Aquarema in Action: Improving the YouTube QoE in Wireless Mesh Networks

Barbara Staehle, Matthias Hirth, Rastin Pries, Florian Wamser, Dirk Staehle
University of Würzburg, Institute of Computer Science, Chair of Communication Networks, Würzburg, Germany
{bstaehle, matthias.hirth, pries, florian.wamser, dstaehle}@informatik.uni-wuerzburg.de

***Abstract*—Internet access networks in general and wireless mesh access networks in particular, are the bottleneck of today's communication networks and consequently most strongly responsible for determining the user satisfaction. The limited bandwidth and the fact that access network are most often used as mere bit pipes are however unfavorable for the users' quality of experience (QoE). The lack of application-specific service guarantees is especially inadequate in the face of an increasing degree of heterogeneity of Internet applications and their individual service requirements. Application and quality of experience resource management, *Aquarema* for short, addresses this challenge by enabling application specific network resource management and thereby improves the user QoE. This is achieved by the interaction of application comfort (AC) monitoring tools, running at the clients, and a network advisor which may trigger different resource management tools. AC quantifies how well an application is running and in particular, enables a prediction of the user experience, thereby allowing the network advisor to act upon an imminent QoE degradation. We demonstrate the appeal of Aquarema at the example of YouTube video streaming in a congested IEEE 802.11 based mesh testbed where AC-aware traffic shaping guarantees the successful video playback.**

I. Introduction

In today's consumer Internet, most traffic is transmitted on a best effort basis without supporting application-specific quality requirements. Neither in the backbone nor in the wireless access networks,the focus of our research interest, exist service guarantees for the predominant consumer Internet traffic consisting of applications as different as P2P or client-server file sharing, web browsing, gaming or video streaming [1], [2]. Technical solutions enforcing quality guarantees exist, see e.g. [3], but in general the network does neither know if it is carrying Internet applications requiring a service guarantee nor which quality requirements to meet.

These prerequisites for providing quality of service (QoS) support for Internet applications require a cross-layer approach which allows an information exchange between application and network. Detecting packets belonging to a specific application in the packet stream is most often done using deep packet inspection [4]. Deep packet inspection has however become rather challenging as the browser tends to be the user's interface to the Internet for an increasing number of applications like videos, large file downloads or browser games, Obviously, detecting packets would not be necessary if applications simply communicated their presence to the network. The downside of this method is however that it requires a modification of the application.

Detecting packets is only halve the coin as the network has not only to be aware about the presence of an application, it also has to know about appropriate quality parameters. Deriving such application-specific parameters from an observed traffic flow is even more complex than detecting it, in particular for reactive TCP traffic. Letting the client communicate these parameters to the network would again be a solution, but for many applications, like videos with variable bit rate, not all QoS requirements are a priori known.

Giving strict QoS guarantees is moreover neither possible in wireless networks in general nor in the IEEE 802.11 wireless mesh networks (WMNs) we concentrate on, as a link between two nodes typically does not have a constant capacity but has to share its access time with surrounding links. Quality of experience (QoE) based resource management, see e.g. [5], [6] therefore continuously adapts the network resources to quality feedback from the application and is better suited for WMNs.

Resource management in WMNs, see e.g. [7], covers routing including gateway selection, channel and interface allocation in multi-radio multi-channel mesh networks, prioritization of medium access through contention parameters, and finally traffic shaping. Aquarema, realizing an application and QoE resource management, therefore foresees the interaction of a *network advisor* which decides upon the use of different *resource management tools* in case the application QoE is imminent. The structure of such an universal architecture is rather complex, as reputation systems and policies regulating the situation if several applications are active have to be included. We are however convinced that despite these challenges radio resource management when the user experience can be derived from application layer information offers an enormous potential for WMNs.

QoE is a measure for the subjective quality that a user experiences [8]. In contrast to QoS, the QoE depends not only on the network but also on the situation and user specific factors and is therefore less easy to derive. Moreover do the large number of QoE models which exist e.g. for VoIP traffic [9], [10] or video streaming [11], [12], [13] only allow to quantify the user satisfaction *after* the application has been terminated. Our goal is however to use the QoE as an input for a network management tool. Therefore, we need to know the user satisfaction *during* the execution of the application. To avoid a QoE degradation, a network management tool

978-1-4244-8513-0/11/$26.00

has moreover to be notified if a QoE degradation has not yet happened, but is only imminent. This is achieved by monitoring the *application comfort* (AC) which characterizes how well the application is doing and allows an prediction of a QoE degradation. A vital part of Aquarema is therefore the *AC monitor*, a generic tool installed at the client which signals the presence of an application to the network advisor and constantly monitors the AC. If it falls below an alarm threshold, this is immediately reported to the network advisor which triggers a suitable resource management tool.

