.

4.2. QoE Model

In this section, we define the QoE model used to evaluate the proposed framework. We use a metric in the same range of the Mean Opinion Score (MOS), that can be computed as in Eq. 1 [De Vriendt et al. 2013], [Claeys et al. 2014]:

$$QoE_j^i = 5.67 \times \frac{\overline{q}_j^i}{{q_{max}}_j^i} - 6.72 \times \frac{\hat{q}_j^i}{{q_{max}}_j^i} + 0.17 - 4.95 \times F_j^i \tag{1}$$

The QoE experienced by client j in network i is the linear combination of the average quality level requested $\overline{q}_j^i$, its standard deviation $\hat{q}_j^i$ (both normalized with respect to the highest available quality level ${q_{max}}_j^i$) and F_j^i, which models the influence of freezes and is computed as follows:

$$F_j^i = \frac{7}{8} \times max\left(\frac{ln(\phi_j^i)}{6} + 1, 0\right) + \frac{1}{8} \times \left(\frac{min(\psi_j^i, 15)}{15}\right) \tag{2}$$

ϕ_j^i and ψ_j^i are the freezes frequency and the average freeze duration, respectively. All the coefficients reported in Eq. 1-2 have been fixed according to the works by De Vriendt et al. [2013] and Claeys et al. [2014].

4.3. Parameter Analysis

In this section we investigate the impact of the parameters characterizing our HAS framework on clients' performance, and select the configuration to use in the remain-

[2]https://slextensions.svn.codeplex.com/svn/trunk/SLExtensions/AdaptiveStreaming

Table I: Overview of evaluated parameter configuration

Parameter	*Evaluated values*	*Parameter*	*Evaluated values*
qualityWindow [sec]	10, 20, 30, 50, 70	bufferMin [sec]	2, 4, 6
bufferPercentage	0.4, 0.6, 0.8, 1.0	α	0, 0.2, 0.4, 0.6, 0.8, 1.0
T_{fair} [sec]	2, 4, 6		

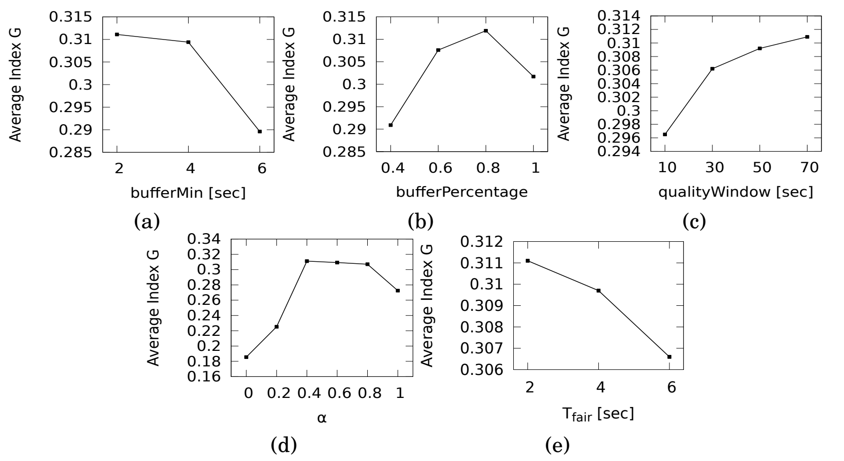


Fig. 4: Analysis of the parameter influence

der of the paper. As explained in Section 3.2 and Algorithm 1, the FINEAS heuristic parameters are: *qualityWindow*, stating how many quality levels have to be kept in memory, *bufferMin*, the panic buffer threshold, *bufferPercentage*, to tune the target buffer level, and α, the global utility function weight. As far as the in-network coordination is concerned (see Section 3.3), the only parameter to set is the fair signal computation period, named T_{fair}. In order to properly tune our HAS framework and select the best configuration, an elaborate evaluation of the parameter space has been performed. The evaluated values are reported in Table I, for a total of 648 feasible configurations (note that the target buffer level needs always to be higher than the panic buffer threshold). We evaluate our solution with the setting reported in Section 4.1; each network containing 10 clients streaming video at the same time. The capacity of links $L_{S\text{-}P0}$ and $L_{P0\text{-}P2}$ is set to 60 Mbps and 40 Mbps respectively, while the bandwidth on links $L_{P0\text{-}P1}$, $L_{P2\text{-}P3}$ and $L_{P2\text{-}P4}$ is variable and has an average of 20 Mbps and a standard deviation of 13 Mbps over the 50 simulated episodes. This simulation setup allows us to have a clear understanding of the impact of the different parameters on the performance of our solution. Depending on the available bandwidth on links $L_{P0\text{-}P1}$, $L_{P2\text{-}P3}$ and $L_{P2\text{-}P4}$, the actual bottlenecks for the three networks dynamically change. This way, we can explore a wide range of network configurations, where the three networks mutually influence each other (due to the bandwidth limitations on links $L_{S\text{-}P0}$ and $L_{P0\text{-}P2}$) or not. Other topologies have been investigated, but the results do not significantly change and are omitted due to space constraints.

The performance evaluation is conducted on the basis of the achieved QoE and fairness. Particularly, we express fairness as the standard deviation of clients' QoE. We introduce a metric G_k to evaluate the overall performance of the analysed configurations, defined as in Eq. 3:

$$G_k = \frac{1}{N}\sum_{i=1}^{N} J_k^i - \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(J_k^i - \frac{1}{N}\sum_{i=1}^{N} J_k^i\right)^2} \quad \text{with} \quad J_k^i = \overline{QoE}_k^i - \widehat{QoE}_k^i \tag{3}$$

