Using buffered playtime for QoE-oriented resource management of YouTube video streaming

F. Wamser[1]*, D. Hock[1], M. Seufert[1], B. Staehle[2], R. Pries[1] and P. Tran-Gia[1]

[1] University of Würzburg, Institute of Computer Science, Germany
[2] Fraunhofer IIS, Nuremberg, Germany

***Correspondence**
F. Wamser, University of Würzburg, Institute of Computer Science, Germany.
E-mail: florian.wamser@informatik.uni-wuerzburg.de

1. INTRODUCTION

YouTube is the most important online platform for streaming video clips. According to Cisco Systems [1], Internet video generated 10.423 exabytes (EB) of traffic per month in 2011. This is about 51% of all consumer Internet traffic. Short-form Internet video (video generally less than 7 min in length) is responsible for around 1.211 EB per month, representing 12% of the entire consumer traffic. The popularity as well as the continuously rising number of users pose new challenges for Internet service providers (ISPs). Especially, in access networks where the resources are limited and the providers are interested in reducing their operational expenditure, it is becoming increasingly important to optimise the network efficiently for popular services like YouTube video streaming.

The providers have to address two conflicting issues here. On the one hand, they have to reduce their operational expenditure for the network. On the other hand, they are faced with the increasing quality demands of the users. Consequently, the network resources have to be operated in an efficient way. This raises the need for novel resource management solutions, specifically taking into account the most frequently used applications and their use of resources. Especially, the user-perceived quality of the applications used in the network is in focus of the optimization because customers rate the network quality according to the performance of their used applications.

To optimise the use of transmission resources and the quality of the network, there is a shift from Quality of Service (QoS) resource management [2–6] to quality of experience (QoE)-based resource management [7–11]. QoS resource management maintains network-level parameters such as packet loss to provide good network quality. A good network quality, however, does not necessarily imply a smooth-running application because the heterogeneous applications depend on very different parameters. Video streaming, for example, has other requirements than web browsing. Web users are interested in a short page load time. Video users, however, expect a good video quality or a smooth playback without interruptions. Another example is Skype that combines different functions such as chatting, telephony and live video streaming in one application. Each function has its own requirements to the network. To comply with this, QoE-based resource management directly addresses the perceived quality of the application

at the end user. In addition to objective quality factors, it considers the subjective experience and the satisfaction of a user with a particular service [12].

In this paper, we consider YouTube video streaming as an important service for access networks. We address the question on how to implement an efficient resource management for YouTube video streaming that considers the user-perceived quality. We show different options for resource management for YouTube video streaming and evaluate them in a wireless mesh testbed. Two approaches seem to be the most promising: network control and service control. Both are introduced in this paper to address the challenge to improve the efficiency of the network. Network control tries to optimise the network resources, whereas service control focuses on the applications running in the network. For service control, we propose a network-wide quality change algorithm that is able to reduce the YouTube video resolution if a user in the network experiences a bad QoE. Both, network and service controls, are coordinated by a central entity in order to avoid situations in which multiple resource management actions for different clients interfere with each other.

The remainder of this paper is organised as follows. In Section 2, an overview of the publications related to our work is given. In Section 3, technical details about YouTube video streaming are summarised, followed by a general introduction to application-aware resource management in Section 4. A detailed description of the resource management architecture for YouTube video streaming can be found in Section 5. Afterwards, Section 6 contains an evaluation of various resource management algorithms. Finally, conclusions are drawn in Section 7.

2. RELATED WORK

Traditionally, resource management in communication networks is performed based on algorithms that focus on satisfying QoS requirements of applications [2–6]. The users are classified into different classes according to their needs to the network such as minimum data rate and maximum packet latency. The classes are commonly defined by the network and the forwarding is performed on flow basis according to QoS definitions that are associated with the classes. Especially, video traffic is supported by special QoS classes in modern communication networks [13]. In IEEE 802.16e [14], for example, in addition to the best effort QoS class, a constant-bit-rate service and a real-time polling service (rtPS) are defined. rtPS is designed to support real-time service flows that generate data packets of variable size on a periodic basis, such as MPEG video.

Although it is important to fulfil the QoS requirements on network level, it is even more important to satisfy the subjective user demands (QoE). All aforementioned approaches have the disadvantage that they do not process the video data according to the quality that the user actually experiences at application level.

Gross et al., Khan et al. and Ameigeiras et al. [7–9] carry out a cross-layer optimization. The approach is as follows. In the case of limited transmission resources, important packets in MPEG videos, mainly I-frames, are prioritised to optimise the perceived quality. In [15], a multilayer video encoding with scalable video codec is used. The goal is to control the different layers in different QoS classes with different priorities to specifically drop video layers with less importance if the network is congested. In [11], QoE-based scheduling for wireless mesh networks is proposed. The authors take into account not only video and audio streaming but also data traffic using simple MOS metrics, which map QoE to simple QoS parameters.

Commonly, the relationship between QoE and QoS network parameters is not of linear scale, that is, altering the QoS parameters results in different QoE levels [16]. Consequently, there are resource management algorithms that define an acceptable end user quality at minimal resource utilisation as control objective [17].

The work about QoE in general can be divided into two distinctive research fields. There are papers about understanding and modelling QoE for different applications [16, 17] and there are papers that make use of QoE as resource management metric [7–11].

To extend this concept to general applications, other approaches have been developed. Application-aware management approaches use cross-layer information from the running application to adjust the control decisions [18, 19]. In [18], QoE metrics are used to differentiate between different services and thus, generate a one-dimensional optimisation function for the radio resource management. This approach can be seen as a continuation of former work which used a so-called utility function for resource management to define a scheduling order with respect to different application cases [20–24].

Finally, there are video streaming products such as Adobe Systems's HTTP Dynamic Streaming, Apple's HTTP Live Streaming and Microsoft's Smooth Streaming. They all automatically vary the quality and size of the streams dynamically during playback to provide the best possible viewing experience for the users. Related to these solutions, there is also an ISO/IEC (International Organization for Standardization/International Electrotechnical Commission) standard of the MPEG working group called Dynamic Adaptive Streaming over HTTP (DASH) [25]. It allows video streaming where file segments of video streams are dynamically requested with HTTP by the client according to the network conditions or user preferences. The goal is to enable an efficient and high-quality delivery of streaming services over the Internet.

Our approach is partly similar to the DASH standard because DASH also allows for the consideration of application parameters such as buffered playtime of videos. However, we exploit the client information in the network as well as at the client. This allows both coordinated client-side quality management and network-based resource management.

3. TECHNICAL DETAILS ON YOUTUBE VIDEO STREAMING

YouTube is a streaming platform that mainly offers small to medium-sized video clips to its users. The video are encoded according to the H.264/MPEG-4 Advanced Video Coding (AVC) as default videos compression format. YouTube uses progressive HTTP streaming as streaming technology. The client essentially downloads the video data over an HTTP connection and already starts playing the video while the download is not yet completed. The client application is a precompiled Adobe Flash player assembly that runs at the client in the web browser. Downloaded data is stored in a temporary file that serves as buffer for video playtime. YouTube is doing pre-buffering which means that the client starts playing only after a certain level of playtime is available in the buffer. The time from requesting a video until the buffer is sufficiently filled such that the video starts playing is called *startup delay*.

While the video is playing, the server refills the buffer by periodically transmitting blocks of video data to the client. The actual data arrival at the client is regulated by Transmission Control Protocol (TCP) and depends on the available bandwidth.

These two transmission phases, the initial filling of the buffer before the video starts playing and the periodic refilling of the buffer while the video is playing, are controlled by the YouTube content servers. The transmission patterns during these phases are adapted for every video according to the total video rate [26–29]. *Stalling*, that is, an interruption of the video playback, occurs if the playtime buffer becomes empty during video playback. Hence, the major goal of a resource management mechanism designed for supporting YouTube streaming is to achieve a short startup delay and to avoid stalling.