In this work we illustrate the potential of Aquarema using a simple setup with one supported application and one resource management tool. For this setup the presence of a network advisor is not necessary, as the AC tool may directly trigger the resource management tool. YouTube is chosen as a representative example for an Internet application, as YouTube and Flash video (FLV) streaming portals in general, make up for roughly 10% of the traffic volume of a private households which use a WMN as Internet access network [2] and will exceed 91% of the overall Internet traffic in 2014 [1]. The resource management tool is realized by a bandwidth shaper.

The rest of this paper is structured as follows: In Section II we give an overview of the publications related to our work. The YouTube AC monitoring tool YoMo and the bandwidth shaping tool OTC are introduced in Section III and Section IV respectively. Their cooperation is evaluated in Section V. In Section VI we summarize the contributions of this paper and give an outlook on future work.

II. Related Work

Works related to the ideas beneath Aquarema cover recently published radio resource management techniques and frameworks. Of particular interest are approaches deriving the QoE from network or application layer parameters and using them for resource management.

The IEEE is currently developing standards towards an improved usage of radio resources in heterogeneous wireless access network. A management system allowing the distributed optimization of radio resource usage and improving the QoS in heterogeneous wireless networks is defined by IEEE 1900.4 [14]. The upcoming IEEE 802.21 has the goal of specifying standards for media independent handover in heterogeneous radio access networks [15]. Both approaches have in common that they aim at optimizing resource usage in heterogeneous wireless networks, establish a signaling framework between terminals and network, and try to enable context-aware resource management decisions.

A similar idea is presented in [16] where Ong and Khang introduce a cooperative radio resource management framework for enabling seamless multimedia service delivery. The framework enables a terminal to take a handover decision based on the QoS information broadcasted by the 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. A comparable but more general framework has been presented by Bullot et al. [17] who propose an architecture for decentralized network management and control. Their main contribution is the idea of a piloting system which uses a control plane to measure different system parameters and provides them to control algorithms like routing or mobility management. This reduces the measurement and storage overhead, as system information can be used by several algorithms. In a simulation study of a mobile-initiated handover, this framework taking into account the load of the APs, results in less rejected VoIP calls and a lower end-to-end delay than the traditional policy of considering the received signal strength only.

For WMNs many other possibilities for resource management than just handover exist. Akyildiz et al. [7] give an overview on the existing alternatives, we therefore refrain from an exhaustive enumeration and introduce the work of Pries et al. [6] which has inspired this work, as an example. 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 two tools, the traffic observer (TO) and the traffic controller (TC) running at each mesh node. The TO continuously monitors the QoE of the VoIP flows, which is calculated as Mean Opinion Score (MOS) from the measured packet loss. As soon as the MOS falls below a threshold, signaling messages are sent via the OLSR Hello message system to assure that the TC instances running on the same and on the neighboring nodes throttle the interfering best effort traffic. The evaluation of this concept in a mesh testbed shows that TO and TC together allow to maintain a satisfying MOS score even in the presence of disturbing traffic.

The exponential relation between packet loss and MOS used in [6] has been discovered by Hoßfeld et al. [18] and is one example of mapping measurable QoS parameters to user experience. Many contributions similar to this work exist, see e.g. [9], [10], [11], which are increasingly used for resource management. One example is the work of Bohnert et al. [19] who modify the QoE model for speech quality proposed by Raake [9] in order to obtain a QoE estimation for an aggregate of VoIP calls. The resulting QoE based admission control scheme successfully avoids long periods of user dissatisfaction in an IEEE 802.16 access network. Another example is the work of Khan et al. [20] who use the peak signal to noise ratio for deriving the average QoE for video streams. The authors present a resource management scheme that optimizes video source rate, time slot allocation and modulation scheme in order to maximize the average video stream QoE.

For the case of YouTube, this and other QoE models for video streaming (cf. to [11] for a survey) are not applicable, as they assume UDP as transport layer protocol. UDP results in a strong impact of delay, jitter, and packet loss on the video QoE as they cause artifacts or missing frames. YouTube videos, in contrast, are transported via HTTP over TCP. Consequently, YouTube has not to cope with lost or reordered packets, and the only quality degradation which may be caused by the

transmission, is a stalling of the video. The only approach applicable for deriving the YouTube QoE we are aware of is the work of Gustafsson et al. [12] who derive the YouTube QoE from video parameters, the packet loss rate, and the number of buffering events during the playback.

If however the QoE shall be used for network management, the user satisfaction has to be known in real time. Moreover, a management tool needs a prediction of the QoE, i.e. it needs to be notified if the YouTube player is about to stall in order to avoid this event. Dalal et al. [21] propose a QoE prediction mechanism for UDP video streaming. It however uses lost and retransmitted packets only, a method which does not work in case of YouTube. Deriving QoE from QoS parameters is always problematic, as a mapping is highly application and technology dependent. We therefore propose to predict the QoE from *application* and not *network* parameters, for allowing a reactive adaption of network parameters.

