University of Würzburg
Institute of Computer Science
Research Report Series

YoMo: A YouTube Application Comfort Monitoring Tool

Barbara Staehle, Matthias Hirth, Florian Wamser, Rastin Pries, Dirk Staehle

Report No. 467 March 2010

University of Wŕzburg, Germany
Institute of Computer Science
Chair of Computer Networks
Am Hubland, D-97074 Würzburg, Germany
phone: (+49) 931-31-86655, fax: (+49) 931-31-86632
{bstaehle|matthias.hirth|florian.wamser|pries|dstaehle}@informatik.uni-wuerzburg.de

YoMo: A YouTube Application Comfort Monitoring Tool

Barbara Staehle, Matthias Hirth, Florian Wamser, Rastin Pries, Dirk Staehle
University of Wŕzburg, Germany
Institute of Computer Science
Chair of Computer Networks
Am Hubland, D-97074 Würzburg, Germany
phone: (+49) 931-31-86655, fax: (+49) 931-31-86632
{bstaehle|matthias.hirth|florian.wamser|pries|dstaehle}@informatik.uni-wuerzburg.de

Abstract

Today's Internet does not offer any quality level beyond best effort for the majority of applications used by a private customer. In particular, this applies for wire-line or wireless access networks which currently constitute the bottleneck of the communication infrastructure. Following the principle of economic traffic management we propose a collaboration of user and network control in order to achieve a win-win situation through an improved application-aware resource management. In order to do so, a tool runs at the client that continuously monitors the application comfort and communicates it to the network control. The application comfort quantifies how well an application is running. In our example application, YouTube video streaming, the application comfort is expressed by the buffered video playtime. As long as there is enough playtime buffered, the application comfort is high. YoMo, the YouTube monitoring tool, runs on the client, detects YouTube videos and monitors their application comfort and signals it to the resource management tool. An example how to utilize this information for an improved resource management ensuring a high user satisfaction when playing the YouTube video is demonstrated in an IEEE 802.11 based mesh testbed.

1 Introduction

In today's consumer Internet, most traffic is transmitted on a best effort basis without support for the quality requirements of the application. There are no service guarantees for the predominant consumer Internet traffic which is composed of applications like P2P or client-server file sharing, web browsing, or video streaming which make up for more than 80% of today's traffic [1, 2, 3]. The primary reason for this is not the lack of technical solutions [4, 5] enforcing quality guarantees but that the prerequisite, that the network knows the quality requirement of the different Internet applications, is not fulfilled.

As an example, recent and future fixed and mobile wireless access networks like 3GPP UMTS and LTE as well as IEEE 802.16 based WiMAX, offer connections with strict QoS support. These connections are called bearers in 3GPP and service flows in IEEE

802.16. Ludwig et al [5] distinguish session-based services and none-session-based services. Session-based services, e.g. multimedia services established via the IP multimedia subsystem (IMS), are transported over connections with appropriate QoS parameters. Non-session-based services, in particular the "Internet Access" service, are transported via a connection with default QoS parameters that offer only a best effort service. All above mentioned Internet applications rely on this Internet access service, and as a consequence, they are multiplexed on a single best effort connection and use the access network as a mere bit pipe.

Nonetheless, Internet applications like VoIP (Skype), Gaming, Video streaming (YouTube), or even simple web browsing have more or less strict quality requirements. The prerequisite for providing QoS support for Internet applications is first to detect the flows/packets belonging to the application in the packet stream and second to determine appropriate quality parameters. The detection is currently done using deep packet inspection (DPI) [6] , i.e. by looking not only at TCP/IP headers but also at application headers or even using heuristics on the observed traffic pattern and TCP connection usage. The evolved packet system of 3GPP LTE [7, 5], introduces the possibility of a network-initiated QoS bearer establishment if the user application communicates to the network via a standard socket-API and not the vendor-specific QoS-API which mostly means that the application uses the Internet access services. A DPI box observes the traffic at the gateway and if an application with known quality requirements is detected, the network establishes a radio bearer if appropriate QoS parameters for the application are available in the user profile.

However, DPI is rather challenging since the browser tends to become the user's interface to the Internet for an increasing number of applications like videos (YouTube), large file downloads (Rapidshare), browser games, etc. All these applications are transported via HTTP and sum up to around 60% of the traffic for residential broadband Internet access [3, 8]. Additionally, P2P applications and even Skype use port 80, to tunnel through NATs or to hide from easy detection. All this makes the classification of packets within the network rather complicated. Moreover, DPI typically only detects applications. Deriving appropriate QoS parameters for an application from the traffic flow observed in the network is even more complex, in particular for reactive TCP traffic. Thus, only a limited set of applications with a priori known QoS parameters can be supported by establishing approapriate connections.

An alternative concept how to learn about active applications and their QoS requirements at the network control, is gathering this information at the client and communicating it to the network. The 3GPP evolved packet system, e.g., also specifies the possibility of a network-initiated QoS bearer establishment where the network control receives the information about the application from the user and not through deep packet inspection. Consequently, the QoS connection is not established using the QoS-API at the terminal but by the network itself. The advantage of this solution is that the client does not need to be aware of the access specific QoS parameters but may communicate access agnostic QoS requirements that the network control translates into corresponding QoS parameters. The evident solution that an application directly communicates with the network control again requires a modification of the application. Instead, we propose

to implement a generic tool installed by the user that monitors the applications on the client, derives application QoS requirements, and communicates this information to the network.

This approach, however, has two prerequisites: First, appropriate QoS requirements must be available or measurable when the application starts. E.g. for videos with variable bit rate codecs, the precise bandwidth is not known such that the QoS requirements are typically overdimensioned which critical in especially in wireless networks. Second, the access network must be able to guarantee strict QoS. E.g. guaranteeing strict QoS in IEEE 802.11 wireless mesh access networks is very complex if possible at all. The usage of strict QoS parameters can be avoided by using a quality of experience (QoE) based resource management [9, 10, 11] that continuously adapts the network resources to quality feedback from the application. QoE [12, 13] is a measure for the subjective quality that a user experiences. In contrast to QoS, the QoE depends not only on the network but also on the user's environment. However, the environmental impact is often neglected and frequently QoE is interpreted or approximated as a function of measurable QoS parameters like latency, jitter, or bandwidth, see e.g. [13]. In principle there are two ideas how to perform a QoE based resource management. The first one is to assign network resources such that the total QoE in the network is maximized [14, 9]. This approach requires that a QoE evaluation for all applications and traffic flows is available. Alternatively, the network resources are partitioned into a part with and a part without QoE based resource management. The other approach is to continuously monitor the QoE of some applications [15] and to change the resource management, accordingly. An overview of QoE based resource management schemes is given in Section 2.

In this paper, we follow the latter approach and apply it to a YouTube video streamed over a wireless mesh access network. Most QoE models are proposed for VoIP [16, 17, 18, 19] or video streaming [20] transported over a UDP connection. Consequently, QoE degradation results from delay, jitter, and packet loss and a scalable instantaneous QoE can be measured. YouTube videos, however, are transported via TCP, i.e. no packets are lost and all packets are delivered in order. QoE degradation occurs when the playtime buffer is empty and the video stalls. Thus, we can only measure a binary instantaneous QoE: either the video plays or the video stalls. As an extension, we introduce the concept of Application Comfort (AC) that not only considers the currently observed QoE but generally evaluates the current status of the application with respect to future performance. For a YouTube video, AC is expressed by the buffered playtime. Let us consider an example to make the difference between QoE and AC clear: The instantaneous QoE of a YouTube video that started quickly and played continuously up to the observed time has an optimal QoE even if it is about to stall in the next moment because the playout buffer is almost empty. The AC, however, is bad since it considers the playout buffer and predicts the future QoE development in terms of an imminent stall time. Thus, monitoring the AC allows to react to an imminent QoE degradation before it occurs. For most UDP based applications, however, AC is identical to QoE unless it is possible to really predict future packet losses or delays.