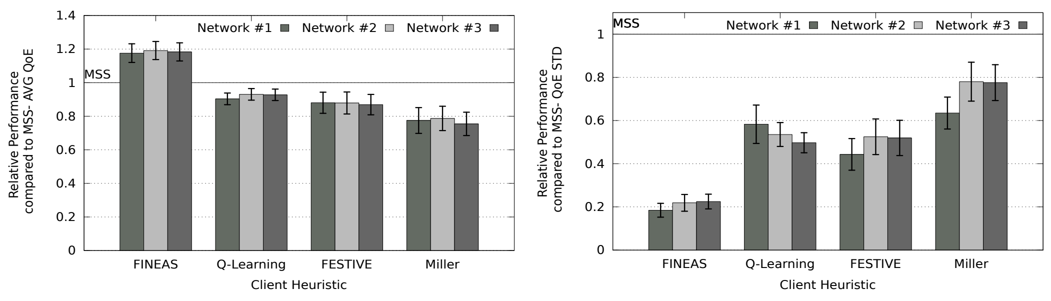
N represents the number of networks containing HAS clients streaming video. J_k^i represents the performance of network i during the *k-th* episode and is the linear combination of $\overline{QoE}_k^i$, the average QoE computed over the whole group of clients belonging to network i for episode k, and $\widehat{QoE}_k^i$, its standard deviation. This way, J_k^i encodes the fairness objective for network i, i.e. maximizing QoE while improving fairness. G_k is used to evaluate the overall performance of the analysed configuration and is given by the linear combination of the average J_k^i over all networks and of the negative of its standard deviation. Our aim is to select a configuration that allows to maximize G_k, in order to achieve the highest possible performance for every network while keeping the deviation among them as low as possible. For every configuration, the index G has been computed as the average of G_k over the 50 simulated episodes. For each evaluated value of each parameter, the average G of the five best configurations containing the evaluated value has been calculated. Using more than 5 values does not significantly affect the outcome of this analysis. The results are shown in Figure 4. Small values of the panic threshold *bufferMin* lead to better performance, as they avoid to often request the lowest quality level. The increased risk for video freezes caused by a low panic threshold is counterbalanced by using higher values of the buffer percentage *bufferPercentage*, which led to higher buffer targets. A trade-off is present in this case: a high percentage reduces the risk of freezes but also the possibility to request a high quality level. The best performance is reached for a value of 0.8. As far as the *qualityWindow* is concerned, it appears clearly that the more quality levels stored, the better. This entails that the client has a more comprehensive vision of the actions taken in the past, and can choose the next quality level accordingly. The parameter α states to what extent the clients follow the indication given by the fairness signal. The best value is obtained at 0.4, meaning that the two opposite objectives of maximizing clients' QoE and achieving fairness are well-balanced. Strictly following the fairness signal (i.e. values of α close to zero) leads to perfect fairness but bad performance in terms of QoE, because of the frequent switches that would occur. This result also highlights that it is more convenient to incorporate the fairness signal into the adaptation heuristic rather than using it directly to enforce the quality selection process. As expected, a small value of the fairness signal computation period T_{fair} leads to the best performance, as the in-network computation can quickly follow the bandwidth variations. The introduced communication overhead can be kept minimal by adding an HTTP header to transport the fairness signal.

Based on this analysis, the configuration with *qualityWindow* equal to 70 seconds, *bufferMin* to 2 seconds, *bufferPercentage* to 0.8, α to 0.4 and T_{fair} to 2 seconds has finally been chosen. It is worth noting that this configuration is also the best overall, thus showing the validity of the parameter analysis.

4.4. Variable Bandwidth Scenario

After the parameter analysis, we evaluate our solution with an increased number of clients. A fixed bandwidth scenario has also been evaluated, but the results are omitted due to space limits. In a fixed bandwidth scenario, all the simulated heuristics are able to guarantee a high QoE, but only the FINEAS heuristic is able to guarantee fairness.

The same simulation setting as in Section 4.1 is used, with 30 clients per network streaming a video at the same time. In this case, links $L_{S\text{-}P0}$ and $L_{P0\text{-}P2}$ are fixed to 180 Mbps and 120 Mbps respectively, while the bandwidth on links $L_{P0\text{-}P1}$, $L_{P2\text{-}P3}$ and $L_{P2\text{-}P4}$ is variable and presents an average of 60 Mbps and a standard deviation of 40 Mbps over the 50 simulated episodes. In this scenario, the three networks can mutually influence their performance, as the links $L_{S\text{-}P0}$ and $L_{P0\text{-}P2}$ act as common bottlenecks. It is fundamental to stress that the variable bandwidth pattern has been collected on a



(a) (b)

Fig. 5: Comparison between the different clients, from a QoE perspective, for a variable bandwidth scenario. Each network contains 30 clients streaming video. The graphs report the relative performance of the considered clients in terms of (a) average QoE and (b) its standard deviation compared to MSS. The standard deviation of clients' QoE is used as fairness metric.

Table II: Performance summary in the variable bandwidth scenario, in terms of quality components. The average value over the 50 episodes is reported, together with the confidence interval at 95%.

		MSS	*FINEAS*	*Q-Learning*	*FESTIVE*	*Miller*
Network 1	QoE	2.96±.18	3.41±.15	2.61±.07	2.64±.26	2.35±.32
	$\bar{q}_j^i$	4.51±.19	5.19±.13	3.69±.03	5.60±.14	5.70±.15
	$\hat{q}_j^i$	0.91±.08	0.98±.06	0.55±.03	1.23±.04	1.51±.09
	Bitrate [Mbps]	1.01±.08	1.30±.05	0.74±.01	1.47±.05	1.51±.06
	Freeze [sec]	0.08±.02	0.07±.02	0.07±.04	4.21±1.16	3.19±.59
	Freeze number	0.07±.01	0.05±.02	0.06±.03	3.42±.84	3.40±.49
Network 2	QoE	2.76±.15	3.25±.16	2.52±.08	2.44±.23	2.18±.24
	$\bar{q}_j^i$	4.21±.16	5.16±.15	3.68±.04	5.14±.11	5.24±.12
	$\hat{q}_j^i$	0.83±.07	1.12±.06	0.62±.04	1.14±.03	1.38±.06
	Bitrate [Mbps]	0.89±.06	1.29±.06	0.73±.01	1.28±.04	1.32±.05
	Freeze [sec]	0.08±.02	0.09±.03	0.10±.03	3.31±.87	2.66±.48
	Freeze number	0.08±.01	0.07±.02	0.11±.02	2.95±.71	2.99±.39
Network 3	QoE	2.78±.15	3.25±.16	2.52±.07	2.44±.22	2.11±.24
	$\bar{q}_j^i$	4.21±.16	5.16±.15	3.63±.04	5.14±.11	5.23±.12
	$\hat{q}_j^i$	0.81±.07	1.12±.06	0.57±.03	1.14±.03	1.40±.05
	Bitrate [Mbps]	0.89±.06	1.29±.06	0.73±.01	1.28±.04	1.31±.05
	Freeze [sec]	0.08±.03	0.09±.03	0.11±.03	3.35±.89	2.86±.51
	Freeze number	0.07±.01	0.07±.02	0.11±.02	2.91±.69	3.15±.39