III. Monitoring the YouTube AC

The QoE of a YouTube user depends on factors as different as the video content, the video- and audio-quality, or the time until the playback starts. Determining the YouTube QoE is hence a difficult task and out of scope of this work. What in any case negatively affects the user satisfaction is a *stalling* of the video, i.e. an interruption of the video playback. This is our motivation for defining the YouTube AC as "bad" if the video playback will be interrupted soon and "good" otherwise.

The YouTube AC monitoring tool YoMo has to fulfill several tasks: (1) it has to detect that a YouTube flow exists and to communicate this information to the network advisor. (2) it has to collect as much information as possible about the YouTube flow. And (3), it has to monitor the YouTube AC and raise an alarm if the AC becomes bad. To make the concept of YoMo more easy to understand, we first of all analyze the technology behind YouTube in III-A, before we introduce the main ideas of YoMo and their implementation in III-B and III-C.

A. The Technology Behind YouTube

The YouTube player is a proprietary Flash application which concurrently plays a Flash video (FLV) file and downloads it via HTTP. Each YouTube video is encoded as an FLV file which is a container format for media files developed by Adobe Systems. An FLV file encapsulates synchronized audio and video streams and has a dedicated header starting with an FLV signature. The tags encapsulate the data from the video streams and contain information on their payload including the payload type, the length of the payload, and the time to which the tag payload applies. FLV files may also contain metadata which contain information about the duration of the video, the audio and video rate, and the file size.

The concept of concurrent video download and playback used by YouTube is called pseudostreaming. This technique follows a dual-threshold buffering strategy which means that at the beginning the download, the client fills an internal buffer and starts the video playback as soon as a minimum buffer level, γ, is reached. During the time of simultaneous playback and downloading, the buffer grows until a certain larger threshold is reached and as long as the download bandwidth is larger than the video rate. Otherwise it shrinks. If the buffer runs empty, the video stalls and the YouTube player state changes from "playing" to "buffering". This state is hidden to the normal user, but can be retrieved from the YouTube API by JavaScript or ActionScript.

Fig. 1. The YoMo Parameters

B. The Main YoMo Functionality

The YouTube player opens a new TCP connection each time it downloads a new FLV file or if the user jumps to another time in the video. YoMo runs at the client and parses all incoming TCP flows. Consequently it recognizes each flow containing FLV data as soon as the header of an FLV file is found. Once a flow containing FLV data is recognized, its data is continuously parsed in order to retrieve the available meta information from the FLV file. Detecting the YouTube flow is thus easily done. The AC monitoring task is more complex and will be explained in the following.

The YouTube AC is defined as the buffer status of the YouTube player. This is simply the time, β, the player can continue playing if the connection to the server is interrupted. Fig. 1 shows β for the YouTube video "*Madagascar I like to move music video*"[1] and how it can be calculated as the difference between the amount of already downloaded play-time T and the current time of the video t. YoMo constantly computes and visualizes β in a GUI and checks that it does not drop below an alarm threshold β_a. In such a situation, like the one depicted in Fig. 1, the video is still playing, but $\beta < \beta_a$ which means that the AC is bad as the video is about to stall. YoMo predicts this upcoming stalling and notifies the network advisor which now is able to take measures for preventing the stalling and the QoE degradation.

[1] http://www.youtube.com/watch?v=0x3W6hutEj8, last accessed 07/10, removed due to YouTube terms of service violations

C. Estimating the Buffered Playtime

YoMo decodes the FLV tags in real time, and is hence able to derive the currently available playtime T from 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 to play the video. During our measurements we found that the dual-threshold buffering strategy makes this task non-trivial. Recall that the playback of a YouTube video does not start immediately after the player has loaded, but only after an amount γ of bytes has been downloaded. An experiment with different videos and connection speeds revealed that γ is independent of the connection speed, but is varying between 50 and 300 kB for different videos. Analyzing the coefficient of correlation between γ and different video characteristics including information about the frame types of the original H264 file embedded in the FLV tags, did not show a clear correlation which allows to derive γ from the properties of the displayed video (cf. to [22] for details).

It is hence not possible to calculate the amount of time elapsing between the user request and the time when the video actually starts to play. The assumption that the video playback starts as soon as the first FLV tag is completely downloaded introduces an error in the calculation of β. In [23] we showed that in the case of a sudden connection interruption this method estimates the video to stall time on average 2 seconds earlier than it actually stalls. The estimation error decreases with an increasing connection speed, but we decided to improve YoMo's estimation accuracy by obtaining t from the YouTube player API. Recall that the YouTube API can be accessed by scripting languages only. In order to make YoMo applicable for productive environments, it has to work with the original YouTube web page where no scripts querying the player API are running. It is also unrealistic to redirect all YouTube traffic to a dedicated web page where scripts for YoMo are running. Hence, YoMo uses a Firefox plugin which runs a JavaScript that retrieves t from the YouTube player. The plugin additionally embeds a Java applet in the YouTube site which sends the actual value of t to YoMo.