YouTube, furthermore, provides an application programming interface (API) to allow a control of the YouTube player at the client side. Possible commands are basic player controls such as play, pause, jump or stop as well as player updates regarding size and design. In addition, statistics can be queried and the playback quality can be set or changed. The latter one is used later on.

4. APPLICATION-AWARE AND QUALITY OF EXPERIENCE-AWARE RESOURCE MANAGEMENT

Aquarema is a framework for **A**pplication and **Qu**ality of Experience-**A**ware **Re**source **Ma**nagement introduced in [19]. In this section, we describe its general concepts, whereas its implementation is described in Section 5. This section demonstrates how the Aquarema framework can be used to implement a traffic and resource management for an efficient and QoE-aware delivery of YouTube streams in an access network.

The key idea of Aquarema is to collect information about network and application status at a central entity, the *network advisor*. It is able to trigger a number of resource management actions that either change the traffic handling in the network or the traffic produced by the application. The first type of actions are summarised under the term *network control actions*, the second type of actions are referred to as *service control actions*.

The general goal of Aquarema is to improve the overall QoE in the network or rather to avoid QoE degradation. It follows a reactive approach for resource and traffic management. Resource management actions are only triggered if (i) a QoE degradation or an indication for an imminent QoE degradation has been detected; and (ii) a resource management action is available that will avoid or limit the QoE degradation without overly harming the QoE of other users.

Aquarema does not follow a proactive approach to optimise a QoE-based metric and it is also not targeting at an optimisation of network parameters such as a balanced link utilisation. As a consequence, it can or even should run in addition to a 'traditional' traffic or resource management mechanism that does not take into account application layer performance but relies on typical network performance indicators such as load, available bandwidth, delay, packet loss and traffic classification. The idea is not to interfere with other mechanisms as long as the application layer performance and QoE is good enough. Only if a QoE degradation occurs in spite of these traffic and resource management mechanisms, Aquarema will trigger actions in order to avoid a QoE degradation.

The resource management actions are intentionally designed for access networks. They are only conducted inside the access network where the situation is known and only if the monitoring entities detect that the problem is located in the access network. Consequently, problems originating outside the access network cannot be solved and are not in focus of the resource management.

Figure 1 shows the different resource management components of the Aquarema framework. This is, first of all, the *network advisor* that receives information from the application and network monitors. It is additionally connected to the nodes in the network that actually enforce resource or traffic management decisions. When notified by an application monitor about a critical state of an application, the network advisor evaluates its set of resource management actions. This means that based on the information on application and network status, it predicts how a resource management action changes the network status and estimates whether the QoE situation improves. From all resource management actions with potentially positive outcome, the one to be executed is selected based on (i) the confidence in the prediction; (ii) the degree of the QoE improvement; or (iii) the effort for performing the resource management action.

Another key component is the *application monitor* that is running on the client. It sends the application status to the network advisor. This communication can be either event-driven or periodic and either push-based or pull-based. The task of the monitor is to keep track of the status of traffic

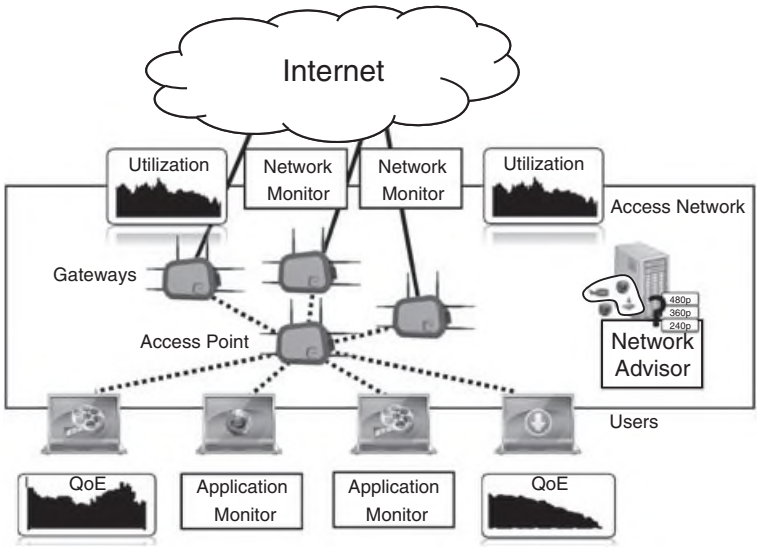


Figure 1. The Aquarema concept.

intensive or QoE sensitive applications that are subject to the resource management decisions. The status of an application is a collection of key performance indicators that the customer will directly perceive as quality parameters. These key performance indicators are application specific and describe whether the current performance offered by the network leads to a QoE degradation.

The key performance indicators cannot be directly mapped to a QoE value, as QoE describes the overall experience of a user with a service. Therefore, it also depends on many other factors such as non-measurable subjective user demands. However, the performance indicators indicate if a QoE degradation is imminent. Using these key performance indicators, Aquarema is able to indirectly consider the QoE of applications within the resource management and avoid degradations.

To give an example, key performance indicators for RTP streaming are bandwidth on application layer, packet loss or jitter that may be mapped to a QoE metric by using a QoE model. Key performance indicators for HTTP streaming services are the bandwidth on application layer or the buffered playtime in the client as described in Section 3. When the buffered playtime is low and the bandwidth is below the video rate, a period of stalling will probably occur if no measures are taken. According to [30, 31], stalling is the factor dominating the QoE for video clips clearly exceeding the significance of video resolution as a second impact factor. In the following, we focus on the buffered playtime and only consider resource management actions that take the buffered playtime into account.

The third major component of Aquarema is the *network and flow monitor*. It monitors typical network parameters such as load, packet loss, buffer status of the network interface and number of connections and sends these parameters to the network advisor. Additionally, it is able to monitor a certain flow in the network to observe its current throughput and state. Again, the communication may be pull-based or push-based and periodic or event-driven.

5. IMPLEMENTATION DETAILS AND DESCRIPTION OF THE CONTROL ALGORITHMS

In the following, the implementation of Aquarema is explained in detail. Required components for the resource management are network and application monitors, a network advisor and resource management actions. First, we describe the monitoring part of our implementation. The monitoring provides the necessary application information and helps to identify the overall network situation to enable a targeted resource management. Thereafter, the network advisor and the control algorithms and actions are described. We distinguish between network control and application-layer service control. In this paper, because we focus on YouTube, all entities are described with respect to YouTube video streaming.

5.1. YouTube application monitor

Application monitoring is performed directly at the client. Therefore, a Mozilla Firefox extension is installed at the client that monitors the instances of Adobe Flash embedded on a website. If a YouTube Flash player is detected, the plug-in uses the YouTube API and reads all relevant parameters that are required for the resource management. Parameters are queried for two purposes. On the one hand, we query the key performance indicator, in this case, the buffered playtime of the YouTube player. On the other hand, we request a general information for

Table I. Priority classification.

Buffered playtime (s)	Priority class
> 15	5
> 10	4
> 5	3
> 2	2
$\leqslant 2$	1

the resource management actions. This includes for example the possible video resolutions for YouTube which are both offered by YouTube and currently supported by the end-user device.

In particular, the YouTube application monitor queries the following parameters and forwards them to the network advisor, (i) current buffered playtime in seconds; (ii) available video resolutions as defined in [32], (iii) current video resolution; and (iv) flow information such as transport layer ports and IP addresses.

The buffered playtime in seconds can unfortunately not be directly read from the YouTube API. The YouTube player only exports the amount of loaded bytes of the video content. We use the approach proposed in [33]. Because the loaded bytes result in a different amount of playtime depending on the current video resolution and encoding, the exact buffered playtime is derived out of the captured video data. In version 3.6, the YouTube application monitor is able to analyse Adobe Systems's flash video (FLV) which is the current default video container format for YouTube videos.