We give an example how an application comfort monitoring (ACM) tool running at the client may extract application layer information and interact with the network at

the example of a YouTube video stream in a wireless mesh network. The YouTube monitoring tool, YoMo, runs at the client and (1) detects YouTube streams, (2) at the beginning of the YouTube stream extracts metadata including duration, video and audio rate, and (3) continuously monitors the AC in terms of the playtime buffer while the video is playing. We chose YouTube as the currently most popular video portal [3, 8], but the presented tool works for all types of flash videos which are estimated to become one of the major traffic sources in the future Internet [21].

Wireless mesh networks (WMNs) are the most complex types of access networks with respect to resource management and performance guarantees. The resource management, see e.g. [22], covers routing including gateway selection, channel and interface allocation in multi-radio multi-channel mesh networks, prioritization of medium access through contention parameters as in IEEE 802.11, and finally traffic shaping. Performance guarantees are difficult to achieve since a link between two nodes typically does not have a constant capacity but it shares the radio capacity with the surrounding links due to interference and resulting collisions. We chose the example of WMNs since (1) WMNs are an attractive low-cost alternative for broadband wireless Internet access, (2) practically implementing resource management strategies is possible on open source mesh nodes, and (3) WMNs provide an enormous potential for an improved radio resource management when application layer information is available. The focus of this paper, however, is the monitoring tool YoMo running at the client. The proposed resource management algorithm is not the main contribution of the paper and shall mainly serve as a proof of concept and a first step towards a QoE or AC based resource management architecture for wireless mesh networks.

The rest of this paper is structured as follows: In Section 2 we give an overview of the related work on application layer and QoE monitoring as well as frameworks that try to use application layer information for an improved resource management. Section 3 gives an overview of Flash video and how YoMo works. In Section 4, we introduce the resource management tool OTC and demonstrate its performance at an example scenario in Section 5. In Section 6 we summarize the contributions of this paper and give an outlook to future work.

2 Related Work

In this section, we review contributions related to this work. We start by describing a number of recently published radio resource management techniques and frameworks. As most network management decisions are taken in order to avoid a degradation of the user experience, we subsequently discuss a number of approaches suitable for deriving the QoE from network or application layer parameters.

The IEEE is currently developing standards towards an improved usage of radio resources in heterogeneous wireless access network. The IEEE 1900.4 standard [23] was published in February 2009 by the newly founded 'IEEE Standards Coordinating Committee 41" (SCC41) that focuses on standards projects in the areas of dynamic spectrum access, cognitive radio, interference management, coordination of wireless systems,

advanced spectrum management, and policy languages for next generation radio systems [24]. The IEEE 1900.4 standard hence defines a management system which allows the distributed optimization of radio resource usage and improves the QoS in heterogeneous wireless networks. The IEEE 802.21 working group specifies a standard for media independent handover in heterogeneous radio access networks. Both approaches have in common that they try to optimize resource usage in heterogeneous wireless networks, establish a signaling framework between terminals and network, and try to make context-aware resource management decisions. Among other factors, the context includes required QoS levels, radio network and terminal capabilities as well as measurements both from the terminal and the network [25, 26]. With respect to this paper, the following two points are remarkable: On the one hand, both approaches rely on services with strict QoS parameter settings, i.e. session-based services. A proper treatment of applications using the non-session-based "Internet access" service without precisely specified QoS levels, i.e. the common Internet applications, is not supported. On the other hand, both standards are context-aware and specify an interface for information exchange between terminal and network. Consequently, the information gathered by the ACM tool contributes to the terminal's context and essential information for a good resource management for common Internet applications.

A similar idea is presented in [27] where Ong and Khang introduce a cooperative radio resource management framework which shall enable seamless multimedia service delivery. The framework consists of two components: a distributed terminal-oriented network-assisted handover architecture is intended to support the convergence of heterogeneous wireless access networks by making the exchange of QoS information among them possible. Furthermore, a generic dynamic access network selection algorithm allows the cooperation between network terminals for making more informed access or handover decisions. This means that a terminal can take a handover decision based on the QoS information broadcasted by the access networks instead of measuring the channel quality itself. Both concepts together allow to maintain a QoS-balanced system, i.e. to reach a state with a similar level of QoS in all access networks. A simulation study demonstrates that the QoS broadcasting mechanism may be implemented within the IEEE 802.11 beacon frames and that the scheme allows to decrease the uplink and downlink packet delay and packet loss rates while maximizing the network throughput in a WLAN with two access points. Kassar et al. [28] also focus on vertical handover. They introduce an intelligent management system which is responsible for transparently switching between cellular and non-cellular Internet access networks. This module handles the handover information gathering, handover decision, and handover execution phase. Information is gathered from terminal and network side. This allows to decide the necessity of handover according to user preferences or velocity on the one hand and bandwidth or network coverage on the other hand. Terminal and network side characteristics like user preferences or terminal battery status and cost or QoS characteristics respectively are also used to assist during the decision for an access network. The authors were able to show that in a simulation with one mobile user and a 3G/WLAN environment the scheme works as well as a traditional scheme.

A comparable but more general framework has been presented by Bullot et al. [29]

who propose an architecture for decentralized network management and control. The architecture consists of a data, a control, and a management plane, and a piloting system. What is novel about this proposal is that the piloting system decides which system parameters have to be measured by the control plane and which parameters have to be given to which algorithms. The advantage of this solution is that the control algorithms like routing, mobility management etc. can use information which is stored by the piloting plane and perhaps used by other algorithms, too. In traditional architectures in contrast each algorithm is responsible for retrieving the parameters it requires. The piloting plane is realized in a distributed manner. Cooperating agents are running on each network element and exchange their own view on the network with neighboring agents. As an evaluation of their architecture, the authors simulate a mobile-initiated handover in a setting with three 802.11 access points and up to 30 mobiles. The study demonstrates that the traditional policy for such a handover, the received signal strength, is outperformed in terms of rejected VoIP calls and increased end-to-end delay by a distributed decision making process taking into account the load of the APs.

For mesh networks which do in general not have mobile nodes nor allow the user to select among several access networks, handover is not an option for resource management. However, many other possibilities like node placement, channel and routing allocation, or MAC layer optimization exist. Akyildiz et al. [22] give a good overview on the existing alternatives, we refrain from an exhaustive enumeration and introduce as an example the work of Pries et al. [11]. The authors propose to dynamically constrain the bandwidth of best effort traffic in order to ensure the quality of service requirements of multimedia applications. This is realized by the interaction of a Traffic Observer (TO) and a Traffic Controller (TC). An instance of the TO is running on each mesh node and continuously monitors the QoE of the VoIP flows in terms of the Mean Opinion Score (MOS) as a function of the measured packet loss and delay. As soon as the TO detects that the MOS becomes bad, the TC running on the same and on the neighboring nodes throttle the interfering best effort traffic. The evaluation of this concept in a mesh testbed showed that TO and TC together allow to maintain a satisfying MOS score even in the presence of disturbing traffic.

The approach of [11] to use a user experience-oriented metric for managing decision is another point which distinguishes this work from the previously introduced contributions. The authors use the exponential relation between packet loss and MOS discovered by Hoßfeld et al. [30] to find a mapping between a measurable parameter and the user experience. Many similar contributions [16, 17, 18, 19, 20] exist which allow to derive the QoE for a specific application from a QoS parameter. An exemplary work is the one of Mohamed et al. [19] who obtain a mapping function between the loss rate and the mean size of loss bursts and the VoIP user QoE in terms of MOS. For this purpose, a number of test persons evaluate VoIP samples which were transmitted over varying link conditions. Those rated samples are used to teach a random neural network the interdependency between QoS and QoE.

Piamrat et al. [31] use this idea for an admission control mechanism which ensures the QoE of multimedia users in a WLAN environment. In a simulation study the authors are able to show that this scheme is able to guarantee a high user satisfaction while

admitting a higher number of flows and achieving a better bandwidth utilization as loss-base admission control policies. Bohnert et al [32] also propose an QoE based admission control scheme. They modify the QoE model for speech quality proposed by Raake [16] in order to obtain a QoE estimation for an aggregate of VoIP calls. An evaluation of the QoE based admission control scheme for an IEEE 802.16 access network shows that the scheme successfully avoids long periods of user dissatisfaction. In [33], Bohnert et al. propose a QoE aware scheduler for VoIP calls in an IEEE 802.16 access network. The idea is to give priority to VoIP calls that instantaneously experience a low QoE, i.e. those calls that recently suffered from packet loss. The authors demonstrate by a simulation study that the QoE based scheduler outperforms a simple FIFO scheduler and performs similar to a channel-aware scheduler that gives priority to mobiles with higher SNR.