The cooperation with the Firefox plugin allows YoMo to estimate the video on average to stall roughly 0.1 sec earlier than it actually did [22], [23]. In most cases, YoMo underestimates the remaining play time, i.e. predicts the time of stalling earlier than it actually happened. The maximal estimation error in this direction is 0.5 sec which is accurate enough for radio resource management. YoMo and the Firefox plugin may be downloaded from the G-Lab website[2].

IV. WMN Resource Management for YouTube

If a YouTube video stalls, this means that the corresponding download has not enough bandwidth. In a wireless environment, this could be caused by a bad link quality, fading, 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 the video. One reason for a lack of bandwidth could hence be that links used by the YouTube flow are overloaded. The other reason could be the one that 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. 2 which shows the wireless mesh testbed used for evaluating Aquarema.

In both situations the YouTube flow gets more bandwidth and the video playback continues smoothly if the bandwidth of the cross traffic is reduced. This is exactly the idea of the bandwidth shaper tool OTC we describe in the following. OTC is short for *obedient traffic controller* as it is inspired by the previously discussed work of Pries et al. [6] who propose TC traffic controller instances which cooperate with TO instances running on the same mesh node. OTC in contrast, reacts only if it receives messages sent by YoMo and is hence called *obedient* traffic controller. Realizing the communication between AC monitoring tools like YoMo, the network advisor and OTC in a distributed fashion is necessary as application information can only be gathered at the edge of the network. The advantage of this solution is that measuring the AC allows to predict an imminent QoE degradation and works also for cases where the dependency between network parameters and QoE is not as direct than in cases like the one studied by [6].

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 signaling traffic. The Linux tool tc allows OTC to set up an egress root queuing discipline as a priority queue with two different sub queues. The sub queue for high priority traffic uses the stochastic fairness queuing discipline. 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 uses the token bucket filter queuing discipline which allows to upper bound the rate. 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.

In contrast to the approach proposed by [6] which uses the OLSR messaging system, the communication between YoMo and the OTC instances is independent from the underlying routing protocol. It is assured by two types of dedicated messages: The *flow detection message* (DM) is sent as soon as YoMo detects a new YouTube flow. If the AC becomes bad, i.e. if β falls below β_a, YoMo sends a *flow alert message* (AM). Both message types contain the unique combination of source and destination IP and port of the flow they refer to, and are tagged to be high priority in order to ensure the packet delivery despite a congested network. If a mesh access point receives a DM or AM from a YoMo instance running at the client, it broadcasts this message. All mesh nodes which receive a DM or an AM and which are forwarding the corresponding flow, rebroadcast this message one time. This ensures that only

[2]http://www.german-lab.de/go/yomo, last accessed 01/11

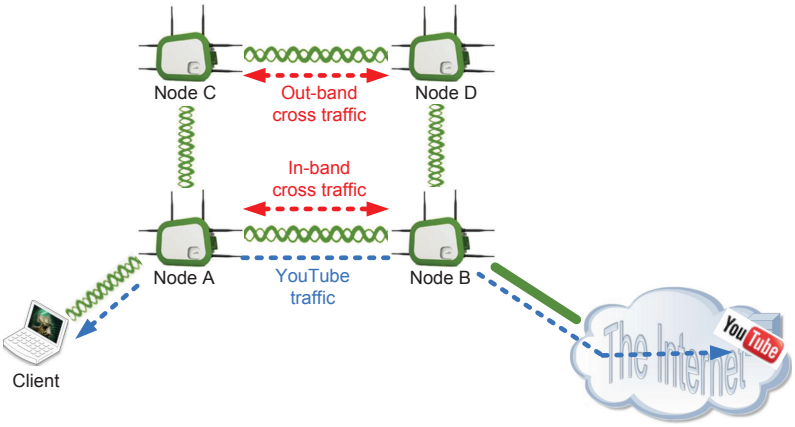


Fig. 2. The mesh testbed used for the experimental evaluation

nodes forwarding the YouTube flow, or direct neighbors to this flow receive the DM and AM messages.

Upon the reception of a DM, a node forwarding the corresponding flow tags it as high priority. Additionally, the in-band cross traffic is throttled by reducing the maximal available bandwidth for the best effort traffic is reduced by a factor ΔB_d. To handle the out-band cross traffic, both forwarding and non-forwarding nodes reduce the bandwidth of the best effort traffic by ΔB_d if they receive an AM. As long as β is smaller than β_a, YoMo sends AM messages every ΔT_d sec. If the YouTube video has been completely downloaded or if $\beta > \beta_a$, no AM messages are sent any more. 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. This ensures that cross traffic not limiting the capacity of the YouTube flow is not unnecessarily constrained.