The application information is continuously measured at the client but only reported in an event-based way. Information for resource management actions such as available video resolutions can be identified directly at the beginning of the YouTube video playback. The data is sent once immediately after the video playback was detected. If the user changes the video resolution or a new video stream is requested by the YouTube player, a similar message is sent to the network advisor right after the detection of the change. The buffered playtime is reported according to configurable thresholds. It may be necessary to adjust these thresholds based on the resource management action that uses the values, for example, see Section 5.4.2 and Table I.

5.2. Network and flow monitor

The network and flow monitor has two different functions. It measures the utilisation of individual links and the current throughput of individual flows in the network. This information is used by the network advisor to estimate the benefit of possible resource management actions.

The load of the different links to the Internet is directly measured at the corresponding router or switch to the Internet. It is described by two values as follows: the maximum capacity and the current throughput on the link. The current throughput is periodically polled every second by the network advisor from the network monitor at the router or switch. A moving average is calculated with a window size of 5 s to compensate short load peaks. We assume that the maximum capacity of the link is fixed and known at the network advisor.

To determine the throughput and state of individual flows, the network advisor sends the flow signature consisting of the IP address and transport layer port to the network monitor. The network monitor at the router uses a connection tracking module to gather the information. In the case of Linux OS, the kernel module *conntrack* is used. For Microsoft Windows-based systems, the Event Tracking for Windows is used in the network and flow monitors. If a flow is monitored, the router sends once in a second the current throughput of the flow to the network advisor which, again, calculates the moving average of 5 s for this flow.

5.3. Network advisor

The network advisor is the central entity that triggers the resource management. It periodically collects information from the network and receives information from the event-based application monitors. All information is stored in a database so that a set of information about current applications in the network and the current network situation are known. On the basis of this information, the network advisor is able to trigger a number of resource management actions.

To be able to conduct the resource management actions, *strategies* are defined. Strategies map a certain application key performance indicator to a set of resource management actions. For example, there is a strategy for the buffered playtime of YouTube video streaming that is associated with the resource management action gateway change. This strategy is introduced in Section 5.4.1. In contrast, there is additionally a strategy that allows combined resource management. Here, in the resource management strategy, two actions are included, for instance, gateway change (network control) and video resolution change (service control), see Section 5.4.4.

Within each strategy, for each application and key performance indicator, a critical threshold is defined. If this threshold is exceeded, the network advisor assumes that the application is in a critical condition. If this is the case, it runs the resource management actions of the set of actions defined in the strategy. Each resource management action returns a status information as return value that indicates (a) whether the action was successful or (b) how long it should wait before the next resource management action is triggered. A waiting period after a transacted resource management action is necessary because it takes a short time until an action is enforced in the network. After each action, the network advisor waits the time that the previous resource management action returned. If the action was not successful, it executes the next resource management actions in the list. If a resource management action

was successful, the network advisor terminates resource management and evaluates again the key performance indicators whether the application is in a critical state or not.

5.4. YouTube resource management actions

We distinguish between two different types of resource management actions: network control and service control.

The concept of network control covers all measures that alter network properties or influence the packet flow in the network. The general goal of this concept is to improve the overall QoE of the users. To achieve this, the network has to react dynamically to changing network conditions and requirements of the users' applications.

In this work, two resource management actions that belong to network control are implemented: *gateway change* and *buffer-based prioritisation*. The first management action allows a rerouting of packet flows to different gateways with less utilisation. The second network management action implements traffic shaping to fairly distribute the available capacity according to the application needs. This is performed by the prioritisation of network flows in order to help applications if their QoE deteriorates. We further refer to this as buffer-based prioritisation. A detailed description of the algorithms is given in the following subsections after the enumeration of the implemented service control mechanisms.

The second type of resource management action which is investigated in this work is service control. This includes mechanisms that control the users' applications such that the QoE of a single service is assured. Similar to network control actions, this implies that applications must accept resource management commands. As soon as a service level cannot be sustained, the service control mechanism notifies the application. If a degradation in the quality of the service is imminent, the application is adapted to the new conditions, if possible. Consequently, the application quality experienced by the user can be alleviated slowly and abrupt service failures which ruin users' QoE [16] can be avoided.

There are many different mechanisms for reaching this goal. In this work, video quality reduction of YouTube video streams is implemented by subsequently decreasing the video resolutions. In the following, it is called *quality change*. Other approaches include the adaptation of audio/video codecs as it is already implemented by Skype or within the Annex G extension of H.264/MPEG-4 AVC video standard, which is commonly referred to as scalable video codec.

Both types of resource management actions improve the QoE in the network. Consequently, a combination of them is desirable for an efficient resource management. The combination, furthermore, can be performed according to various objectives or provider preferences. For example, one provider policy can save network resources as long as an acceptable QoE can be maintained, or in contrast, the QoE can be maximised by using all available network resources. Eventually, a combined network and service control shall be provided, which utilises network parameters, application parameters and provider-dependent directives to maximise the perceived quality for a set of users in the network while in each situation, minimising network operator's costs.

5.4.1. Network control: gateway change.

In access networks such as wireless mesh networks, multiple gateways to the Internet might exist. The resource management tool *Nigel* is responsible for dynamically assigning the clients to these different gateways [19]. Changing the Internet gateway of a client during run-time requires to take care of the active connections between a user and the Internet. To achieve this goal, Nigel follows the Mobile IPv4 approach. It establishes an overlay network that ensures a seamless TCP handover. According to the Mobile IPv4 approach, an anchor— the 'gatekeeper'— is located in the Internet as so-called home agent which maintains the IP connection to the corresponding service. The overlay network between the access point and the home agent is established via IP tunnels. Thus, selecting another Internet gateway changes the routing of the IP tunnel. As a consequence, changing the Internet gateway of a client does not affect the actual connection between the home agent and the Internet service as only the virtual paths of the IP tunnels are changed. On the basis of the monitored information about the current gateway utilisation and the needs of the hosted application streams, the network advisor decides which stream is assigned to which gateway. In the following, the algorithm and Nigel's gateway switching policy are described in detail.

Nigel is installed at the edges of the access network, namely, on each access point and on the gatekeeper. The Nigel instance at the access point manages the uplink direction, whereas Nigel running at the gatekeeper is responsible for the downlink direction. To switch a stream to another gateway, a message is sent to Nigel running on the client's access point, naming the new gateway. It switches the uplink and sends a message to Nigel on the gatekeeper also naming the new gateway. Nigel on the gatekeeper switches the downlink and confirms the gateway switch.

In Figure 2, the resource management policy of the gateway change is depicted. The network controller checks at each status update of a YouTube flow if the condition for the resource management action is met and if the video is already playing. The condition for gateway change for YouTube video streaming is that the buffered playtime is below 10 s and that the start time is at least 5 s ago.

We define for YouTube a threshold of 10 s buffered playtime and a start delay of at least 5 s to ensure that the web page and the video player is loaded as well as that the playing of the video has already begun. The download and initialisation of embedded flash objects within the browser can take up to a few seconds. Moreover, it may

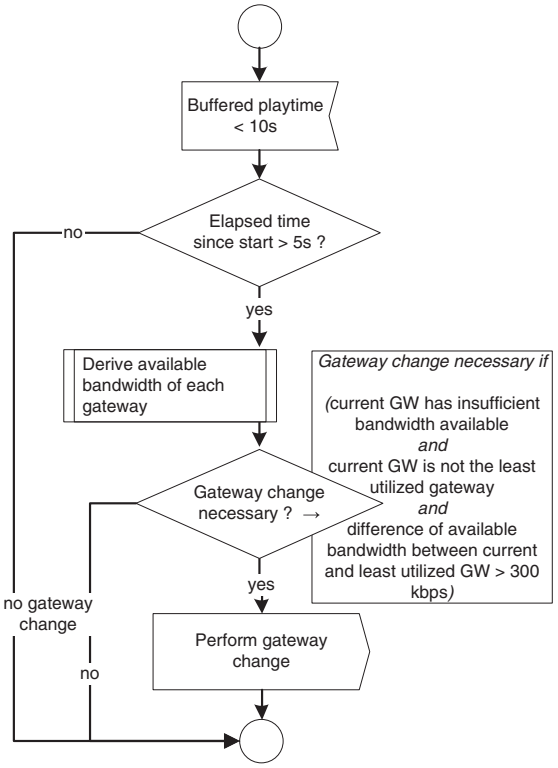


Figure 2. Gateway change algorithm.

happen that a YouTube video request of the video player is redirected in some cases with HTTP error code 302 to a secondary YouTube server due to overload, which costs some additional time.