Khan et al. [14] present a resource management scheme that optimizes the average QoE for video streams in a wireless network using cross-layer design. The video QoE metric - though not explicitly referred to as QoE in the paper - uses the PSNR model. The resource management scheme optimizes both network resources and video parameters in order to maximize the average PSNR experienced by the video streams. In [9], the model is extended to three applications, VoIP, video, and FTP, and a concrete resource management scheme for HSDPA is described.

The above approaches are just some examples which use QoE as a basis for management decisions. The methods of deriving it from QoS parameters is however problematic, as a mapping between QoS and QoE is highly application dependent. An earlier experiment [34] with human test persons who remotely used Microsoft Word even revealed that the satisfaction of the users depends on the task they are doing, too. Some users, who were typing for instance, were totally satisfied with a lossy and very slow connection, whereas users who used the scroll bar were already annoyed if the connection speed was at a medium, level. The experiment also showed that the time which is required for completing the task, hence the application comfort in our terminology, is a good metric for the user satisfaction. Michaut et al. [35] give an overview on application-oriented measurement tools and techniques, which they do understand to be classical network characteristics like bandwidth, one-way delay or round-trip time. This is a good example, as to our knowledge, no work exists which tries to find a relation between the application comfort and the network condition.

3 Monitoring the YouTube Application Comfort

In 2008, Wamser et al. [3] conducted a study of the Internet usage of 250 households connect by a broadband wireless access network. The traffic classification revealed that nearly 1/4 of the consumed bandwidth belongs to streaming traffic. This value is even more impressive if one is aware that during a similar measurement the authors conducted in 2007, only 4% of streaming traffic were found. An analysis of the streaming traffic in turn revealed that 30% of it is Flash video (FLV) traffic, the technology used by YouTube. A tool which is able to monitor the application comfort of YouTube and FLV

video streaming portals in general, is hence able to provide information on roughly 10% of the traffic volume of a WMN.

In the remainder of this section we describe YoMo, a YouTube application comfort monitoring tool. YoMo has to fulfill all tasks which are mandatory for an ACM: It has to firstly, detect that a YouTube flow is existing which has to be monitored. It has to secondly collect as many information as possible on the YouTube flow and has to thirdly to monitor the YouTube AC. To make our approach more easy to understand, we first of all analyze the technology behind YouTube in Section 3.1, before we introduce the core YoMo ideas and their implementation in Section 3.2.

3.1 The Technology Behind YouTube

The YouTube player is a proprietary Flash application[1], which concurrently plays an FLV file and downloads it via HTTP. At the beginning of this so-called pseudo streaming, the client fills an internal buffer and starts the playback of the video as soon as a minimum buffer level is reached. During the time of simultaneous playback and downloading, the buffer level is dependent on the download bandwidth and the video rate. As long as the download bandwidth is higher than the video rate, the buffer increases, otherwise it shrinks. If the buffer runs empty, the video stalls.

A stalling of the video can be detected by a change of the YouTube player state from "playing" to "buffering". The player state is hidden to the normal user, but can be retrieved from the YouTube API which may be accessed by developers who embed the YouTube player into web pages. Furthermore, the API allows to control the video playback and to get information about the currently displayed video.

Each YouTube video is encoded as an FLV file which is a container format for media files developed by Adobe Systems[2] . An FLV file encapsulates synchronized audio and video streams, and is divided into a header and a body. The header starts with an FLV signature and contains information about the available tags in the file. The body consists of tags and separators. The tags encapsulate the data from the streams and contain information on their payload. This information includes the payload type, the length of the payload, and the time to which the tag payload applies. FLV files may also contain metadata encapsulated in a tag with a script data payload. The available properties depend on the software used for the FLV encoding and may include the duration of the video, the audio and video rate, and the file size.

3.2 The Functionality of YoMo

The YouTube player opens a new TCP connection each time it downloads a new FLV file. Each FLV file begins with an FLV-identifier which is detected by YoMo which constantly monitors the client's incoming traffic. Once a flow containing FLV data is recognized, the data is continuously parsed in order to retrieve the available meta information from the FLV file. The first two tasks of an ACM, detecting a YouTube flow and extracting

[1] http://www.adobe.com/devnet/flashplayer/, last accessed 02/10
[2] http://www.adobe.com/devnet/flv/, last accessed 02/10

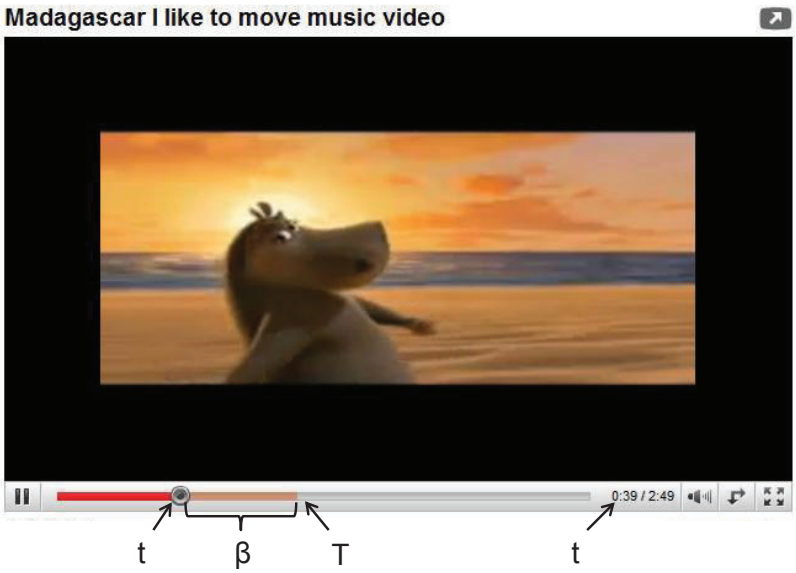


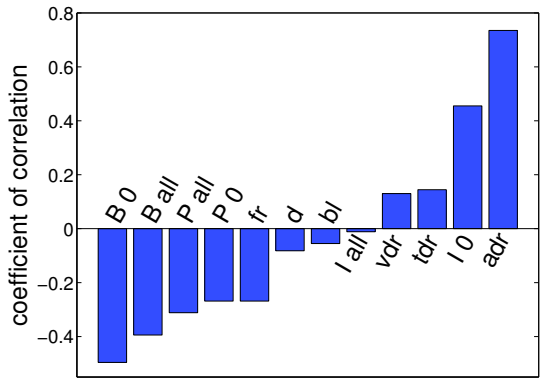
Figure 1: The YouTube Player together with the parameters used by YoMo

available information, are hence no problem for YoMo. The third task, the application comfort monitoring is more difficult to solve for reasons we explain in the following.

Recall that we defined the YouTube AC as the buffer status of the YouTube player, which is defined as the amount of playtime β, the player can continue the playback if the connection to the server is interrupted. As long as β is not zero, the video does not stall. To avoid a stalling of the video and thereby a QoE degradation, YoMo constantly monitors β and checks that it does not drop below an alarm threshold β_a. Fig. 1 depicts how β can be calculated as the difference between the currently available playtime T and the current time of the video t.

As YoMo decodes the FLV tags in real time, it exactly knows the currently downloaded playtime T which is the time stamp of the last completely downloaded tag. Intuitively, t could easily be calculated as the time difference between the actual time and the time when the player starts the video download. During our measurements we however found that this is not possible. The playback of a YouTube video does not start immediately after the player has loaded, but only after a certain amount γ of bytes have been downloaded. The analysis of experiments with 10 different videos and 5 different connection speeds shows that for each video one value of γ exists which is independent from the connection speed and the servers from which the video is downloaded. We moreover were not able to find a relation between γ and different video characteristics. This is illustrated by Fig. 2, where the coefficient of correlation between γ and the considered video characteristics which include information about the frame types of the original H264 file embedded in the FLV tags, is depicted.