V. Experimental Evaluation

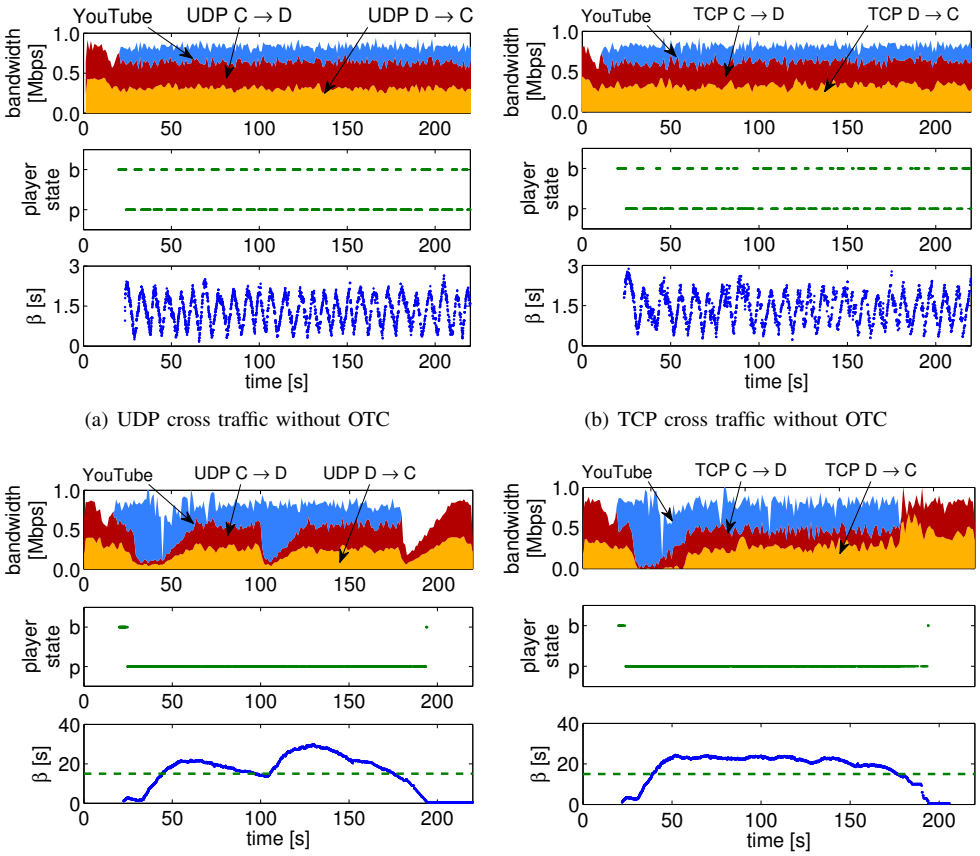
This section evaluates the functionality of Aquarema in the form of the cooperation between YoMo and OTC. Section V-A examines the improvements of the user QoE resulting from this cooperation. The effects of different parameterizations for the alarm threshold β_a are more deeply investigated in Section V-B.

A. Cooperation of YoMo and OTC

The experimental setup used for the evaluation is shown in Fig. 2. The client is a standard laptop running Microsoft Windows Vista, the Mozilla Firefox browser and YoMo. The WMN consists of four Saxnet Meshnode III which use one channel of the IEEE 802.11b spectrum in the 2.45 GHz band. All mesh nodes run OTC which is configured with $\Delta T_d = 2$ sec, $\Delta B_d = 4$, $\Delta T_i = 0.6$ sec, and $\Delta B_i = 20$ kbps. YoMo is configured with $\beta_a = 15$ sec. The used parameters performed best during our experiments, an extensive parameter study is the scope of future works. Mesh node *B* is connected to the access router of the university via Ethernet. The client is connected to mesh node *A*, which is configured as access point. On node *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. As our testbed is located in one of our laboratories, it is fully meshed. The indicated topology is created with restricting firewall rules.

For this experiment, the YouTube video "*Madagascar I like to move music video*" whereof a snapshot is shown in Fig. 1 is used. It has a playtime of 2:49 minutes and a file size of 6.8 MB. 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 video playback. To make it easier to generate overload on the links, the link rate is reduced to 1 Mbps. The Linux tool iperf generates the cross traffic as bidirectional TCP flows, which consume as much bandwidth as possible and bidirectional UDP flows at a fixed rate of 800 kbps. Two test scenarios are used: an in-band scenario where the cross traffic is generated at nodes *A* and *B*, and an out-band scenario where the cross traffic is between nodes *C* and *D*. For each of the four combinations of in-band/out-band, TCP/UDP we perform two types of 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 type of experiments, OTC is disabled, for the second type, it is enabled. During all experiments, all packets are captured with tcpdump and analyzed with Wireshark.