If all conditions are met, the resource management starts and the available capacity of each gateway is determined. As long as the current gateway has sufficient capacity, as long as the current gateway is the least utilised gateway or as long as the capacity difference between the current and the least utilised gateway is negligibly small (less than 300 kbps), no gateway change is carried out. In all other cases, the flow is allocated to the least utilised gateway.

5.4.2. Network control: buffer-based prioritisation.

When multiple YouTube streams compete for the available capacity of a gateway, the capacity assignment is handled arbitrarily by the TCP protocol control mechanisms. Therefore, it is possible that streams with similar needs get strongly different shares of the available capacity. Consequently, one video might struggle unnecessarily with low buffer sizes making stalling (i.e. interruptions) more likely. To overcome this TCP-caused behaviour means to prioritise struggling streams and to distribute the available capacity more fairly according to the application state that is required.

Table I shows the prioritisation policy, which is performed on each gateway. The video stream is assigned the respective priority 5 down to 1 with 1 being the highest priority. For this action, not only one critical state threshold, that is, buffered playtime is below one certain threshold, is considered. Instead, depending on the current buffered playtime of a YouTube stream, its priority is updated on every status update. The provided thresholds are critical for the resource management and have been obtained empirically as the most adequate values. They must be far enough apart that the system does not tend to overreact and close enough to allow sufficient priority changes, in order to avoid situations where one video is preferred for an excessively long period.

With this algorithm, the bandwidth can be allocated to the flows according to their buffered playtime. All flows of the highest priority class are processed first. The remaining capacity is now available for the flows of the second highest priority class. Again, they are served according to their needs and likewise, the remaining capacity is available for the next lower priority class. This distribution is continued until either no more streams or no more capacity is left. Thus, it is possible that flows of lower priority classes are not assigned any bandwidth at all. As their buffered playtime decreases, consequently, their priority increases and their needs are served again. Currently, no actions are taken to distribute the available bandwidth equally within a priority class.

5.4.3. Service control: quality change.

In case of YouTube video streaming, this resource management action allows to dynamically change the video resolution on request. Depending on the uploaded video, YouTube currently offers 240p (i.e. 240 pixels vertical resolution), 360p, 480p and even high-definition videos with 720p, 1080p or 'original', which means a resolution of up to 4096 × 3072 pixels (4K). Each playback video quality requires different download bandwidths and consequently, a change in the video quality results in a change of the throughput of the YouTube video. This effect is exploited by the resource management action. If there is enough bandwidth available, the video quality is changed to the highest possible quality. However, if the network is congested and the application monitor measures a low buffer level of the YouTube video, a lower quality is suggested for the video to ensure a smooth video playback without stalling. The implementation of the quality change and the service control policy are described in the succeeding text.

To change the quality of a streamed video, the algorithm uses the YouTube player API, which provides the possibility to set the playback quality of the video. The function causes the video to reload at its current position in the new quality just as if the user herself clicked the corresponding button at the video player. The old data is discarded and a new stream is requested from the YouTube servers. Beginning with a new flash video header, the servers start to stream the video in the new quality, that is, the new resolution. Because of the new video stream and because the

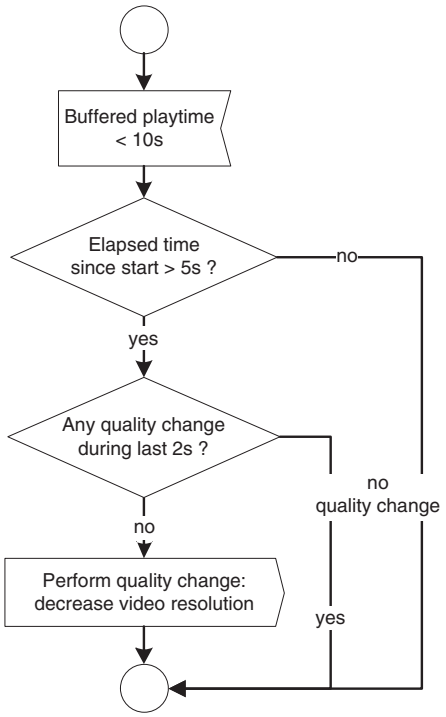


Figure 3. Quality change algorithm.

old data is discarded, every quality change causes a short stalling event but prevents the video from struggling with unsatisfiable needs, which would result in even more and longer stallings.

To determine whether a quality change is necessary, the resource management algorithm performs checks on each critical application state, which are depicted in Figure 3. The quality of the video is switched to the next lower level only if the video playback time is already longer than 5 s and there was no other quality change in the network during the last 2 s. These checks assure that only video streams, which are not in the initial phase and are struggling (i.e. have a small buffer size) are changed by the algorithm.

5.4.4. Combined control.

Although both, network and service controls, have proven their effectiveness in different test cases, for different purposes, combined control actions are required. As a start, two simple strategies are defined as follows:

Network control first. As long as the problem can be solved by the network, only network control is used.

Service control first. As long as a sufficient QoE can be guaranteed, only service control is used.

For example, if the goal is to optimise the overall QoE, the first approach is useful. This means that all possible resources are utilised without considering the costs for transmission. In contrast, if the goal is to reduce the required transmission resources, the second strategy should be preferred. It can be used to a certain extent to rather provide a medium quality for all users than a high quality which may be, from the provider point of view, expensive compared with a lower quality.

6. EVALUATION OF DIFFERENT RESOURCE MANAGEMENT ALGORITHMS FOR YOUTUBE

The resource management framework used for the measurements is based on Aquarema as described in Sections 4 and 5.

For the evaluation, the framework is installed in a wireless mesh testbed, which serves as an access network for clients. The network consists of four mesh nodes, which are connected by WiFi. One of the nodes is the access point (i.e. node to which all clients are connected) and the other three nodes are gateways (i.e. mesh nodes having access to the Internet). The structure of the testbed is the same as depicted in Figure 1. Each gateway has a fixed capacity of 3 Mbps, and thus forms the bottleneck of the network. Compared with the bandwidth of 3 Mbps at the gateway, we assume that the connection between testbed and the YouTube server provides enough capacity, so that it does not have any effect on the measurements. The network monitor tool is installed on each gateway node to report its utilisation and available capacity to the Internet.

Up to four client computers are connected to the access point node by WiFi. They give users the possibility to watch YouTube videos in a browser. On each client, the YouTube application monitor is installed to signal the presence of video streams and to collect information. Additionally, one separate computer within the mesh network hosts the network advisor, which receives all information from both, mesh and application monitors, and decides about resource management actions. The network monitors at the gateways are connected directly to the advisor. The application monitors communicate with the advisor through the access point.

6.1. Reference scenarios

The objective is to evaluate the resource management actions. Therefore, we compare the behaviour without resource management with the behaviour when resource management is enabled. We consider different video qualities and distinguish between synchronous, that is, the videos start at the same time and asynchronous start of YouTube videos. As metric for the evaluation, we focus on the buffered playtime because according to [30, 31], stalling is the factor dominating the QoE of YouTube video streaming.