Table III: Statistical significance of the average QoE and the average standard deviation of the QoE, using a two-tail paired t-testing with significance level 0.05.

	MSS	*FINEAS*	*Q-Learning*	*FESTIVE*	*Miller*
Average QoE	2.78±.50a	3.24±.55b	2.50±.38c	2.46±.58c	2.17±.60d
Average QoE Standard Deviation	0.69±.16a	0.13±.04b	0.32±.05c	0.31±.08c	0.46±.09d

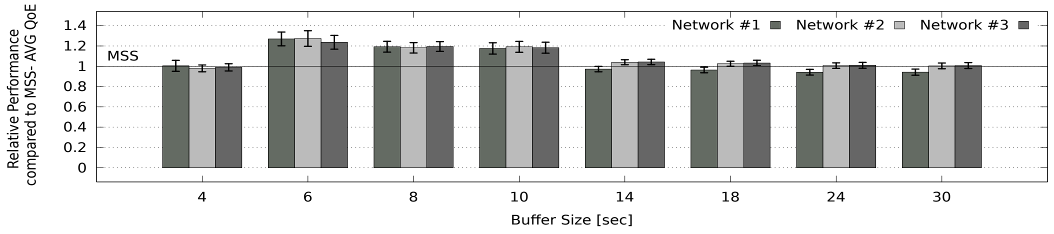
real 3G/HSDPA network, as described in Section 4.1. We consider the MSS as reference client and compute, for each simulated episode and for each network, the ratio between the average QoE of the analysed client and that of the MSS. Figure 5a reports the average value over the 50 episodes of this ratio, together with the confidence intervals at 95%. Figure 5b reports the average value over the 50 episodes of the ratio between the QoE standard deviation of the MSS algorithm and that of the analysed client, together with the confidence intervals at 95%. The QoE standard deviation is used as fairness metric. Our solution is able to increase the average QoE by almost 20% for each of the three networks and to improve fairness with almost 80% when compared to MSS. Also the FESTIVE and Miller clients improve fairness, but consistently reduce the average QoE. This entails that the final QoE at the clients with these two heuristics is lower than that obtained using the MSS heuristic. These results show the sub-optimality of these two heuristics in case of a variable bandwidth, caused by frequent quality switches and video freezes. As far as the Q-Learning client is concerned, it improves fairness by about 50% with respect to MSS, but with a loss of 8-10% in terms of average QoE. This negative behaviour is mainly due to the mutual influence among the learning processes of the clients and the uncoordinated nature of Q-Learning [Claus and Boutilier 1998]. When a client selects a certain quality level, it uses a portion of the shared bandwidth. This decision has an impact on the performance of the other clients and thus also on their learning process. Since the clients do not share any information, this leads to a sub-optimal quality adaptation policy.

Table II summarizes the results for this scenario. We compute, for each episode and for each network, the average of clients' QoE, $\bar{q}_j^i$, $\hat{q}_j^i$, requested bitrate, freeze time and freeze number. $\bar{q}_j^i$ and $\hat{q}_j^i$ represent, respectively, the average quality level requested by client j in network i and the standard deviation from this average (see Eq. 1). We then average these values over the 50 simulated episodes. The largest gain of the FINEAS heuristic is an increased average quality level (15% higher than MSS) and requested bitrate, without affecting the freeze duration. The FESTIVE and Miller clients reach the highest quality level, but the freeze duration, the number of freezes and quality switches increase. This means that the rate adaptation process is not performed optimally and, consequently, does not lead to the highest possible QoE. The main problem of the Q-Learning algorithm is the low average requested quality level, which strongly affects the final QoE. This behaviour is a consequence of the mutual influence among the clients. When a client requests a high quality segment, it could experience a freeze due to the congestion caused by the other clients. This way, the clients develop a more conservative policy, where high quality levels are avoided to limit video freezes. This problem is further intensified by the non-stationarity of the simulated environment.

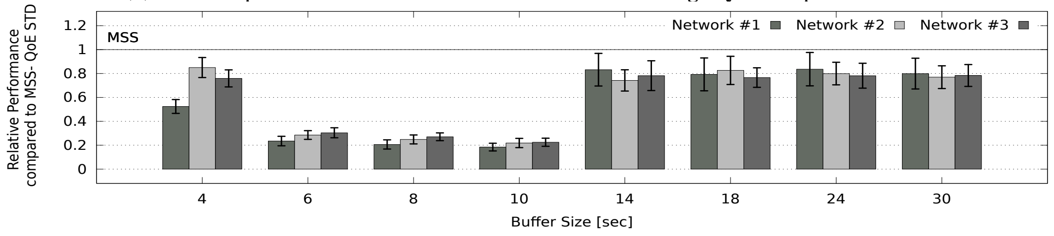
Table III reports the statistical significance of the results. For each algorithm and for each episode, we compute the average QoE and the average standard deviation over the three networks. Next, we perform a two-tail paired t-test for each pair of heuristics. Results are reported as the mean (and its standard deviation) of the average QoE and the average QoE standard deviation, over the 50 simulated episodes. Taken two heuristics, they are statistically different if they are associated to different letters. For example, the results for the MSS and FINEAS heuristics are statistically different. Only the Q-Learning and the FESTIVE clients do not show any significant difference in performance.