As it is not possible to derive γ from the properties of the displayed video, t can not be calculated, but has to be queried from the YouTube API. This API can however only



Label	Meaning
B 0	B frames loaded till playback begin
B all	Number of B frames
P all	Number of P frames
P 0	P frames loaded till playback begin
fr	Frame rate
d	Duration
bl	Size of the file in bytes
I all	Number of I frames
vdr	Video data rate
tdr	Total data rate (= adr + vdr)
I 0	I frames loaded till playback begin
adr	Audio data rate

Figure 2: Coefficient of correlation between video characteristics and γ

be used by JavaScript or ActionScript embedded in a web site. Obviously, the original YouTube web page can not be modified, nor is it unrealistic to redirect all YouTube traffic to a dedicated web page. Hence, YoMo uses a Firefox plugin which is able to embed a JavaScript statement and a JavaBean in the YouTube site when it is loaded. The JavaScript retrieves the information from the YouTube player and the JavaBean sends the information to YoMo. YoMo uses this to compute and to visualize β in a GUI. Additionally, β can be sent to a network engineer or the network provider. The notifier can, however, also be configured to send an alert message to a network management instance if β is smaller than an alarm threshold β_a. YoMo and the Firefox plugin may be downloaded from the G-Lab website [3].

4 Application Comfort-based Quality Support in WMNs

If a YouTube video stalls, this simply means that the corresponding download lacks of bandwidth. In a wireless environment, this could be caused by a bad link quality due to fading while moving or a crowded channel. We, however, consider a WMN with stationary clients and assume that the links used by the YouTube flow are fast enough to display a YouTube video if no other traffic exists. Under these assumptions, the flow can only lack of bandwidth if the links it is using are overloaded or if the nodes forwarding the flow can not access the channel often enough as neighboring nodes are highly loaded and thereby cause too much interference. The first problem is known from the wired domain and caused by *in-band* cross traffic. The second problem is wireless specific and due to *out-band* cross traffic. We illustrate the difference between those two types of cross traffic in Fig. 3 where the wireless mesh testbed which is used for evaluating our concepts is shown. The YouTube flow from the client is routed over nodes A and B, a traffic flow using the link between A and B or B and A is hence called in-band, using the link between C and D or D and C is called out-band cross traffic.

[3] http://www.german-lab.de/go/yomo, last accessed 10/02

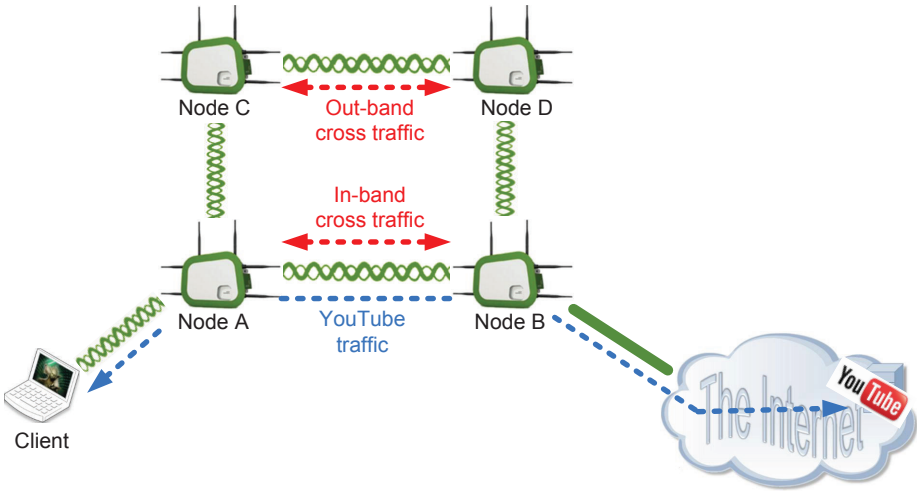


Figure 3: The mesh testbed used for the experimental evaluation

If the bandwidth of the cross traffic is reduced, the YouTube flow gets more bandwidth and the video playback continues smoothly. This is exactly the idea of a bandwidth shaper tool we describe in the following. It is inspired by previously discussed work of Pries et al. [11] who propose the interaction of traffic controller and traffic observer instances running on each of the mesh nodes. We illustrate this functionality using the setting depicted in Fig. 3. If the amount of disturbing traffic is too large, the packet loss of the YouTube flow increases. The TO instances on A and B detect this problem and notify their associated TC instance which deal with the in-band cross traffic by throttling the non-realtime flows they are forwarding. If the TO on A and B still observe packet loss, they trigger their associated TC instances to use the messaging system of the OLSR routing algorithm to deal with the out-band cross traffic. In this case, alerts are embedded in the OLSR Hello messages to inform the one-hop neighbors C and D which now in turn throttle their best effort flows.

As there is not always a direct relation between packet loss and QoE, we use the application comfort for resource management decisions. For this purpose, a monitoring software like YoMo has to run at the clients, as the application comfort can only be measured at the edge of the network. An instance of the resource management tool which we describe in the following runs at each of the mesh node. It only reacts if it is triggered by YoMo, which runs at the client, and hence called *Obedient* Traffic Controller, or OTC for short, in order to distinguish it from the TC proposed by [11].

OTC classifies all traffic to be either low priority best effort traffic like email or file transfers or high priority traffic, like YouTube, VoIP, or the OTC signaling traffic. High priority traffic is tagged by setting the TOS field to 8 which stands for “minimize delay”. The Linux tool `tc` allows OTC to setup an egress root queuing discipline (QDISC) as a priority queue with two different sub queues. The sub queue for high priority traffic, like YouTube, VoIP, or the OTC signaling traffic uses the stochastic fairness queuing

QDISC. This results in a FIFO queue for each high priority stream from which packets are dequeued in a round robin fashion. The sub queue for the low priority best effort traffic, e.g. email or file transfer traffic uses the token bucket filter QDISC which allows to upper bound the rate of the low priority traffic. In the normal case, both classes share the bandwidth available on the link, but if necessary, the maximum bandwidth of the best effort class can be limited.

OTC is running on all mesh nodes, whereas YoMo is running on the client only, therefore the communication between the instances has to be realized in a distributed fashion. To make the distributed traffic controlling approach independent from the underlying routing protocol, two types of dedicated messages are used: The *flow detection message* (DM) is sent by YoMo to announce a new high priority flow, i.e. a YouTube flow it detected. The *flow alert message* (AM) is used if YoMo detects a problem, which means that $\beta \leq \beta_a$. Both messages contain the identifier of the corresponding flow which is the unique combination of the source and destination IP and port.

The distribution of the AM and DM messages in the WMN works as follows: As soon as a mesh access point, in Fig. 3 node A, receives a DM or AM from a client, it broadcasts this message. All mesh nodes which receive an AM or a DM and which are forwarding the corresponding flow, rebroadcast the message unless they already forwarded the YoMo message. This ensures that only nodes which are forwarding the YouTube flow, or which are direct neighbors to this flow receive those messages. In our example B and C receive the broadcast from A, but only B rebroadcasts the message. The broadcast from B is received by A and D. A recognizes that it already sent the message and does not forward it.

The traffic controlling mechanism works in a similar way as the previously described concept of [11]. Upon the reception of a DM from a YoMo instance, a node forwarding the corresponding flow tags it as high priority by setting the TOS field of all its packets to 8. Furthermore, the bandwidth of the best effort traffic is reduced by a factor ΔB_d. All nodes which receive the DM from an OTC instance do not have to tag the flow, but also limit the bandwidth of the cross traffic. Nodes which do not forward the flow, ignore the DM. In contrast, both forwarding and non-forwarding nodes reduce the bandwidth of the best effort traffic by ΔB_d if they receive an AM, as this means that the YouTube video is about to stall. As long as β is smaller than β_a, YoMo sends AM messages every ΔT_d seconds. If the YouTube video has been completely downloaded or if $\beta > \beta_a$, no AM messages are sent. All mesh nodes which throttled their best effort flows and which do not receive an AM, increase the maximum bandwidth of the best effort class by an amount ΔB_i each ΔT_i sec.