Table II* lists the combinations that are used in the scenarios with a synchronous start of YouTube videos. In

*A line is highlighted in this table because it is discussed later on in Section 6.3.2 or Section 6.3.3.

Table II. Synchronous video start—no control.

Videos	n_s	t_s	$\overline{bw}$	bw_{tot}
0/0/1	0.00	0.00	0.92	0.69
0/1/0	0.00	0.00	1.61	1.20
0/0/2	0.14	0.61	1.84	1.38
0/1/1	0.14	0.79	2.53	1.90
1/0/0	0.00	0.00	2.55	1.90
0/2/0	0.55	13.19	2.98	2.41
1/0/1	0.92	47.45	2.96	2.59
0/0/4	3.50	20.71	2.98	2.76
1/1/0	1.57	144.00	2.97	3.11
2/0/0	1.57	263.43	2.98	3.80

Table III. Delayed video start—no control.

Videos	n_s	t_s	$\overline{bw}$	bw_{tot}
1/0/1	0.14	2.69	2.98	2.59
0/0/4	0.43	1.93	2.99	2.76
1/1/0	0.84	129.47	2.98	3.11
2/0/0	1.78	247.64	2.99	3.80

the next section, it is shown that the startup (synchronous or asynchronous start) has a big impact on the stalling behaviour due to the pre-buffering pattern of YouTube. Therefore, in Table III*, for comparison, scenarios with a delayed start are defined. In this scenario, the videos start with an interval of 30 s. The first column in the tables indicates how many videos are used. If x is the number of videos in resolution 480p, y is the number of videos in 360p and z is the number of videos in 240p, then $x/y/z$ denotes how many videos in 480p, 360p or 240p are used for one test run. The other columns in the tables show the results within the testbed network as a reference without resource management and if only one gateway is used. The tables show the mean values of number of stalling events, stalling length, used bandwidth and theoretical bandwidth, which were measured in the testbed. The discussion of the different results is carried out in the next section. The table headings contain abbreviations. The meaning of the abbreviations is explained in Table IV. For every combination, at least 20 test runs are performed. The columns in the tables show the average of all test runs.

6.2. Performance investigations of the reference scenarios

In the following, we determine in which situation YouTube encounters problems. In particular, this is the case if the network is overloaded. To allow a practical evaluation of our results, we restrict ourselves to the reference scenarios and our test network, and explain, based on estimations and practical measurements, when a critical situation may occur. Consequently, our results apply for the particular network only. However, the statement and the observations are also valid for other small to medium-sized access networks or other network structures.

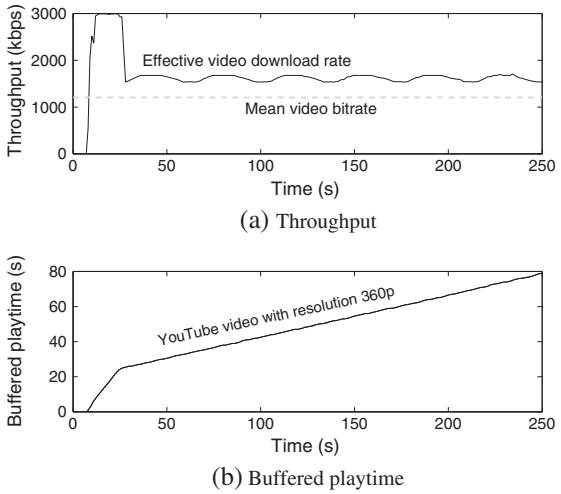
Table IV. Used metrics and their abbreviations.

Abbreviation	Explanation
n_s	Average number of stallings during the video playback.
t_s	Average stalling time during the video playback.
$\overline{bw}$	Average bandwidth used on the gateway.
bw_{tot}	Sum of average video bitrate of all videos.
n_g	Number of conducted gateway changes.
$\|GW\|$	Number of used gateways at the end of the run.
n_p	Number of conducted prioritisation changes.
$\overline{buf}$	Average buffered time of the videos at the end of the run.
n_r	Number of conducted resolution changes.

In the reference scenarios, the same YouTube video with three different sizes of 240p, 360p and 480p is used, which have mean video rates of 0.69, 1.20 and 1.90 Mbps, respectively. When considering only the mean video rate, videos with a total rate of up to 3 Mbps should be able to run smoothly in parallel on a single gateway (e.g. 4×240p with 2.76 Mbps or 1×360p and 2×240p with 2.5 Mbps). The videos, however, are coded differently across the entire playing time using adaptive H.264/MPEG-4 AVC encoding. This may result in variable video bitrates. It means that even if on average, a video may fit on a link, sometimes, a higher temporal data rate is necessary to prevent the video from stalling. A video with high motion at the beginning and a slow 360° video pan over the scenery, for instance, at the end, is highly unequally encoded and requires more data at the beginning than at the end. In order to take this into account, YouTube generally transfers the video content within two transmission phases. At the beginning, the buffer is filled initially with a certain amount of data to compensate for variations in the video coding, see Section 3. This download pattern causes different download rates that have to be considered in the resource management.

The specific download pattern of a YouTube video which is streamed in the testbed is shown in Figure 4. In the upper figure, it can be seen that from the beginning, the video uses the maximal available bandwidth of 3 Mbps and the buffered playtime increases rapidly (lower figure). After the initial burst, the stream is in periodic refill phase and the used bandwidth drops to a rate slightly above the mean video rate. As a consequence, the buffer occupancy increases more slowly.

Our measurements showed that the videos 240p, 360p and 480p request a mean bandwidth of 0.92, 1.61 and 2.55 Mbps, respectively, in the first 5 min of the video. Compared with the mean total video rates, these values are about 25% higher. Considering these higher bandwidth values, the number of videos being able to run on a gateway in parallel need to be reconsidered. For instance, in case of



(a) Throughput

(b) Buffered playtime

Figure 4. Reference measurement of YouTube streaming behaviour for a video with a resolution of 480 × 360 pixels in the testbed.

1×360p and 2×240p despite of the total mean video rate of 2.60 Mbps, the videos try to request a total bandwidth of 2×0.92 Mbps $+ 1 \times 1.61$ Mbps $= 3.45$ Mbps. Obviously, this data rate of 3.45 Mbps is too large for the gateway such that not all demands of the videos can be satisfied.

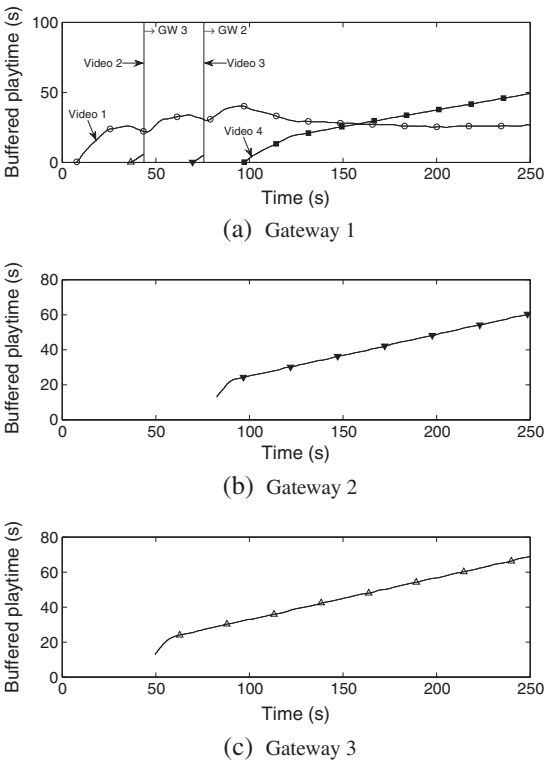
The reference scenarios can be divided into three categories depending on the mean video rate of the videos. For each category, a different kind of resource management is performed later on.