The four figures of Fig. 3 show the results from the four different experiments carried out for the out-band scenario. The results for the in-band scenario are very similar and can be found in [22]. Fig. 3(a) and 3(b) depict the experiments without, Fig. 3(c) and 3(d) the experiments with OTC. Each of the four figures contains three subfigures visualizing the behavior of the three different considered parameters. In the topomost subfigure, the bandwidth of the three flows are shown. Here, the area at the bottom accounts for the bandwidth consumption of the flow from nodes *D* to *C*, the area above for the bandwidth consumption of the flow from nodes *C* to *D*, and the area on top for the bandwidth consumption of the YouTube traffic. The subfigure in the middle shows the player state which is either buffering (b) or playing (p). The



(a) UDP cross traffic without OTC

(b) TCP cross traffic without OTC

(c) UDP cross traffic with OTC

(d) TCP cross traffic with OTC

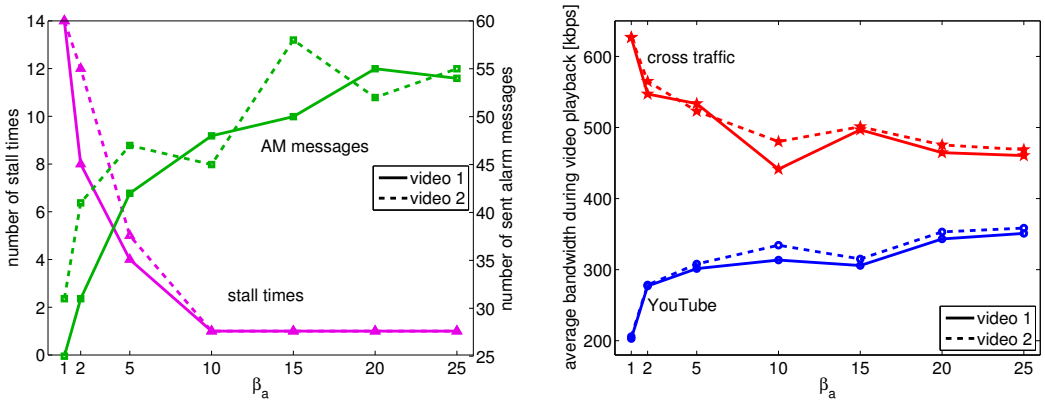
Fig. 3. Video playback with out-band cross traffic

subplot at the bottom depicts the amount of buffered playtime β, where, in Fig. 3(c) and 3(d), a dashed line visualizes the alarm threshold β_a.

The fact that the buffer playtime is always under the critical value in the experiments with a disabled OTC visualized by Fig. 3(a) and 3(b) already illustrates that the YouTube AC in this case is always bad and the video is moreover repeatedly stalling. Let us now discuss the results obtained for these experiments more closely. In the presence of both TCP and UDP cross traffic without OTC the YouTube flow gets about one third of the total bandwidth. The representation of the player state and of the buffering level in Fig. 3(a) and 3(b) demonstrate that this is not enough to guarantee a smooth playback but alternates between buffering and playing. The reason for this is is visualized by the zig-zag shape shown by the subfigure at the bottom: the buffered playtime increases as long as the player stalls. As the download rate is smaller than required, the buffer shrinks again as soon as the video playback starts and the player stalls again. The small amount of available bandwidth also causes the download not to complete until the 230 sec of the measurement we show, whereas the playtime of the video is only 169 sec. Moreover does the video stall 41 times regardless the cross traffic type.

Let us now see if and how YoMo and OTC improve the situation. During the experiments with OTC, the video playback is terminated normally which is illustrated by the fact that the player state depicted in the middle subfigures of Fig. 3(c) and 3(d) does not change to buffering after the initial buffering phase any more. The reason for this is that YoMo sends a DM and AMs until β is larger than the alarm threshold as soon as it has detected the first FLV tag. As a result nodes *C* and *D* throttle their traffic, an effect which translates to decreasing cross-traffic bandwidth shown in the upper-most subfigure. The zig-zag shape of the bandwidth consumption curves of Fig. 3(c) illustrates that as soon as no AM messages are sent any more, the UDP cross traffic increases, thereby causing the YouTube buffer to decrease and the controlling algorithms has to start again. In contrast, the TCP congestion control mechanism limits the increase of the cross traffic (cf. Fig. 3(d)).

A comparison between the player states shown in Fig. 3(a) and 3(c), and Fig. 3(b) and 3(d) respectively visualizes that the video playback does not begin faster with OTC than without. The reason for this is that YoMo detects the YouTube connection only after the first FLV tag has been downloaded. The download of the YouTube player and the first tag are hence not prioritized. Currently we are working on mechanisms suitable for recognizing a YouTube flow before the first FLV



(a) Stallings and alarm messages

(b) Bandwidth consumptions

Fig. 4. Influence of the alarm buffer threshold β_a

tag has been downloaded in order to speed up the begin of the playback. The representation of the cross traffic bandwidth in Fig. 3(c) visualizes another inherent problem of YoMo: Shortly before the video download is completed, the playtime buffer reaches the alarm threshold, AMs are sent, and hence nodes *C* and *D* limit the best effort traffic. In this case this is unnecessary, as the video is nearly completely loaded and the YouTube flow does not need as much bandwidth any more. A future extension for YoMo will hence incorporate the status of the download progress in the decision about sending AMs.