5 Results

In this section the functionality of our concept is evaluated. For this purpose we first demonstrate that YoMo is suitable to estimate the stall time of a YouTube video. This analysis is contained in Section 5.1. Section 5.2 reports results from an experiment where YoMo cooperates with OTC. The value of the alarm threshold β_a has a strong

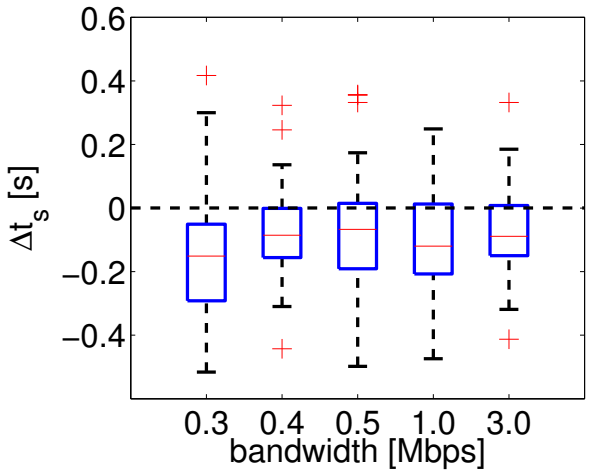


Figure 4: Difference between estimated and actual stalling time

impact on the performance of a WMN where YoMo and OTC are running. Section 5.3 therefore more deeply investigates the influence of this parameter on application and on network level.

5.1 YoMo Accuracy

In order to find out how accurately YoMo predicts the time when a video stalls, we use a client running a measurement web page which allows to dump the YouTube player state. The client is connected to the Internet via a proxy which is able to modify the connection speed and to interrupt the connection. It runs YoMo which is modified to log the buffer estimation and the corresponding timestamps. During our experiments we observed that $\beta = 0$ is a sufficient but not a necessary condition for a stalling video: many videos already stall if $\beta \approx 0.5$ sec. We therefore consider a video to stall as soon as $\beta \leq 0.5$ sec. In Fig. 4 we depict the estimation error Δt_s between the time when YoMo considers the video to stall and the video actually stalls which is the instance when the player state changes to buffering. We depict experiments with 10 different randomly chosen YouTube videos displayed at different connection speeds. For each considered bandwidth, the box depicts the inter quartile range of the estimation errors and whiskers which are 1.5 times longer than the interquartile range. Values beyond this range are shown by red crosses. This allows to see that the estimation error is independent of the bandwidth. YoMo estimates the video on average to stall roughly 0.1 sec earlier than it actually did. In most cases, YoMo underestimated the remaining play time, i.e. predicted the time of stalling earlier than it actually happened. The maximal estimation error in this direction was 0.5 sec. In some cases, YoMo overestimated the remain play time with a maximal error below 0.5 sec. Taking the inherent error of our assumption that a video already stalls if $\beta < 0.5$ into account, these results demonstrate that YoMo is working as intended.

5.2 Cooperation of YoMo and OTC

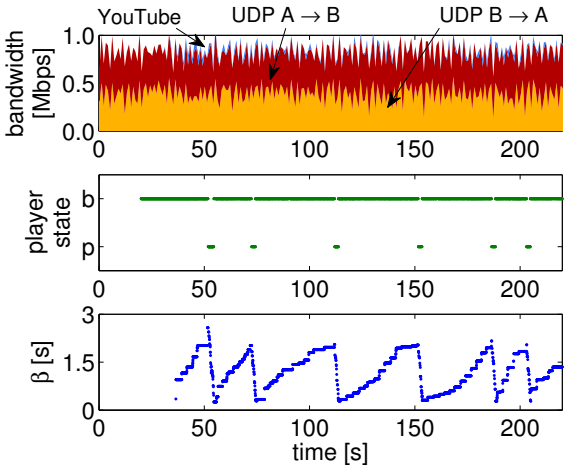
For an evaluation of the cooperation of YoMo and OTC, we use the experimental setup shown in Fig. 3. The client is a standard laptop running Microsoft Windows XP and YoMo with $\beta_a = 15$ sec. The WMN consists of four Saxnet Meshnode III which use the IEEE 802.11b spectrum in the 2.45 GHz band and run an instance of OTC which uses $\Delta T_d = 2$ sec, $\Delta B_d = 4$, $\Delta T_i = 0.6$ sec, and $\Delta B_i = 20$ kbps for the control of the cross traffic. Those values performed best during our experiments. An extensive parameter study is the scope of future works.

In our setup, the mesh gateway B is connected to the access router of the university via Ethernet. The client is connected to A, which is configured as access point. On A, two wireless interfaces are activated, one for the connection to the WMN and the other for the connection to the client. All other nodes use only one interface for building the WMN. As our testbed is located in one of our laboratories it is fully meshed, the indicated topology is created with restricting firewall rules.

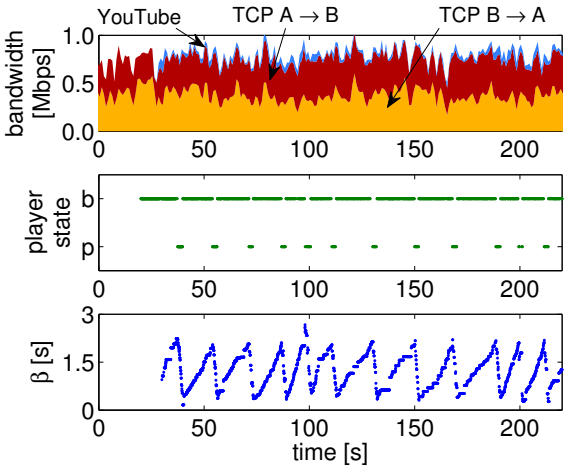
To evaluate the performance of OTC, we need a heavily loaded network where the YouTube flow gets less than the roughly 300 kbps it needs for a smooth playback. As our testbed consists only of four mesh nodes, it is difficult to overload a 54 Mbps link, especially in the out-band scenario. Therefore, we reduced the link rate to 1 Mbps, to make it easier to generate overload on the links. For our test, the client displays the YouTube video "*Madagascar I like to move music video*"[4] whereof a snapshot is shown in Fig. 1. It has a playtime of 2:49 minutes and a file size of 6.8 MB. The cross traffic is generated with the Linux tool iperf[5] as both TCP and UDP flows. The TCP flows use as much bandwidth as possible, each UDP flow generates disturbing traffic at a fixed rate of 800 kbps. We furthermore distinguish two test scenarios: an in-band scenario where the cross traffic is generated at A and B, and an out-band scenario where the cross traffic is between C and D. For each of the four combination of in-band/out-band, TCP/UDP we perform two experiments. At the beginning of each experiment, the cross traffic flows are started. Roughly ten seconds later, the client starts to display the video. During the first experiment, OTC is disabled, for the second run, it is enabled. All packets are captured with tcpdump[6] and analyzed with Wireshark [7] afterwards.

Let's start discussing the results from the experiments conducted for the in-band scenario. They are shown in Fig. 5. Fig. 5(a) and 5(b) depict the experiments without, Fig. 5(c) and 5(d) the experiments with OTC. For each of the four experiments we show the behavior of three different parameters. On the top, the bandwidth of the three flows which share the link are shown. The area at the bottom accounts for the bandwidth consumption of the flow from B to A, the area above for the bandwidth consumption of the flow from A to B, and the area on top for the bandwidth consumption of the YouTube traffic. The subplot in the middle shows the YouTube player state which is either "b" for buffering or "p" for playing. The subplot at the bottom shows the amount

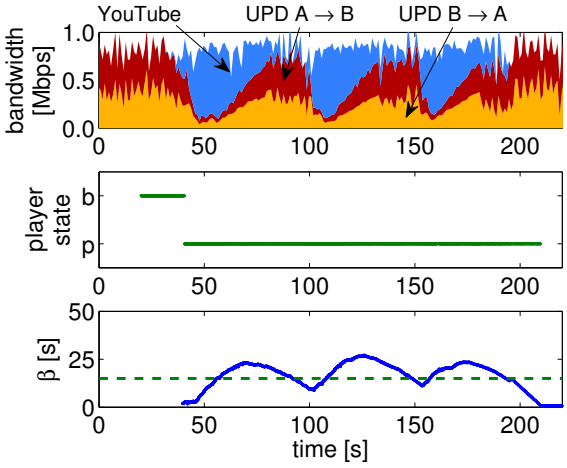
[4] http://www.youtube.com/watch?v=0x3W6hutEj8, last accessed 02/10
[5] http://sourceforge.net/projects/iperf/, last accessed 02/10
[6] http://www.tcpdump.org/, last accessed 02/10
[7] http://www.wireshark.org/, last accessed 02/10