Category 1. Video combinations having a total theoretical bandwidth of less than 2.1 Mbps. The average stalling length is around 0 s. They run smoothly on the gateway. No resource management is required.

Category 2. Video combinations having a total theoretical bandwidth between 2.1 and 3 Mbps. They use the maximal available bandwidth but stalling occurs occasionally. The performance of the individual videos depends strongly on the starting delay and order.

Category 3. Video combinations having a total theoretical bandwidth of more than 3 Mbps. They cannot run smoothly on the gateway and are almost permanently stalling. Therefore, a resource management has to be performed to reduce their required bandwidth. This is addressed later on in Section 6.3.3.

The second category is the most interesting one. If a video combination of this category is put on a gateway and the videos are started at the same time (synchronous start), there are two possible resulting effects. In the first case, one or two videos manage to fill their buffers as desired, resulting the third video being not even able to keep the buffer on a constant level. After a while, one of the videos will start stalling. In contrast, the others fill their buffer excessively. In particular, videos with higher resolutions suffer from this situation due to their higher bandwidth demands. The other possible effect is that all videos share the bandwidth equally. Especially in case of



(a) Gateway 1

(b) Gateway 2

(c) Gateway 3

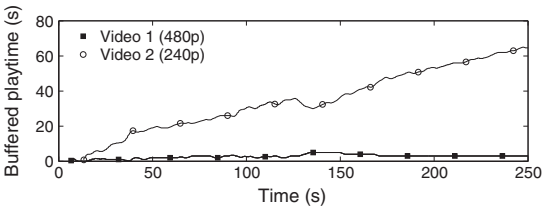
Figure 5. Dynamic gateway change with four 360p videos, subfigures show individual gateways.

different resolutions, this is not the best choice. Instead, the videos should share the bandwidth proportional to their mean video rate because the throughput of videos with higher resolution should be higher than the throughput of low-quality videos, even if progressive download is used and a buffer is filled. Our measurements showed that basically, all combinations without any resource management mechanisms end up in the first described situation.

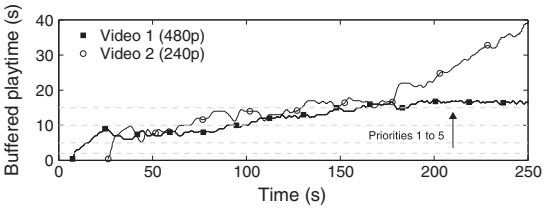
Compared with Table II where the videos are started at the same time, in Table III, the results for the asynchronous start are depicted. The videos have their initial buffering phase one after another, which leads to less stalling. Thus, a simple possibility for resource management is to delay the start of the videos. However, even if a video is started delayed, video combinations having a theoretical bandwidth of more than 3 Mbps cannot run smoothly without stalling.

Table V. Delayed video start—dynamic gateway change.

Videos	n_s	t_s	n_g	$\|GW\|$	$\overline{bw}$	bw_{tot}
0/0/4	0.04	0.17	1.33	2.33	3.71	2.76
1/1/0	0.10	0.20	1.50	2.00	4.18	3.11
1/2/0	0.11	0.83	2.37	3.00	5.74	4.31
0/4/0	0.03	0.15	2.80	3.00	6.16	4.82



(a) Without prioritization



(b) With prioritization

Figure 6. Buffer-based prioritisation with a 480p and 240p video, subfigures show the situation with and without prioritisation.

Table VI. Synchronous video start—buffer-based prioritisation.

Videos	n_s	t_s	n_p	$\overline{bw}$	bw_{tot}
0/2/0	0.18	0.42	36.00	2.99	2.41
0/1/2	0.33	1.64	95.00	2.99	2.59
1/0/1	0.63	7.07	116.73	2.99	2.59

6.3. Evaluating different control approaches

6.3.1. Network control: gateway change.

First, we consider the network control action gateway change. Four 360p videos are started sequentially with an interval of 30 s using the same gateway. Figure 5 shows the temporal progress of the buffered playtime of the videos for the different gateways. At first, a single video is transmitted over Gateway 1 and its playout buffer increases in the usual way. A second video is added to the gateway but its playout buffer cannot be filled properly. Thus, the stream is switched to Gateway 3 where it gets enough capacity to fill its buffer. The same mechanism is applied to two more videos that use Gateway 1 as the initial gateway. The first stream is switched to another gateway. The last stream is not switched because it would not improve the situation. In the end, Gateways 2 and 3 host one stream each, and Gateway 1 hosts two streams and all videos have sufficiently filled buffers.

This example shows that the gateway switching control mechanism helps struggling streams to increase their playout buffers, which avoids stalling of the videos. According to [30, 31], stalling is the factor dominating the QoE for video clips. Consequently, the QoE of the users is increased. From a network perspective, this resource management leads to a balanced load on the available network resources. Compared with common load balancing on network layer, however, we take the instantaneous application

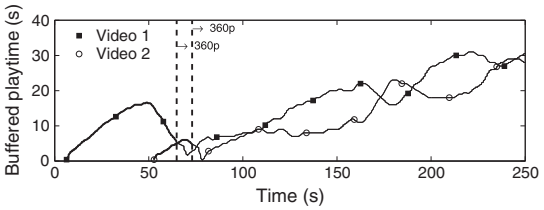


Figure 7. Video resolution change of two YouTube videos.

Table VII. Synchronous video start—video resolution change.

Videos	n_s	t_s	n_r	$\overline{bw}$	bw_{tot}
1/1/0	0.70	15.98	2.10	2.45	3.11

Table VIII. Delayed video start—video resolution change.

Videos	n_s	t_s	n_r	$\overline{bw}$	bw_{tot}
0/2/0	0.08	0.89	0.27	2.86	2.41
1/1/0	0.43	6.54	1.35	2.83	3.11
2/0/0	0.76	17.83	2.86	2.55	3.80

state into account, that is, the current buffered playtime. This means that even situations where videos with different resolutions are used, or when users pause the video or jump within the video can be addressed. For example, if a user is manually selecting a higher resolution, the resource management algorithm will recognise this due to a low buffer state and will relocate the flow to another gateway if capacity is available.

In Table V, the other test runs and their aggregated statistics can be seen. Compared with the reference scenario, the average number of stallings and average stalling lengths have diminished as up to three gateways are used. Especially when the videos are started delayed, the videos face almost no stalling. However, four 480p videos do not fit well on the three gateways of our testbed. Thus, with this combination stalling cannot be prevented.

In general, from a QoE perspective, stalling can be avoided if enough capacity is available to support all YouTube videos. However, situations where a video on one gateway buffers too much data and certain videos suffer from this cannot be avoided. This issue is addressed in the next section.

6.3.2. Network control: buffer-based prioritisation.

In this section, we show that buffer-based prioritisation of video streams helps to avoid stalling. In this example, a 480p and a 240p video stream compete for the bandwidth of the same gateway. Both videos do not fit at the same time on a single gateway and cause each other to stall as shown in Table II, highlighted row.

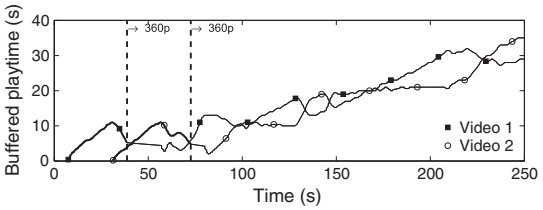


Figure 8. Video resolution change with buffer-based prioritisation for two YouTube videos.

Table IX. Synchronous video start—video resolution change combined with buffer-based prioritisation.

Videos	n_s	t_s	n_p	n_r	$\overline{bw}$	bw_{tot}
2/0/0	1.02	4.52	43.38	2.19	2.94	3.80

Table X. Delayed video start—video resolution change combined with buffer-based prioritisation.