4.5. Influence of the Buffer Size

In this section, we investigate clients' performance using different buffer sizes. Based on the results of the previous section, we decided to evaluate our solution and the MSS one, which outperforms the Q-Learning, the FESTIVE and the Miller client in terms of QoE (see Figure 5a). The same simulation setting of the previous section has been used, with different buffer sizes from 4 to 30 seconds. In the 4 seconds case, the panic threshold of the FINEAS heuristic has been disabled (i.e. *bufferMin* set to zero). The results of this investigation are depicted in Figure 6. As expected, when the buffer size increases, the differences between our solution and MSS decrease. Particularly, the MSS heuristic presents a big performance gain in terms of average QoE and its standard deviation when the buffer is larger than 14 seconds. Consequently, the gain brought by the FINEAS algorithm decreases. Nevertheless, the FINEAS heuristic is still able to outperform MSS by about 20% in terms of fairness. A large buffer simplifies the rate adaptation process, since it reduces the risk of freezes and thus the

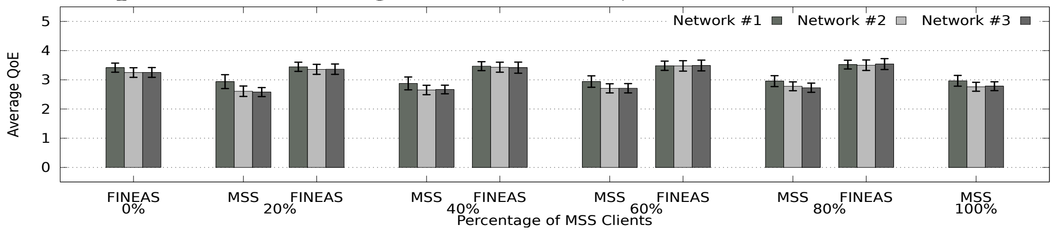


(a) Relative performance of our solution in terms of average QoE compared to MSS.

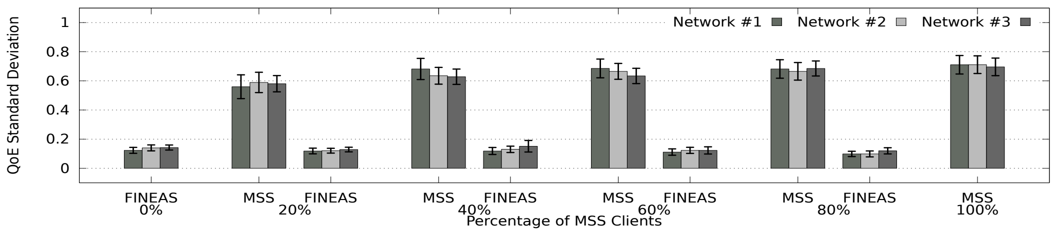


(b) Relative performance of our solution in terms of QoE standard deviaton compared to MSS.

Fig. 6: Evaluation of the FINEAS heuristic, from a QoE perspective, for a variable bandwidth scenario with different buffer sizes. Each network contains 30 clients streaming video. The x-axis reports the buffer size, in seconds.



(a) Average QoE of the two client groups.



(b) QoE standard deviation of the two client groups

Fig. 7: Evaluation of the heterogeneous scenario, from a QoE perspective. Each network contains 30 clients streaming video. The x-axis reports the percentage of MSS clients on the total.

dependency on the current available bandwidth. When the buffer is sufficiently filled, a client can request the best quality level to maximize its QoE even though the current available bandwidth would not allow it. Despite that, a large buffer should be avoided because it causes a long delay when streaming live contents and a big memory occupation on the device. Also when the buffer is only a few seconds (4 seconds case in Figure 6), the performance of the FINEAS client is very close to that of MSS. Unlike the previous case, if the buffer size is very small, the client has to follow almost precisely the available bandwidth, in order to minimize the risk of freezes. Consequently, the optimization possibilities are reduced. Nevertheless, our solution always outperforms MSS from the fairness point of view, while achieving a similar or higher average QoE.

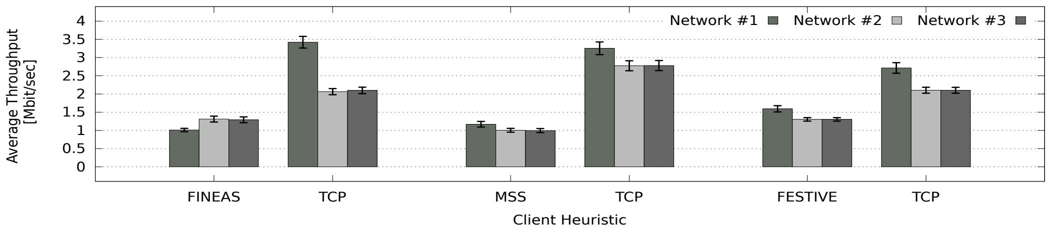


Fig. 8: Evaluation of a scenario with concurrent TCP clients. TCP clients are 25% of the total. The x-axis reports the different heuristics, the y-axis the average throughput for each network over the 50 simulated episodes.