B. 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 create a higher degree of congestion. This is achieved by generating bidirectional TCP in-band cross traffic and bidirectional UDP out-band cross traffic at the same time. As a result, the YouTube flow gets at most 1 kbps before YoMo can send the first message. For each experiment, the cross traffic is started shortly before the client starts to display the video. YoMo is always activated with different values for the buffer alarm threshold. Results from this experiment are summarized in Fig. 4.

Observe first that the number of stall times visualized by Fig. 4(a) does not decrease to zero, even if β_a is very large. The reason for this is that the used video has a particular buffering behavior (cf. Fig. 3(c) and 3(d)): β increases until it is roughly larger than 2 seconds. The playback begins and β decreases nearly to 0 before it starts increasing again. In this experiment, where the network is more heavily congested, this buffering behavior that for all parameterizations of β_a the video stalls after the first 3 seconds of playback, as the playtime buffer is not refilled fast enough. Therefore, we repeated the experiment with a second video, "Al Hirt Rocky"[3] 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 larger than the previously considered *video 1*, the behavior of the buffer status shows the same trend and the video also stalls after the first seconds of playback. Analyzing or optimizing the buffering behavior of YouTube videos is not our goal, we just want to point out that this mechanism is the reason why both videos are stalling at least one time, even if $\beta_a \geq 10$ sec.

A closer analysis of Fig. 4(a) illustrates the drawbacks of unsuitable values for β_a: If the alarm threshold is chosen too small, 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 refill its buffer fast enough. If in contrast, β_a is chosen too large, more overhead in terms of AMs is caused. 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 and the video stalls again. This allows the buffer to fill quickly and only a few messages are sent until β is again larger than the alarm threshold. If in contrast, a larger threshold is chosen, the video playback continues longer thereby consumes a larger amount of buffered playtime. Consequently, it takes longer until β is again larger than β_a and more AM messages are sent. At the moment, OTC is configured to send an AM each two seconds as long as β is below the threshold which we found to be a good compromise between reactivity and overhead. Other message frequencies would change the quantitative outcome of this experiment, but not its qualitative statement.

The price for avoiding a stalling of a YouTube video is not only an increased AM message overhead, but also a reduction of the best effort traffic. In Fig. 4(b) we illustrate the positive and negative consequences of β_a by visualizing the bandwidth consumptions of YouTube and the cross traffic averaged over the video playback time. Recall from the last experiment (cf. Fig. 3(a) and 3(b)) that before YoMo and OTC start to control the bandwidth in the network, the YouTube flow gets roughly 1 kbps. Small values for $\beta_a < 10$ sec already increase this bandwidth, but are not suitable for guaranteeing the roughly 300 kbps required for a smooth playback. This threshold is only only reached if $\beta_a \geq 10$ sec at the price

[3] http://www.youtube.com/watch?v=X60jEHxEAY4, last accessed 01/11

of a significant restriction of the best effort traffic. Observe that values in the range of $\beta_a = 5$ sec are insofar interesting as they allow to strike the balance between a smooth video playback, overhead and reduction of best effort traffic.

The results presented in this section are suitable for illustrating the configuration possibilities of YoMo, but 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 and in larger topologies are hence the subject of our future studies.

VI. Conclusion and Outlook

In this work, the idea of the application and QoE aware radio resource management framework Aquarema has been introduced. For demonstrating Aquarema's appeal, we exemplarily show how the YouTube QoE can be improved if a tool is monitoring the YouTube application comfort and communicates this information to a WMN resource management tool. Monitoring the YouTube AC is done by YoMo which is able to identify a YouTube video and to determine its buffered playtime. Thus, YoMo is able to detect an imminent QoE degradation, i.e. a stalling of the video. In this case, the YouTube flow lacks bandwidth and YoMo sends alarm messages which cause both the forwarding nodes and the one-hop neighbors to limit their best effort traffic. We are able to show that the cooperation of YoMo and the bandwidth shaper OTC successfully avoids stall times for the case where the cross traffic is on a link used by the YouTube flow as well as on an interfering link.

The core idea of Aquarema is application comfort monitoring which enables a cross-layer resource management approach as the usage of applications, their quality requirements, and the experienced AC are measured at the client and communicated to a resource management instance. AC goes beyond QoE because it also considers the future development of the QoE and consequently allows to trigger resource management decisions in order to avoid QoE degradations. If users run YoMo or a similar tool, the provider gets valuable information which it may use for improving the user QoE for specific applications by running a network advisor tool which is able to trigger tools similar to the discussed OTC.

Our future work will therefore be dedicated to developing Aquarema 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 towards improving the Aquarema architecture in order to enable intelligent access mechanisms, gateway selection, channel selection, and routing for WMNs in the presence of various applications.

Acknowledgment

The authors would like to thank Phuoc Tran-Gia and Robert Henjes for their support and valuable comments.