Videos	n_s	t_s	n_p	n_r	$\overline{bw}$	bw_{tot}
2/0/0	0.88	4.25	40.20	2.36	2.82	3.80

In Figure 6, the temporal progress of the buffered playtime is depicted. As described in the reference scenario and as can be seen in Figure 6(a), the 480p suffers most in this situation due to its higher bandwidth demand. The video cannot fill its buffer appropriately and is going to stall. In Figure 6(b), the situation with prioritisation is depicted. The horizontal dashed lines represent the prioritisation classes (cf. Table I). If the buffered playtime of the stream is low, its priority is increased compared with the other stream. Then, the video is able to fill its buffer and the bandwidth requirements of the video are met until its priority becomes lower. Next, if the priority is lower, the other video can fill its buffer. This behaviour continues until the end of the test run. Now, the buffer size of the 480p video oscillates around the last priority threshold and the 240p video continues to fill its buffer. Thus, with buffer-based prioritisation, it is possible that both streams coexist and none has a critically empty playout buffer.

In Table VI, it can be seen that in this example, the average stalling length decreases from 47.45 s (cf. Table II, highlighted line) down to 7.07 s. With the other combinations in the test scenario, the stalling decreases, too. This shows that buffer-based prioritisation as a network control mechanism works well for our test network and is able to avoid TCP-caused problems with bandwidth sharing. With respect to the QoE, this method allows an increase in QoE because a YouTube video that is almost stalling all the time can be supported without stalling, assuming that the available capacity is enough for all YouTube videos.

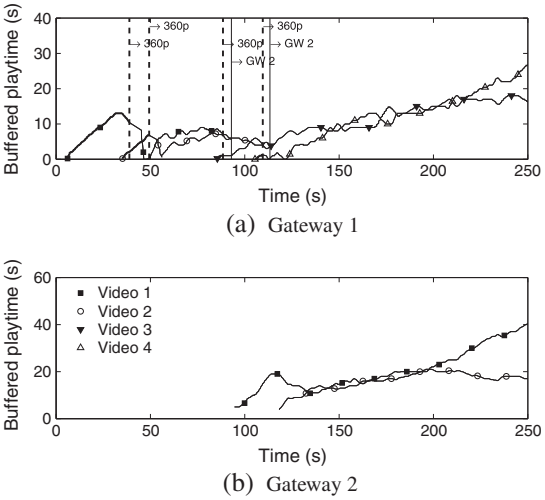


Figure 9. Combined control with policy Moderate Mix, subfigures show the individual gateways (gateway 3 is not used and therefore, not displayed).

6.3.3. Service control: video resolution change.

Now, we want to take a look at the performance of service control. The effects of video resolution change can be seen in Figure 7 in which two 480p videos are started sequentially with an interval of 30 s. The reference scenario (cf. Table III, highlighted row) shows that in a normal situation, the videos would stall permanently. In this scenario, service control is enabled which means that the video resolution is scaled down if the buffer occupancy drops below the control threshold. The figure shows that the 480p videos are changed to 360p one after another. Two 360p videos fit on a single gateway and each video is able to fill its playout buffer. Thus, in this case, almost no stalling occurs and the video streams can coexist in the network. We have to point out that in our implementation, every resolution change (i.e. the start of a new video stream) causes a single stalling event, which could be prevented by a smarter video player. Instead of discarding all data, the smarter player could start the new stream early enough and switch the resolution seamlessly after playing out the whole buffer.

From a network point of view, a change to a lower resolution results in lower bandwidth requirements of the YouTube video. This has only a minor effect on the QoE (please refer to [30, 31] for a more refined analysis) but avoids stalling, which in turn avoids a severe QoE degradation. In fact, this resource management action is particularly useful in overload situations. It works, however, only if the YouTube video is available in different resolutions.

In Tables VII and VIII, the aggregated statistics of the test runs are shown. It can be seen that the service control mechanism is useful as it helps the videos to fill their buffers to an adequate level. Stalling is short (especially with delayed video start) and could be fully prevented by a smarter player. Moreover, with service control, more video

Table XI. Combined control.

Policy	n_s	t_s	n_g	$\|GW\|$	n_p	n_r	$\overline{buf}$	$\overline{bw}$	bw_{tot}
Network control first	0.57	4.03	2.13	3.00	76.88	2.03	43.27	7.55	7.61
Strict service control first	1.04	10.47	1.00	2.00	77.92	8.00	52.93	3.73	7.61
Service control first	1.54	11.96	0.00	1.00	147.61	8.00	7.65	2.20	7.61
Moderate mix	0.86	6.93	2.32	2.00	87.53	4.00	34.86	5.81	7.61

streams fit on the gateway than in the reference scenario. Thus, more users can be served at the same time in the test network.

This resource management action benefits, in our test network, in particular, from the use of prioritisation. Because of the YouTube streaming behaviour with TCP, service control overreactions, that is, too many and unnecessary resolution changes, might occur. To reduce the number of these overreactions, it turned out to be helpful to additionally apply buffer-based prioritisation. In Figure 8, again, two 480p videos are started on the same gateway. This time, the buffer-based prioritisation is activated too. It turns out that both video resolutions are changed down to 360p and that the buffers of the videos are filled faster. The aggregated statistics of this example can be seen in Tables IX and X. In our test runs (delayed start), the average number of resolution changes dropped from 2.86 (cf. Table VIII) down to 2.36 by additional prioritisation. Furthermore, the average stalling length decreased from 17.83 s down to 4.25 s, which further indicates the possible gain of a combined control strategy, which is covered in the next section.

6.3.4. Combined control.

To investigate the performance of a combination of the separately operating mechanisms, we consider the following example video combination. Four 480p videos are started in our testbed on the same gateway, which would result in a heavy stalling according to our reference scenario without any control mechanisms. With combined control, the following three different strategies are examined.

Policy 1: network control first. The four videos are distributed among the three available gateways. On one gateway, two video streams remain, which exceeds the capacity of the gateway. Thus, the resolution of the two videos are changed to 360p. Almost no stalling occurred and all videos could fill their buffers. This strategy reacts quite fast. However, many resources are needed.

Policy 2: service control first. The videos are scaled down to a lower resolution first. As even four 360p videos do not fit on the gateway, the service control is applied again. Then, all videos have a resolution of 240p. As no more service control is possible, one stream is switched to another gateway. This stream could then be switched to a higher resolution again as the capacity of the gateway is sufficient. If prioritisation is additionally enabled, it is even possible to use only a single gateway without much additional stalling. This 'strict' gateway minimisation would be the most resource efficient strategy. However, there is a trade-off between resource utilisation and buffer occupancy (i.e. risk of stalling).

Policy 3: Moderate mix. This example is depicted in Figure 9. All videos are scaled down to 360p first. Then, network control is enabled and two streams are switched to another gateway. Thus, again, two gateways are used but the videos can be kept on a higher resolution. This strategy address the trade-off between network control first and service control first.

The results of all strategies are summarised in Table XI. In case of our test scenario, the lowest number of stallings is achieved with the network control first strategy. Here, all three gateways are activated, which yields to a higher resource utilisation compared with the other policies. Assuming that a quality of 360p is sufficient for an acceptable QoE for YouTube video streaming, the best trade-off between QoE and utilised network resources can be achieved with the moderate mix strategy.

7. CONCLUDING REMARKS

In this paper, we evaluated the impact of different resource management concepts on the user perceived quality for YouTube video streaming. The results illustrate that the application-aware resource management can efficiently increase both, the resource utilisation as well as the perceived quality. This requires to be, on the one hand, aware of the applications running in the network and on the other hand, to be able to perform changes on both, the service and network side. On the network side, the load can be balanced on different gateways, if available, or the prioritisation of the streams can be dynamically changed. Our testbed results show here that the resources are efficiently utilised and more YouTube users can be supported. The second option, service control, allows to change the resolution of the YouTube video. A lower resolution results in a lower bandwidth with only a minor degradation of the QoE [30]. Thus, if no more resources are available or the provider wants to reduce its operational expenditure, service control is the best choice.