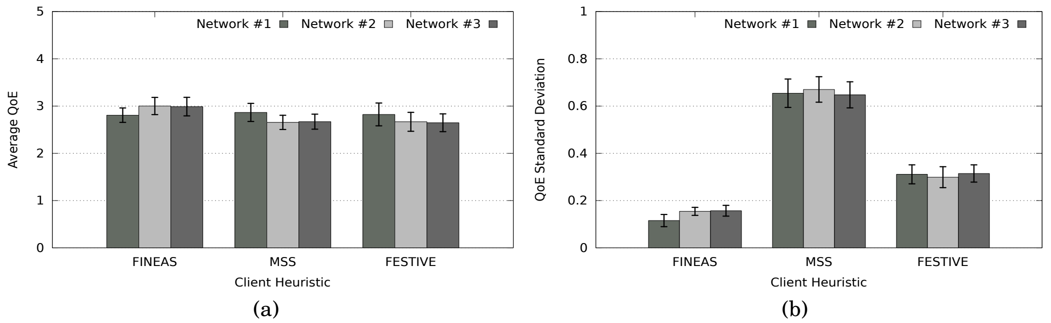
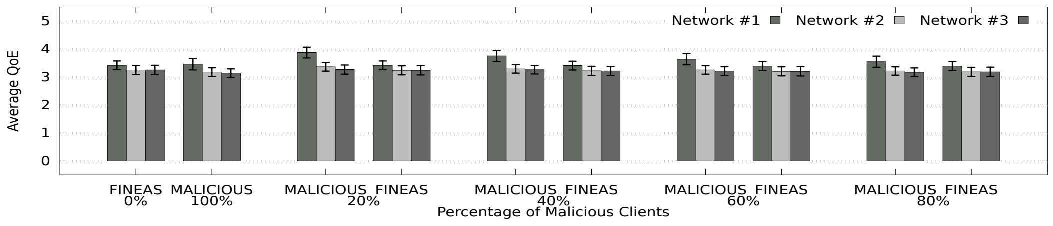


Fig. 9: Evaluation of a scenario with concurrent TCP clients, from a QoE perspective. TCP clients are 25% of the total. The x-axis reports the different heuristics, the y-axis the average QoE (a) and its standard deviation (b) for the HAS clients subgroup.

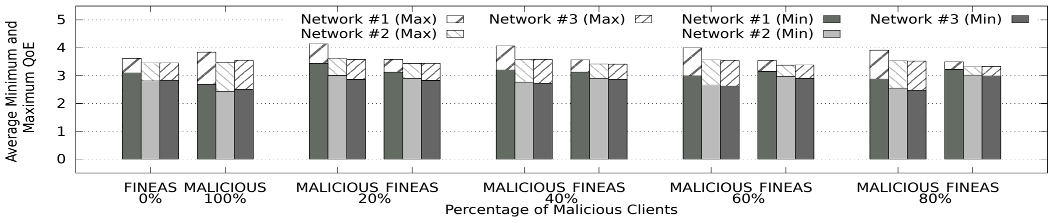
4.6. Client Heterogeneity

So far, we have investigated a homogeneous scenario, where all the clients are equipped with the same heuristic. In a real scenario, different client types can stream video at the same time. We analyse here the interaction between two different groups of clients, one equipped with the FINEAS heuristic and the other equipped with the MSS one. The fairness signal is computed considering the totality of clients streaming video, but is analysed by our clients only. The simulation settings are those reported in Section 4.4. We increase the size of the MSS group from 20% to 80% of the total, for a number of MSS clients ranging from 18 to 72; the assignment of the MSS clients to each network is made randomly at the beginning of each simulated episode. In order to evaluate the results, we compute for each simulated episode, for both client groups, the average QoE and its standard deviation. We then average these values over the 50 simulated episodes. Figure 7 reports the obtained results, together with the confidence intervals at 95%. The FINEAS group is always able to outperform the MSS one, with a gain in the average QoE between 20% and 30% for all the networks in all scenarios. Interestingly, the performance of the two groups remains constant even if the number of MSS clients changes. Thanks to the FINEAS heuristic and the in-network fairness signal computation, our clients are always able to select the best quality level to maximize their own QoE and optimize fairness. This optimization is carried out considering all the HAS clients, since the fairness signal is computed considering all the clients streaming video, and not only the ones equipped with our solution. This entails that our clients do not behave greedily with respect to the MSS group. The difference in performance is thus mainly due to the better rate adaptation of the FINEAS clients.

In a second set of experiments, we investigate the interaction between different HAS clients (FINEAS, FESTIVE and MSS) and generic TCP clients. The simulation settings are those reported in Section 4.4. The number of TCP clients is equal to 23 (i.e. 25% of



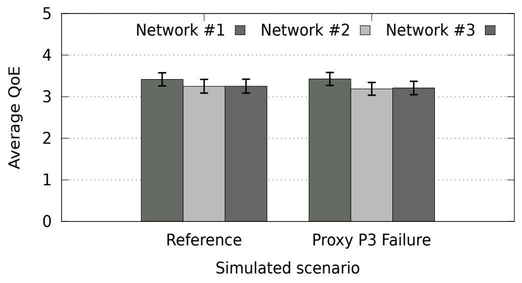
(a) Average QoE of the two client groups. The confidence intervals at 95% are also reported



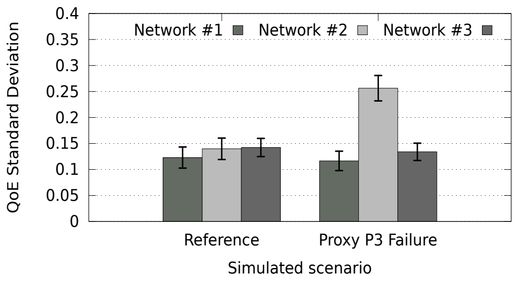
(b) Average minimum and maximum QoE of the two client groups

Fig. 10: Evaluation of the malicious scenario, from a QoE perspective. Each network contains 30 clients streaming video. The x-axis reports the percentage of malicious clients on the total.