This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (Förderkennzeichen 01 BK 0800, GLab). The authors alone are responsible for the content of the paper.

References

[1] Cisco Systems Inc., "Cisco Visual Networking Index - Forecast and Methodology, 2009-2014," White Paper, June 2010.

[2] F. Wamser, R. Pries, D. Staehle, K. Heck, and P. Tran-Gia, "On Traffic Characteristics of a Broadband Wireless Internet Access," *Special Issue of the Telecommunication Systems Journal*, 2010.

[3] "TR-059: DSL Evolution - Architecture Requirements for the Support of QoS-Enabled IP Services," DSL Forum, Tech. Rep., September 2003.

[4] "Traffic Inspection for Visibility, Control and New Business Opportunities," Ericsson White Paper, February 2008.

[5] M. Tsagkaropoulos, I. Politis, T. Dagiuklas, and S. Kotsopoulos, "Enhanced Vertical Handover based on 802.21 Framework for Real-time Video Streaming," in *MobiMedia'09*, London, UK, September 2009.

[6] R. Pries, D. Hock, N. Bayer, M. Siebert, D. Staehle, V. Rakocevic, B. Xu, and P. Tran-Gia, "Dynamic Bandwidth Control in Wireless Mesh Networks: A Quality of Experience Based Approach," in *18th ITC Specialist Seminar*, Karlskrona, Sweden, May 2008.

[7] I. Akyildiz, X. Wang, and W. Wang, "Wireless Mesh Networks: A Survey," *Computer Networks*, vol. 47, no. 4, 2005.

[8] A. Van Moorsel, "Metrics for the Internet Age: Quality of Experience and Quality of Business," in *PMCCS 5*, Erlangen, Germany, September 2001.

[9] A. Raake, "Short- and Long-Term Packet Loss Behavior: Towards Speech Quality Prediction for Arbitrary Loss Distributions," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 6, 2006.

[10] T. Hoßfeld and A. Binzenhöfer, "Analysis of Skype VoIP Traffic in UMTS: End-to-end QoS and QoE Measurements," *Computer Networks*, vol. 52, no. 3, 2008.

[11] U. Engelke and H. J. Zepernick, "Perceptual-based Qality Metrics for Image and Video Services: A Survey," in *3rd EuroNGI Conference*, Trondheim, Norway, May 2007.

[12] J. Gustafsson, G. Heikkila, and M. Pettersson, "Measuring Multimedia Quality in Mobile Networks with an Objective Parametric Model," in *ICIP'08*, San Diego, CA, USA, December 2008.

[13] A. C. Dalal and K. Purrington, "Discerning User-perceived Media Stream Quality through Application-Layer Measurements," in *MSAN'05*, Orlando, FL, June 2005.

[14] IEEE Std. 1900.4, "Architectural Building Blocks Enabling Network-Device Distributed Decision Making for Optimized Radio Resource Usage in Heterogeneous Wireless Access Networks", February 2009.

[15] G. Lampropoulos, H. Harada, A. Salkintzis, and N. Passas, "Media-Independent Handover for Seamless Service Provision in Heterogeneous Networks," *IEEE Communications Magazine*, vol. 46, no. 1, 2008.

[16] E. Ong and J. Khan, "Cooperative Radio Resource Management Framework for Future IP-based Multiple Radio Access Technologies Environment," *Computer Networks*, Article in press, corrected proof.

[17] Bullot, T. and Gaïti, D. and Pujolle, G. and Zimmermann, H., "A Piloting Plane for Controlling Wireless Devices," *Telecommunication Systems*, vol. 39, no. 3, 2008.

[18] T. Hoßfeld, P. Tran-Gia, and M. Fiedler, "Quantification of Quality of Experience for Edge-based Applications," in *ITC20*, Ottawa, Canada, June 2007.

[19] T. M. Bohnert, D. Staehle, G.-S. Kuo, Y. Koucheryavy, and E. Monteiro, "Speech Quality Aware Admission Control for Fixed IEEE 802.16 Wireless MAN," in *ICC'08*, Beijing, China, May 2008.

[20] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, and W. Kellerer, "Application-Driven Cross-Layer Optimization for Video Streaming over Wireless Networks," *IEEE Communications Magazine*, vol. 44, no. 1, 2006.

[21] A. C. Dalal, D. Musicant, J. Olson, B. McMenamy, S. Benzaid, B. Kazez, and E. Bolan. Predicting User-Perceived Quality Ratings from Streaming Media Data. In *ICC'07*, Glasgow, Scotland, UK, June 2007.

[22] B. Staehle, M. Hirth, F. Wamser, R. Pries, and D. Staehle, "YoMo: A YouTube Application Comfort Monitoring Tool," University of Würzburg, Tech. Rep. 467, March 2010.

[23] B. Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, "YoMo: A YouTube Application Comfort Monitoring Tool," in *QoEMCS*, Tampere, Finland, June 2010.