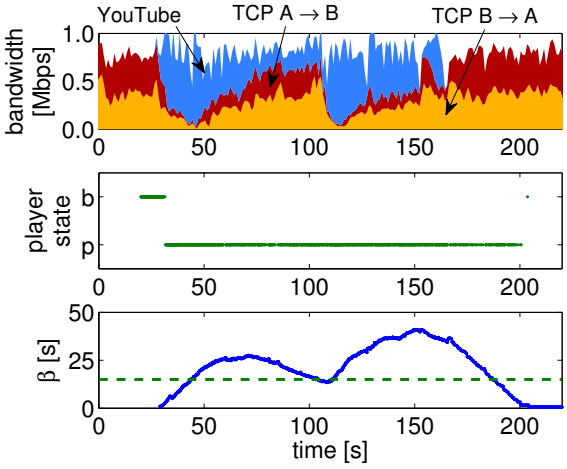
(a) UDP: In-band scenario without OTC



(b) TCP: In-band scenario without OTC



(c) UDP: In-band scenario with OTC



(d) TCP: In-band scenario with OTC

Figure 5: Video playback with in-band cross traffic

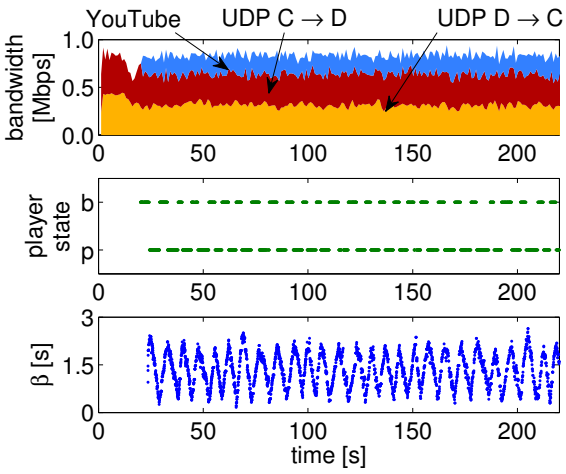
of buffered playtime.

An analysis of the experiments without OTC, represented by Fig. 5(a) and 5(b), demonstrates that the disturbing flows use about half of the bandwidth each. The bandwidth consumptions of the YouTube stream which starts after the disturbing flows are in contrast too small to be clearly visible. As a consequence, the player is spending more time in the buffering than in the playing state and during the first shown 250 sec of the measurement, the video (which has a length of 169 sec) is not completely downloaded. The subplot in the middle shows the behavior of the YouTube player. As soon as it is downloaded it tries to connect to the YouTube server to download the video. During this time, the YouTube API indicates that the player is in buffering state. As the link to the YouTube server is overloaded, the YouTube flow gets only little bandwidth and needs over 30 seconds in the TCP and more than 50 seconds in the UDP scenario before it can begin the playback. The representation of the player state visualizes moreover that the video playback is not smooth but that the player repeats the pattern of 2 seconds of playback and a roughly ten times longer buffering phase. This is simply due to the fact that the YouTube player is not able to fill its buffer in a speed faster than the video rate and also depicted by the zig-zag shape of the curves illustrating the buffer state. With UDP cross traffic the buffer is even more slowly filled than with TCP cross traffic, and the player stalls 88 times until the end of the video and 81 times in the TCP experiment.

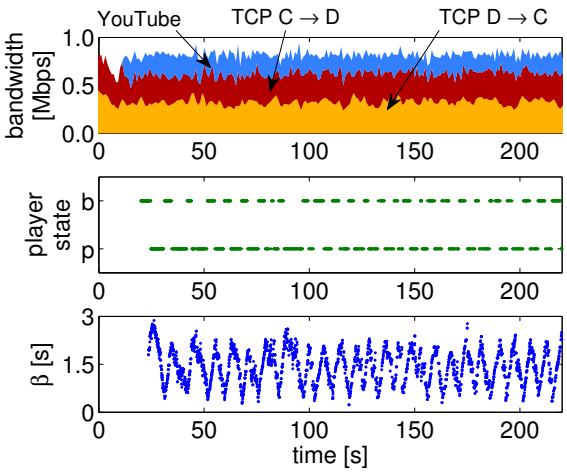
During the experiments with OTC, the video playback is terminated normally which can be recognized by the fact that the player state depicted in Fig. 5(c) and 5(d) does not change to buffering after the initial buffering phase any more. The shown results hence depict the entire measurement. However note, that the player is again 10 and 20 seconds for TCP and UDP respectively in the buffering state before it begins the video playback. This initial delay is caused by YoMo's functionality which detects the YouTube connection after the first FLV tag has been downloaded. If now the link is heavily congested, it is only after this time that YoMo can send DM and AM messages which cause the mesh nodes to throttle the cross traffic. More advanced YoMo versions will hence include mechanisms for recognizing a YouTube flow before the first FLV tag has been downloaded in order to speed up the begin of the video playback.

For both, TCP and UDP, a comparison between the consumed bandwidth and the buffer level shows that YoMo and OTC are working as intended: As soon as YoMo detects the YouTube flow it sends a DM and continues to send AM as long as β is below the threshold. Upon the reception of an AM, A and B reduce their best effort traffic. Hence more bandwidth is available to the YouTube stream and β increases. If β is large enough, YoMo does not send messages any more, and A and B begin to increase the bandwidth of the best effort traffic. Consequently, the bandwidth available for the YouTube stream and thereby β decreases. After it has become smaller than β_a, YoMo starts to send AM messages again, A and B reduce their best effort traffic and β increases. Observe that while β is behaving differently in the TCP and UDP case, the player does not stall in both experiments, YoMo and OTC are hence able to guarantee a perfect QoE.

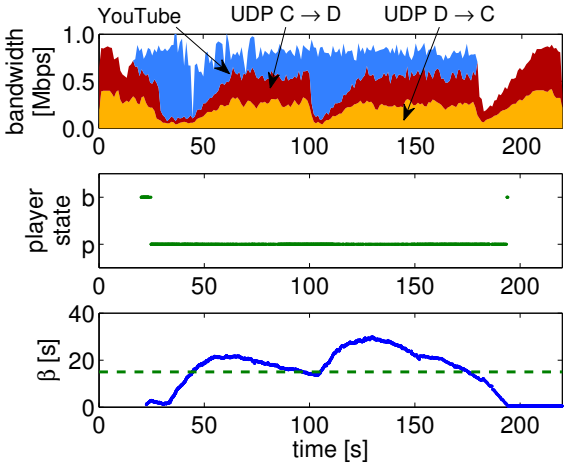
Fig. 6 shows the results from the experiments in the out-band scenario. This time the disturbing traffic is not generated between A and B but on the interfering link C-D.