Finally, the best trade-off between QoE and resource efficiency can be achieved using a combined control approach. Our findings here are that a strategy using a moderate mix of network and service control helps to keep the QoE on a high level without using too much resources in our test network and thus, reducing the energy consumption and the operational expenditure.

In future work, we will evaluate the performance in terms of stalling time by using a smarter video player, which allows to switch the video resolution without any interruption.

ACKNOWLEDGEMENTS

The authors are grateful to Florian Mayer, Sebastian Deschner and Andreas Blenk for their support and their implementation work. The authors are also indebted to Tobias Hoßfeld for his help in modelling the QoE influences of resource management actions on YouTube. Furthermore, the authors would like to thank Dirk Staehle for the many comments and improvements on the text. This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (Förderkennzeichen 01 BK 0800, GLab) and by the German Research Foundation under grant TR 257/28-2 (FunkOFDMA).

REFERENCES

1. Cisco. Cisco visual networking index: forecast and methodology, 2011-2016, 2012.
2. Wang Z, Crowcroft J. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on Selected Areas in Communications* 1996; **14**(7): 1228–1234.
3. Foster I, Roy A, Sander V. A quality of service architecture that combines resource reservation and application adaptation. In *International Workshop on Quality of Service (IWQOS)*. IEEE: Westin William Penn, Pittsburgh, 2000; 181–188.
4. Crawley E, Nair R, Rajagopalan B, Sandick H. A Framework for QoS-based routing in the Internet, 1998.
5. Nichols K, Blake S, Baker F, Black D. RFC 2474: definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers, 1998.
6. Klotz J, Knabe F, Huppert C. Resource allocation algorithms for minimum rates scheduling in mimo-ofdm systems. *European Transactions on Telecommunications* 2010; **21**(5): 449–457.
7. Gross J, Klaue J, Karl H, Wolisz A. Cross-layer optimization of OFDM transmission systems for MPEG-4 video streaming. *Computer Communications* 2004; **27**(11): 1044–1055.
8. Khan S, Peng Y, Steinbach E, Sgroi M, Kellerer W. Application-driven cross-layer optimization for video streaming over wireless networks. *IEEE Communications Magazine* 2006; **44**(1): 122–130.
9. Ameigeiras P, Ramos-Munoz JJ, Navarro-Ortiz J, Mogensen P, Lopez-Soler JM. QoE oriented cross-layer design of a resource allocation algorithm in beyond 3G systems. *Computer Communications* 2010; **33**(5): 571–582. DOI: 10.1016/j.comcom.2009.10.016.
10. Pries R, Hock D, Staehle D. QoE based bandwidth management supporting real time flows in IEEE 802.11 mesh networks. *Praxis der Informationsverarbeitung und Kommunikation* 2010; **32**(4): 235–241.
11. Reis A, Chakareski J, Kassler A, Sargento S. Quality of experience optimized scheduling in multi-service wireless mesh networks. In *IEEE Conference on Image Processing (ICIP)*. IEEE: Hong Kong, 2010; 3233–3236.
12. ITU-T recommendation P.10/G.100 amendment 2 definition of quality of experience (QoE), July 2008.
13. Shin J, Kim JW, Kuo C. Quality-of-service mapping mechanism for packet video in differentiated services network. *Multimedia, IEEE Transactions on* 2001; **3**(2): 219–231.
14. IEEE. IEEE standard for local and metropolitan area networks; part 16: air interface for fixed and mobile broadband wireless access systems, February 2006.
15. Huang C, Juan H, Lin M, Chang C. Radio resource management of heterogeneous services in mobile WiMAX systems [Radio Resource Management and Protocol Engineering for IEEE 802.16]. *IEEE Wireless Communications* 2007; **14**(1): 20–26.
16. Fiedler M, Hoßfeld T, Tran-Gia P. A generic quantitative relationship between quality of experience and quality of service. *IEEE Network, Special Issue on Improving QoE for Network Services* 2010; **24**(2): 36–41.
17. Agboma F, Liotta A. QoE-aware QoS management. In *6th International Conference on Advances in Mobile Computing and Multimedia*. ACM: Linz, Austria, 24–26 November 2008; 111–116.
18. Thakolsri S, Khan S, Steinbach E, Kellerer W. QoE-driven cross-layer optimization for high speed downlink packet access. *Journal of Communications* 2009; **4**(9): 669–680.
19. Staehle B, Wamser F, Hirth M, Stezenbach D, Staehle D. AquareYoum: application- and quality of experience-aware resource management for youtube in wireless mesh networks. *Praxis der Informationsverarbeitung und Kommunikation* 2011; **34**(3): 144–148.
20. Xiao M, Shroff N, Chong E. A utility-based power-control scheme in wireless cellular systems. *IEEE/ACM Transactions on Networking* 2003; **11**(2): 210–221.
21. Andrews M, Qian L, Stolyar A. Optimal utility based multi-user throughput allocation subject to throughput constraints, In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, Vol. 4, Miami, FL, USA, 13–17 March 2005; 2415–2424.

22. Song G, Li Y. Utility-based resource allocation and scheduling in ofdm-based wireless broadband networks. *IEEE Communications Magazine* 2005; **43**(12): 127–134.
23. Saul A. Simple optimization algorithm for mos-based resource assignment, In *IEEE 67th Vehicular Technology Conference: VTC2008-Spring*, Marina Bay, Singapore, 11–14 May 2008; 1766–1770.
24. Pei X, Zhu G, Wang Q, Qu D, Liu J. Economic model-based radio resource management with qos guarantees in the cdma uplink. *European Transactions on Telecommunications* 2010; **21**(2): 178–186.
25. International Organization for Standardization/ International Electrotechnical Commission (ISO/IEC). ISO/IEC 23009-1:2012: dynamic adaptive streaming over HTTP (DASH), April 2012.
26. Alcock S, Nelson R. Application flow control in youtube video streams. *ACM SIGCOMM Computer Communication Review* 2011; **41**(2): 24–30.
27. Rao A, Legout A, Lim Y, Towsley D, Barakat C, Dabbous W. Network characteristics of video streaming traffic, In *7th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, Tokyo, Japan, December 6–9, 2011; 25:1–25:12, DOI: 10.1145/2079296.2079321.
28. Wamser F, Staehle D, Prokopec J, Maeder A, Tran-Gia P. Utilizing buffered youtube playtime for QoE-oriented scheduling in OFDMA networks, In *International Teletraffic Congress (ITC)*, Krakow, Poland, 2012; 1–8.
29. Ameigeiras P, Ramos-Munoz JJ, Navarro-Ortiz J, Lopez-Soler JM. Analysis and modelling of youtube traffic. *European Transactions on Telecommunications* 2012; **23**(4): 360–377.
30. Dobrian F, Awan A, Joseph D, Ganjam A, Zhan J, Sekar V, Stoica I, Zhang H. Understanding the impact of video quality on user engagement, In *ACM SIGCOMM*, Toronto, Ontario, Canada, August 15–19, 2011; 362–373.
31. Hoßfeld T, Schatz R, Seufert M, Hirth M, Zinner T, Tran-Gia P. Quantification of youtube QoE via crowdsourcing, In *IEEE International Workshop on Multimedia Quality of Experience*, Dana Point, CA, USA, 2011; 494–499.
32. Google. YouTube Player API Reference, March 2012.
33. Staehle B, Hirth M, Pries R, Wamser F, Staehle D. YoMo: a youtube application comfort monitoring tool, In *New Dimensions in the Assessment and Support of Quality of Experience for Multimedia Applications*, Tampere, Finland, 2010; 21–23.