the total); the assignment of the TCP clients to each network is made randomly at the beginning of each simulated episode. Each TCP client establishes a connection with the HAS Server, where a TCP sender is installed. Each client downloads the video content from the server in one continuous stream at the fixed rate of 5 Mbps. Consequently, the TCP start-up behaviour is negligible compared to the total video duration and does not affect the outcome of this analysis. During each experiment, the same HAS heuristic is used. As far as the fairness signal computation is concerned, it is performed considering the HAS clients only, since we are not in control of the traffic generated by other non-HAS applications (browsing, file downloading etc.). In order to evaluate the results, we compute for each simulated episode, for both the HAS and TCP groups, the average TCP throughput. We then average this value over the 50 simulated episodes. We also compute the average QoE and its standard deviation over the 50 episodes for the HAS group. Figures 8 and 9 report the obtained results, with confidence intervals at 95%. Based on the simulated topology and the bandwidth pattern (see Section 4.4), each client should obtain, on average, a bandwidth of 2 Mbps. From Figure 8, it appears clearly that the TCP group behaves greedily with respect to the HAS group, independently from the adopted heuristic. TCP clients in Network #1, which are the closest to the HAS server and thus experience the lowest RTT, obtain the highest throughput. If we consider the results from a QoE point of view, it is possible to draw two conclusions. First, the FINEAS heuristic is still able to reach the best results both from the QoE and the fairness perspective, as one can see in Figures 9a and 9b, respectively. Particularly, the average QoE for Network #1 is the same as that of the MSS and the FESTIVE clients, while the QoE standard deviation is considerably lower in the FINEAS case. The gain achieved in terms of average QoE for Network #2 and #3 is about 13%. Second, exploited network resources being equal, the FINEAS client results in a better overall perceived video quality, i.e. is more efficient. As observed in Figure 8, the average TCP throughput reached by the FINEAS and FESTIVE clients for Networks #2 and #3 is almost equal and higher than that reached by MSS. Nevertheless, the QoE achieved by the FINEAS clients is considerably higher than that achieved by the two other heuristics, entailing a better utilization of network resources. The same consideration applies to Network #1. Even though the average throughput reached by the FINEAS clients is lower than MSS and FESTIVE, the average QoE is the same.



(a)



(b)

Fig. 11: Performance evaluation, from a QoE perspective, of a scenario where Coordination Proxy P3 (see Figure 3a) is defective. Each network contains 30 clients streaming video. The graphs report the average QoE (a) and its standard deviation (b).

4.7. Malicious Clients and Proxy Failure

In this section, we analyse the robustness of our solution in presence of malicious clients and in case of a coordination proxy failure.

In the first set of experiments, we simulate the presence of malicious clients equipped with a modified FINEAS heuristic that ignores the fairness signal. The only objective of a malicious client is to maximize its own QoE, neglecting the presence of other clients and fairness. The simulation settings are those reported in Section 4.4. The fairness signal computation is performed considering the totality of clients streaming video, but is analysed by the normal FINEAS clients only. We increase the size of the malicious group from 0% to 100% of the total, for a number of malicious clients ranging from 0 to 90; the assignment of the malicious clients to each network is made randomly at the beginning of each simulated episode. In order to evaluate the results, we compute for each simulated episode, for both client groups, the average QoE, its standard deviation, the minimum and the maximum QoE. We then average these values over the 50 simulated episodes. The results are reported in Figure 10. The first two histograms report the results for the two limit cases where the clients are all equipped with the FINEAS heuristic or are all malicious. As observed in Figure 10a, the two groups of clients obtain very similar average QoE. The FINEAS group obtains a QoE standard deviation half of that obtained by the malicious group (not shown in the paper due to space limitations), as the former follows the fairness signal. More interestingly, Figure 10b highlights that a malicious behaviour is actually detrimental for some clients belonging to the malicious group. This occurs when a malicious client would obtain a better QoE by following the fairness signal instead of ignoring it. Figure 10b shows the average minimum and maximum QoE obtained by the two groups, for each sub-network. The average minimum QoE for the FINEAS group is always higher or equal to the average minimum QoE of the malicious group, i.e. some malicious clients obtain a QoE lower than that they could obtain behaving fairly. Moreover, this behaviour becomes more evident as the number of malicious clients increases. The only exception is represented by Network #1 in the scenario with 20% malicious clients. Nevertheless, the FINEAS group outperforms the malicious one from the fairness point of view, while achieving a similar QoE.

In the second set of experiments, coordination proxy P3 is lacking, so that clients in Network #2 do not receive the fairness signal anymore (see Figure 3a). This scenario represents a situation where proxy P3 is not deployed or cannot forward the fairness signal because of a fault. Figure 11 shows the results, obtained averaging the QoE and its standard deviation over the 50 simulated episodes. As expected, the failure of proxy P3 mainly affects the performance of Network #2 from the fairness point of view: the QoE standard deviation is almost doubled compared to the reference scenario presented in Section 4.4, while the average QoE is the same. This means that some clients experience a lower QoE compared to the reference scenario, as shown also in the malicious clients case. Interestingly, Networks #1 and #3 are unaffected by the failure of proxy P3, both from the QoE and the fairness point of view. This behaviour can be explained considering two aspects. First, both networks are still provided with the fairness signal. Second, the average bandwidth consumed by Network #2 does not differ significantly from that consumed in the reference scenario. The proxy failure affects Network #2 from the fairness point of view only, while the average requested quality level, equal to 5.42, does not significantly change compared to the reference scenario (see Table II). This entails that the bandwidth available to Networks #1 and #3 in case of proxy failure and in the reference scenario is similar and no differences in performance between the two scenarios are noticeable.

5. CONCLUSIONS

In this paper, we presented FINEAS, a HAS heuristic able to dynamically adapt its behavior depending on network conditions, in order to obtain a high QoE. Moreover, this client is able to select the best quality level in order to achieve fairness from the QoE point of view. This was necessary as state-of-the-art rate adaptation heuristics introduced non-negligible differences in obtained quality among clients. Fairness is achieved by means of a system of intermediate nodes, called coordination proxies, in charge of collecting information on the overall network conditions. This information is then provided to the FINEAS clients, which use it to refine their quality decision process. Numerical simulations using NS-3 have validated the effectiveness of the proposed approach. Particularly, we have compared our solution with the Microsoft ISS Smooth Streaming client, a Q-Learning based client [Claeys et al. 2014], a single-client heuristic [Miller et al. 2012] and the FESTIVE algorithm [Jiang et al. 2014]. In the evaluated bandwidth scenarios, we were able to show that our multi-client HAS framework resulted in a better video quality and in a remarkable improvement of fairness, up to 20% and 80% respectively, when compared to the reference algorithms.

ACKNOWLEDGMENTS

The research was performed partially within the iMinds V-FORCE project (under IWT grant agreement no. 130655). This work was partly funded by FLAMINGO, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme. Maxim Claeys is funded by a grant of the Agency for Innovation by Science and Technology in Flanders (IWT).