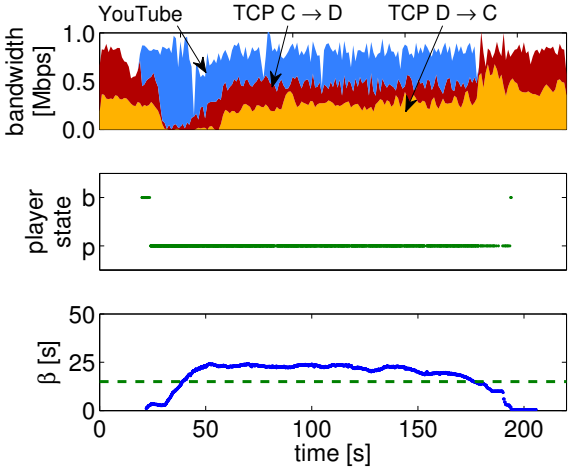
(a) UDP: Out-band scenario without OTC



(b) TCP: Out-band scenario without OTC



(c) UDP: Out-band scenario with OTC



(d) TCP: Out-band scenario with OTC

Figure 6: Video playback with out-band cross traffic

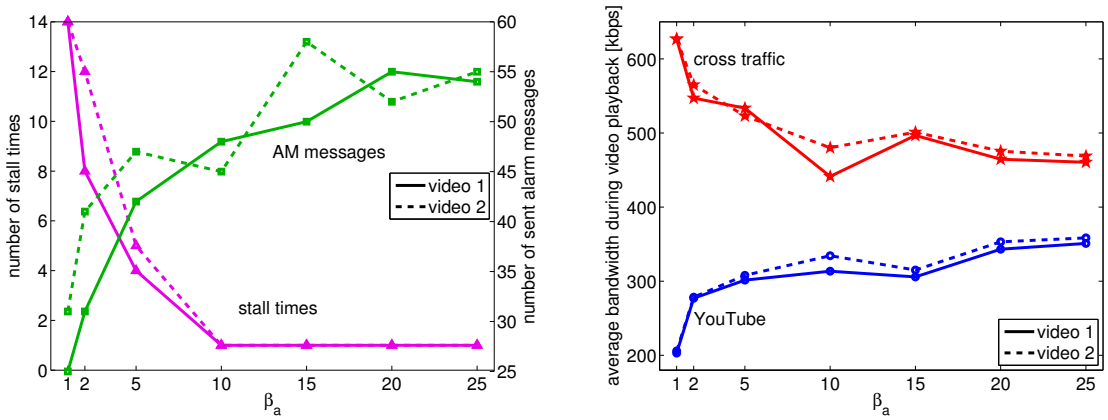
In the experiments without OTC shown by Fig. 6(a) and 6(b), the YouTube flow hence gets a bigger share of the bandwidth than in the in-band case as three nodes compete for the transmission opportunity and as a result the bandwidth is more equally distributed. The representation of the player state and of the buffering level however visualize that this is also not enough to guarantee a smooth playback. As in the experiment before, the shape of the curve depicting the buffer level shows a zig-zag behavior illustrating a sequence of playback and buffering phases. Even if the playing and buffering phases are now alternating more quickly, the download has not completed until the 230 seconds of the measurement we show. Moreover does the video stall 41 times regardless the type of the cross traffic.

In the experiment with OTC shown in Fig. 6(c) and 6(d), we may again observe that the concept is working as intended. After an initial buffering phase, which is longer than in non-congested networks but shorter than in the scenario with in-band traffic, YoMo detects the YouTube flow and sends a DM and AMs until β has reached the alarm threshold. As a result C and D throttle their traffic, the video playback may hence begin earlier than without OTC. We may again observe that as soon as no AM messages are sent any more the UDP cross traffic causes the YouTube buffer to decrease and the controlling algorithms has to start again. This is however not the case for TCP cross traffic. In this situation the disturbing flows keep their bandwidth consumption at a level which allows a fair share of the bandwidth.

In both cases the video did not stall during the experiment, YoMo and OTC were again able to guarantee the user QoE. The representation of the cross traffic bandwidth in Fig. 6(c) visualizes however another inherent problem of YoMo: Shortly before the video playback is completed, the playtime buffer reaches the alarm threshold, AMs are sent, and hence C and D limit the best effort traffic. In this case this is however not necessary, as the video is nearly completely loaded ant eh YouTube flow does not need much bandwidth any more. Future versions of YoMo will hence incorporate the knowledge about the playback end in the decision about sending alarm messages or not.

5.3 Parameter study for β_a

The previous section demonstrated that YoMo and OTC are cooperating successfully. This section will more closely analyze the advantages and disadvantages of different parameterizations of β_a, a factor which has a major influence on the performance of the controlling algorithm. For this purpose we use the same setup as before but generate bidirectional TCP cross traffic between A and B and bidirectional UPD cross traffic between C and D. After the start of the cross traffic, the client starts to display the video. YoMo is always activated with different values for the buffer alarm threshold. Due to the heavily congested network, the YouTube flow gets at most 1 kbps before YoMo can send the first message. In Fig. 5 and 6 it can be observed that the chosen video shows a particular buffering behavior: β increases until it is roughly larger than 2 seconds. The playback begins and the β decreases nearly to 0 before it starts increasing again. In medium congested networks with YoMo enabled this is no problem. In this experiment, it causes however that for all parameterizations of β_a the video stalls after



(a) Stallings and alarm messages

(b) Bandwidth consumptions

Figure 7: Influence of the alarm buffer threshold β_a

the first 3 seconds of playback, as the playtime buffer is not replenished fast enough. For this reason, we repeated the experiment with a second video, "Al Hirt Rocky"[8] with a length of 3:28 min and a size of 8.6 MB. While this video, called *video 2* in the following, is both longer and bigger than the previously considered *vide0 1*, the behavior of the buffer status shows the same trend. Moreover are all results we show in the following comparable, although video 2 is larger and longer than video 1.

This phenomenon is depicted by Fig. 7(a) which shows that both videos are stalling one time if $\beta_a \geq 10$ sec. If the alarm threshold is set to a smaller value, the video stalls significantly more often as the reaction of OTC upon the reception of an AM, simply does not allow the YouTube player to replenish its buffer fast enough. The drawback of a large alarm threshold is also depicted in Fig. 7(a): the larger β_a, the more overhead in terms of AMs is caused. At the moment OTC is configured to send and AM each two seconds as long as β is below the threshold. If this threshold is small, e.g. if $\beta_a = 1$ sec, than the buffer runs empty before the reduction of the cross traffic flows allows the YouTube player to grab more bandwidth. As the video is not played any more, the buffer is filled quickly and at most after 3 sec again larger than the alarm threshold. Consequently each time the buffer is below the threshold at most 2 messages are sent. For the case of a larger threshold, e.g. $\beta_a = 20$ sec, the video playback could continue for 20 seconds from the time when β drops below the threshold. The playback hence continues and more amount of playtime has to be downloaded until β is again larger than β_a in comparison to the setting with $\beta_a = 1$ sec. On average, each time the threshold is too small, 5 AM messages are sent. We saw that the buffer threshold is more often below the alarm threshold if this is set to a small value, but less AMs are sent than if β_a is set to a larger value, as those phases take less time. If a large value for β_a is chosen, YoMo additionally sends a larger number of AM before the playtime buffer is the first time larger than the threshold.

[8] http://www.youtube.com/watch?v=X60jEHxEAY4, last accessed 02/10

The average bandwidth of YouTube and of the cross traffic during the video playback are shown in Fig. 7(b). Before YoMo and OTC start to control the bandwidth in the network, the YouTube flow gets roughly 1 kbps, as the other traffic flows predominate. The fact that small values for $\beta_a < 10$ sec are not advantageous which we already observed in Fig. 7(a) is also visible here: the minimal bandwidth for a simple YouTube video to play without stalling is roughly 300 kbps which is only reached if $\beta_a \geq 10$ sec. The price for this is clearly a significant restriction of the best effort traffic. A configuration with a value in the range of $\beta_a = 5$ sec in contrast, is insofar also interesting as it allows to find a balance between application comfort, overhead and reduction of best effort traffic. Values in this range guarantee a video playback with few more stall times while it reduces the cross traffic less strongly and causes less AM to be sent.

Clearly, any optimization of β_a has to be done in respect to the network situation. More exhaustive parameter studies involving the parameters of OTC under different load conditions are hence the subject of our future works. This results presented in this section are however able to illustrate the configuration possibilities for our scheme.

6 Conclusion and Outlook

Application comfort monitoring presents an approach to monitor the usage of applications, their quality requirements and the experienced application comfort at the client. Application comfort goes beyond quality of experience in so far that it also considers the future development of the QoE if a prediction is possible. In the scope of our example application, Flash video streaming over TCP connections, the application comfort is measured as the buffered playtime. YoMo, the ACM tool for Flash videos or concretely for YouTube streaming, consists of two parts: a browser plug-in that monitors the state, in particular the current playtime, of the Flash player and a packet sniffer that detects new Flash video transfers, extracts the videos metadata, and monitors the available playtime. Both components together allow to determine exactly the buffered playtime, even if the user jumps within the video.