REFERENCES

3GPP. 2013. 3GPP system - fixed broadband access network interworking (3GPP TS 23.139 version 11.3.0 Release 11). (2013), 1–90. http://www.3gpp.org/DynaReport/23139.htm

V. Adzic, H. Kalva, and B. Furht. 2011. Optimized adaptive HTTP streaming for mobile devices. *Proceedings of SPIE, Applications of Digital Image Processing* (2011).

S. Akhshabi, L. Anantakrishnan, A. C. Begen, and C. Dovrolis. 2012a. What Happens when HTTP Adaptive Streaming Players Compete for Bandwidth?. In *22nd International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV '12)*. ACM, 9–14.

S. Akhshabi, S. Narayanaswamy, A. C. Begen, and C. Dovrolis. 2012b. An Experimental Evaluation of Rate-adaptive Video Players over HTTP. *Signal Processing: Image Communication* 27, 4 (2012), 271–287.

N. Bouten, J. Famaey, S. Latré, R. Huysegems, B. De Vleeschauwer, W. Van Leekwijck, and F. De Turck. 2012. QoE optimization through in-network quality adaptation for HTTP Adaptive Streaming. In *2012 International Conference on Network and Service Management (CNSM) and 2012 Workshop on Systems Virtualiztion Management (SVM)*. 336–342.

M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck. 2014. Design and Optimization of a (FA)Q-Learning-based HTTP Adaptive Streaming Client. *Connection Science* 26, 01 (2014), 27–45.

C. Claus and C. Boutilier. 1998. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In *Fifteenth National / Tenth Conference on Artificial Intelligence / Innovative Applications of Artificial Intelligence (AAAI '98 / IAAI '98)*. American Association for Artificial Intelligence, 746–752.

L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. 2013. ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH). In *2013 International Packet Video Workshop (PV)*. 1–8.

L. De Cicco, S. Mascolo, and V. Palmisano. 2011. Feedback Control for Adaptive Live Video Streaming. In *Second Annual ACM Conference on Multimedia Systems (MMSys '11)*. ACM, 145–156.

J. De Vriendt, D. De Vleeschauwer, and D. Robinson. 2013. Model for estimating QoE of video delivered using HTTP adaptive streaming. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. 1288–1293.

A. El Essaili, D. Schroeder, D. Staehle, M. Shehada, W. Kellerer, and E. G. Steinbach. 2013. Quality-of-experience driven adaptive HTTP media delivery. In *2013 IEEE International Conference on Communications (ICC)*. 2480–2485.

P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race. 2013. Towards Network-wide QoE Fairness Using Openflow-assisted Adaptive Video Streaming. In *2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking (FhMN '13)*. ACM, New York, NY, USA, 15–20.

R. Houdaille and S. Gouache. 2012. Shaping HTTP Adaptive Streams for a Better User Experience. In *3rd Multimedia Systems Conference (MMSys '12)*. ACM, 1–9.

D. Jarnikov and T. Ozcelebi. 2010. Client intelligence for adaptive streaming solutions. In *2010 IEEE International Conference on Multimedia and Expo (ICME)*. 1499–1504.

J. Jiang, V. Sekar, and H. Zhang. 2014. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive. *IEEE/ACM Transactions on Networking* 22, 1 (Feb 2014), 326–340.

R. Kuschnig, I. Kofler, and H. Hellwagner. 2010. An Evaluation of TCP-based Rate-control Algorithms for Adaptive Internet Streaming of H.264/SVC. In *First Annual ACM SIGMM Conference on Multimedia Systems (MMSys '10)*. ACM, 157–168.

Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran. 2014. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *IEEE Journal on Selected Areas in Communications* (2014), 719–733.

C. Liu, I. Bouazizi, M. M. Hannuksela, and M. Gabbouj. 2012. Rate Adaptation for Dynamic Adaptive Streaming over HTTP in Content Distribution Network. *Image Communication* 27, 4 (2012), 288–311.

K. J. Ma and R. Bartos. 2011. HTTP Live Streaming Bandwidth Management Using Intelligent Segment Selection. In *2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)*. 1–5.

K. Miller, E. Quacchio, G. Gennari, and A. Wolisz. 2012. Adaptation algorithm for adaptive streaming over HTTP. In *2012 International Packet Video Workshop (PV)*. 173–178.

R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang. 2012. QDASH: A QoE-aware DASH System. In *3rd Multimedia Systems Conference (MMSys '12)*. ACM, 11–22.

C. Müller, S. Lederer, and C. Timmerer. 2012a. An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments. In *4th Workshop on Mobile Video (MoVid '12)*. ACM, 37–42.

C. Müller, S. Lederer, and C. Timmerer. 2012b. A proxy effect analysis and fair adaptation algorithm for multiple competing Dynamic Adaptive Streaming over HTTP clients. In *2012 IEEE Visual Communications and Image Processing Conference (VCIP)*. 1–6.

S. Petrangeli, M. Claeys, S. Latré, J. Famaey, and F. De Turck. 2014. A multi-agent Q-Learning-based framework for achieving fairness in HTTP Adaptive Streaming. In *2014 IEEE Network Operations and Management Symposium (NOMS)*. 1–9.

H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen. 2012. Video Streaming Using a Location-based Bandwidth-lookup Service for Bitrate Planning. *ACM Transactions on Multimedia Computing, Communications and Applications* 8, 3, Article 24 (Aug. 2012), 19 pages.

R. de O. Schmidt, R. Sadre, A. Sperotto, and A. Pras. 2013. Lightweight link dimensioning using sFlow sampling. In *2013 International Conference on Network and Service Management (CNSM)*. 152–155.

G. Tian and Y. Liu. 2012. Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming. In *8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '12)*. ACM, 109–120.

B. Villa, P. Heegaard, and A. Instefjord. 2012. Improving Fairness for Adaptive HTTP Video Streaming. In *Information and Communication Technologies*, Rbert Szab and Attila Vidcs (Eds.). Lecture Notes in Computer Science, Vol. 7479. Springer Berlin Heidelberg, 183–193.

S. Xiang, L. Cai, and J. Pan. 2012. Adaptive Scalable Video Streaming in Wireless Networks. In *3rd Multimedia Systems Conference (MMSys '12)*. ACM, 167–172.

C. Zhou, X. Zhang, L. Huo, and Z. Guo. 2012. A control-theoretic approach to rate adaptation for dynamic HTTP streaming. In *2012 IEEE Visual Communications and Image Processing Conference (VCIP)*. 1–6.

Received April 2014; revised XXXX XXXX; accepted XXXX XXXX

Online Appendix to: QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming

STEFANO PETRANGELI, JEROEN FAMAEY, MAXIM CLAEYS and FILIP DE TURCK,
Department of Information Technology (INTEC), Ghent University–iMinds
STEVEN LATRÉ, Department of Mathematics and Computer Science, University of Antwerp–iMinds

In this appendix, we formally present the multi-client fairness problem addressed in the paper, based on our previous work [Petrangeli et al. 2014]. We assume a scenario where multiple local access networks are connected to the core Internet Service Provider (ISP) network and to the Internet by means of several traffic concentrators. Particularly, we assume a set of N different access networks, where network i contains n^i HAS clients, streaming video at the same time. The problem we want to solve is two-fold. First, clients have to obtain the highest possible video quality. Second, they have to show similar performance if they share bottleneck links, i.e. fairness. Based on this consideration, all the clients sharing the same bottlenecks should act fairly, even if they belong to different access networks.

The formal characterization of the fairness problem is given in Definition .1. We define K_j^i to be the number of segments the video content streamed by client j in network i is composed of, ${q_{max}}_j^i$ the highest available quality level, $q_j^i(k)$ the quality level requested for the $k-th$ segment and q_j^i the vector containing all the quality levels requested by client j. q^i is the matrix containing all the quality levels requested by the clients belonging to network i. $d_j^i(k, q^1, \ldots, q^i, \ldots, q^N, \beta^i)$ represents the download time of the $k-th$ segment for client j, while $b_j^i(k)$ denotes the video player buffer filling level of client j when the $k-th$ segment download starts. β^i is the vector containing the available bandwidth for the HAS clients belonging to network i, i.e. what remains of the network capacity once all the non-HAS traffic has been served.

Definition .1. Fairness Optimization Problem

$$\mathrm{J(q^i) = \xi \times QualityIndex(q^i) - (1-\xi) \times FairIndex(q^i)} \quad \text{with} \quad \xi \in [0;1],\ \mathrm{q^i = (q^i_1, \ldots, q^i_{n^i})}$$

$$\mathrm{G(J(q^1), \ldots, J(q^N))} = \nu \times \frac{1}{\mathrm{N}} \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{J(q^i)} - (1-\nu) \times \sqrt{\frac{1}{\mathrm{N}} \sum_{\mathrm{i=1}}^{\mathrm{N}} \left(\mathrm{J(q^i)} - \frac{1}{\mathrm{N}} \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{J(q^i)} \right)^2} \quad \text{with} \quad \nu \in [0;1]$$

$$\begin{aligned} \underset{q^1,\ldots,q^N}{\text{maximize}} \quad & \mathrm{G(J(q^1), \ldots, J(q^N))} \\ \text{subject to} \quad & 1 \le q_j^i(k) \le {q_{max}}_j^i \quad \forall i = 1 \ldots N, \forall j = 1 \ldots n^i, \forall k = 1 \ldots K_j^i \\ & d_j^i(k, q^1, \ldots, q^i, \ldots, q^N, \beta^i) \le b_j^i(k) \quad \forall i = 1 \ldots N, \forall j = 1 \ldots n^i, \forall k = 1 \ldots K_j^i \end{aligned}$$

The function $J(q^i)$ represents the performance of network i and is the linear combination of two terms. $QualityIndex(q^i)$, measures the overall video streaming quality

© 0000 ACM 1551-6857/0000/-ART0 $15.00
DOI: http://dx.doi.org/10.1145/0000000.0000000

of the clients in network i, while $FairIndex(q^i)$ measures fairness. By maximizing the linear combination of the average of $J(q^1), \ldots, J(q^N)$ and of the negative of their standard deviation, we aim to achieve the highest possible performance for every network while keeping the deviation among them as low as possible. Depending on applications and scenarios, ν can be tuned to benefit one of the two terms.

The final formulation of $QualityIndex$ and $FairIndex$ depends on the focus given to the multi-client optimization problem. From an application-aware point of view, the focus is on the user perceived video quality, denoted as QoE. In this case, $QualityIndex$ *can be characterized as the average of clients' QoE, while* $FairIndex$ *as the standard deviation of this average*. This way, we aim to maximize clients' QoE and keep the deviation as low as possible. The QoE model used in this paper is presented in Section 4. From an application-agnostic point of view, the focus is on clients' bit rate. With this formulation, we are interested in fairness from a network point of view, i.e. clients have to fairly share the available bandwidth. Consequently, $QualityIndex$ *and* $FairIndex$ *can be computed as the average of clients' bit rate and as the standard deviation of this average*, respectively. In the design of our client, we focused on the application-aware interpretation, since it is directly correlated to the user perceived video quality.

In light of the above, it is clear why $QualityIndex$ and $FairIndex$ have to be optimized together. If we only optimize fairness, clients could obtain similar but unacceptable video qualities. Instead, our goal is also to reach the highest possible video quality. Depending on applications and scenarios, ξ can be tuned to benefit one of the two terms.

The second constraint of the optimization problem is intended to avoid freezes in the video playback. The download time of the next segment $d^i_j(k, q^1, \ldots, q^i, \ldots, q^N, \beta^i)$ has to be lower than the video player buffer filling level when the download starts, $b^i_j(k)$. This way, the video player buffer will never be empty and freezes are avoided. It is worth noting that the download time of the $k-th$ segment is a function of the quality levels downloaded simultaneously by all the other clients $(q^1, \ldots, q^i, \ldots, q^N)$ and of the available bandwidth β^i.