As discussed in the introduction and the related work section, ACM can be a solution to provide some kind of QoS support to many Internet applications that by themselves do not communicate their quality requirements. Possible use cases for YoMo are within the scope of the connection setup and adaptive parameterization in future wireless networks like LTE or WiMAX. Additionally, application comfort monitoring has the potential to deliver essential context information on the application usage for resource management frameworks like IEEE 802.21 or IEEE 1900.4.

As a scenario for our proof of concept we used a wireless mesh network where YoMo is running at the client. It autonomously detects YouTube videos and signals that presence to the OTC instances running at the mesh nodes. This enables OTC to classify the YouTube flow as high priority and to restrict the bandwidth of low priority traffic if necessary. To ensure a good quality in mesh networks this is however not sufficient. Much traffic with low priority on interfering nodes may nevertheless decrease the bandwidth of the YouTube flow and hence lead to stall times. In order to prevent this, YoMo

continuously monitors the buffered playtime. If this decreases below a threshold YoMo sends alarm messages to the OTC instances which cause both the forwarding nodes and the one-hop neighbors to limit their best effort traffic. In our testbed we set up two test cases. In the in-band case, the cross traffic is generated on a link used by the YouTube flow, in the out-band case, the cross traffic is on an interfering link. In both cases the cooperation of YoMo and OTC is able to avoid stall times.

YoMo demonstrates the appeal of ACM for both costumers and network providers. YoMo is lightweight and easy to install while it provides valuable information to a network manager. If users run YoMo and the network provider a tool similar to the discussed OTC, both parties may greatly benefit as the provider gets information for free it can use for improving the user QoE. Our future work will therefore be dedicated to develop this concept in two directions. Firstly, we will refine YoMo and develop similar tools for other popular applications like VoIP, scalable video or web traffic. Secondly, we will work on an improved radio resource management scheme for wireless mesh networks which include intelligent access mechanisms, gateway selection, channel selection, and routing.

Acknowlegements

The authors would like to thank Phuoc Tran-Gia and Robert Henjes for their support and valuable comments. More thanks go to Sebastian Deschner for his help during the experiments.

References

[1] Cisco Systems Inc. Cisco Visual Networking Index - Forecast and Methodology, 2008-2013. White Paper, 2009.

[2] Schulze H, Mochalski K. Internet Study 2008/2009. http://www.ipoque.com/resources/internet-studies/ 2009.

[3] Wamser F, Pries R, Staehle D, Heck K, Tran-Gia P. On traffic characteristics of a broadband wireless internet access. *Special Issue of the Telecommunication Systems (TS) Journal*, 2010; .

[4] TR-059: DSL Evolution - Architecture Requirements for the Support of QoS-Enabled IP Services. *Technical Report*, DSL Forum 2003.

[5] Ludwig R, Ekstrom H, Willars P, Lundin N. An evolved 3gpp qos concept. *VTC'06-Spring*, Melbourne, Australia, 2006.

[6] Traffic inspection for visibility, control and new business opportunities. Ericsson White Paper, 2008.

[7] Hannes Ekstrm. QoS Control in the 3GPP Evolved Packet System. *IEEE Communications Magazine*, 2009; **2**.

[8] Maier G, Feldmann A, Paxson V, Allman M. On dominant characteristics of residential broadband internet traffic. *IMC '09*, Chicago, IL, USA, 2009.

[9] 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.

[10] Tsagkaropoulos M, Politis I, Dagiuklas T, Kotsopoulos S. Enhanced vertical handover based on 802.21 framework for real-time video streaming. *MobiMedia'09*, London, UK, 2009.

[11] Pries R, Hock D, Bayer N, Siebert M, Staehle D, Rakocevic V, Xu B, Tran-Gia P. Dynamic bandwidth control in wireless mesh networks: A quality of experience based approach. *18th ITC Specialist Seminar on Quality of Experience*, Karlskrona, Sweden, 2008.

[12] Van Moorsel A. Metrics for the internet age: Quality of experience and quality of business. *PMCCS 5*, Erlangen, Germany, 2001.

[13] 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.

[14] 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).

[15] Hock D, Bayer N, Pries R, Siebert M, Staehle D, Rakocevic V, Xu B. Qos provisioning in WLAN mesh networks using dynamic bandwidth control. *European Wireless 2008*, Prague, Czech Republic, 2008.

[16] Raake A. Short-and long-term packet loss behavior: towards speech quality prediction for arbitrary loss distributions. *IEEE Transactions on Audio, Speech, and Language Processing* 2006; **14**(6).

[17] Guéguin M, Le Bouquin-Jeannès R, Gautier-Turbin V, Faucon G, Barriac V. On the evaluation of the conversational speech quality in telecommunications. *EURASIP Journal on Advances in Signal Processing* 2008.

[18] Hoßfeld T, Binzenhöfer A. Analysis of Skype VoIP traffic in UMTS: End-to-end QoS and QoE measurements. *Computer Networks* 2008; **52**(3).

[19] Mohamed S, Rubino G, Varela M. Performance evaluation of real-time speech through a packet network: a random neural networks-based approach. *Performance Evaluation* 2004; **57**(2).

[20] Engelke U, Zepernick HJ. Perceptual-based quality metrics for image and video services: A survey. *Proc. rd EuroNGI Conference on Next Generation Internet Networks*, 2007.

[21] Cisco Systems Inc. Approaching the Zettabyte Era. White Paper, 2008.

[22] Akyildiz I, Wang X, Wang W. Wireless mesh networks: a survey. *Computer Networks* 2005; **47**(4).

[23] Architectural Building Blocks Enabling Network-Device Distributed Decision Making for Optimized Radio Resource Usage in Heterogeneous Wireless Access Networks, 2009.

[24] Guenin J. Overview of SCC41 and the 1900.x Working Groups. IEICE Software and Cognitive Radio Expo and Technical Conference, 2008.

[25] Buljore S, Harada H, Filin S, Houze P, Tsagkaris K, Holland O, Nolte K, Farnham T, Ivanov V. Architecture and enablers for optimized radio resource usage in heterogeneous wireless access networks: the IEEE 1900.4 working group. *IEEE Communications Magazine* 2009; **47**(1).

[26] Lampropoulos G, Salkintzis A, Passas N. Media-independent handover for seamless service provision in heterogeneous networks. *IEEE Communications Magazine* 2008; **46**(1).

[27] Ong E, Khan J. Cooperative radio resource management framework for future IP-based multiple radio access technologies environment. *Computer Networks* 2009; **In Press, Corrected Proof**.

[28] Kassar M, Kervella B, Pujolle G. An Intelligent Handover Management System for Future Generation Wireless Networks. *EURASIP Journal onWireless Communications and Networking* 2008.

[29] Bullot, T and Gaïti, D and Pujolle, G and Zimmermann, H. A piloting plane for controlling wireless devices. *Telecommunication Systems* 2008; **39**(3).

[30] Hoßfeld T, Tran-Gia P, Fiedler M. Quantification of quality of experience for edge-based applications. *ITC20*, Ottawa, Canada, 2007.

[31] Piamrat K, Ksentini A, Viho C, Bonnin J. QoE-Aware Admission Control for Multimedia Applications in IEEE 802.11 Wireless Networks. *VTC 2008-Fall*, Calgary, Canada, 2008.

[32] Bohnert TM, Staehle D, Kuo GS, Koucheryavy Y, Monteiro E. Speech quality aware admission control for fixed IEEE 802.16 wireless MAN. *ICC'08*, Beijing, China, 2008.

[33] Bohnert TM, Staehle D, Monteiro E. Speech quality aware resource control for fixed and mobile wiMAX. *WiMAX Evolution*, Marcos Katz FF (ed.). John Wiley & Sons, 2009; 227.

[34] Staehle B, Binzenhöfer A, Schlosser D, , Boder B. Quantifying the influence of network conditions on the service quality experienced by a thin client user. *MMB'08)*, Dortmund, Germany, 2008.

[35] Michaut F, Lepage F. Application-oriented network metrology: Metrics and active measurement tools. *IEEE Communications Surveys & Tutorials* 2nd Quarter 2005; **7**(2).