Proxy-based TCP-friendly streaming over mobile networks

Lei Huang
University of Southern California
Department of Electrical Engineering-Systems
Los Angeles, CA 90089-2564, USA
leihuang@usc.edu

Uwe Horn
Ericsson Research
Mobility Applications Laboratory
Herzogenrath, Germany
Uwe.Horn@eed.ericsson.se

Frank Hartung
Ericsson Research
Mobility Applications Laboratory
Herzogenrath, Germany
Frank.Hartung@eed.ericsson.se

Markus Kampmann
Ericsson Research
Mobility Applications Laboratory
Herzogenrath, Germany
Markus.Kampmann@eed.ericsson.se

ABSTRACT

Mobile media streaming is envisioned to become an important service over packet-switched 2.5G and 3G wireless networks. At the same time, TCP-friendly rate-adaptation behavior for streaming will become an important IETF requirement. In this paper we investigate TCP-friendly on-demand streaming over wired and wireless links. We consider two approaches for achieving TCP-friendliness: first, by tunneling RTP packets over TCP and secondly by employing an RTP server rate control which does not exceed a variable rate constraint derived from the recently developed TFRC protocol. To allow a reasonable fair comparison between TCP and TFRC, we assume a simple retransmission mechanism on top of TFRC. We consider streaming from a server in the public Internet to both wired and wireless clients. For the wireless case we assumed a client which is connected to the public Internet via a dedicated 64 kbps WCDMA streaming bearer. Simulation results carried out in ns-2 show that TCP and TFRC can not fully utilize the WCDMA bearer at 5% packet loss rate over the shared public Internet link. Smooth playout of a typical 64 kbps video stream would require high initial buffering delays (> 10 seconds) and large receiver buffer sizes (> 60 KB). We finally investigate the gains from a proxy that splits the connection and uses TCP-friendly congestion control only over the shared part of the client-server connection. Simulation results show improvements in average throughput and wireless link utilization. By employing appropriate packet rescheduling mechanisms, the initial buffering delay and the client buffer size for a typical 64 kbps video stream can be decreased by a factor of three to four.

keywords: mobile streaming, TCP-friendly rate control, bandwidth smoothing, packet scheduling, proxy

WoWMoM'02, September 28, 2002, Atlanta, Georgia, USA.

1. INTRODUCTION

With the rapid development of high-speed internetworks, streaming media applications have become important components of multimedia communications in today's wired Internet. Streaming technologies for transmitting real-time stored continuous audio and/or video media over Internet pose many challenges in various areas including media compression, application-QoS control, continuous media distribution services, streaming servers, media synchronization mechanisms, and protocols for streaming media [1].

In the near future, third-generation (3G) mobile communication systems will extend the scope of streaming solutions by introducing standardized streaming services [2, 3], targeting the mobile user's specific needs. By offering data-transmission rates up to 384Kbps for wide-area coverage and 2Mbps for local-area coverage, 3G systems will be able to provide high quality streamed Internet content to the rapidly growing mobile market. When extending streaming services to wireless users, new challenges need to be addressed, such as the specific characteristics of wireless links and terminal and access network heterogeneity [4].

With respect to the architecture and the used protocols, wireless streaming is similar to wired Internet streaming. A wireless streaming service consists of a server and a wireless client. The data sent from the server to the client (e.g. compressed video frames or audio samples) contain presentation timestamps that tell the client at what time a certain chunk of data need to be played out. Therefore, streaming requires timely delivery of data to ensure their availability at the client when they are needed for playback. On the other hand, due to the nature of audio and video compression schemes, limited data losses can be tolerated if error concealment methods are applied at the client. Thus, streaming applications have significantly different requirement compared to pure download applications, the latter requiring error-free data delivery without any specific time constraints.

By means of special receiver buffer, a streaming client can tolerate variations in the available bandwidth, as long as the

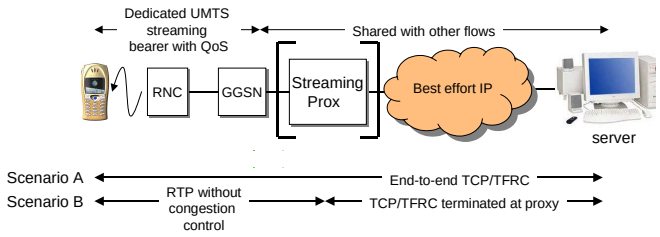

Figure 1: Illustration of investigated streaming scenarios: (a) end-to-end scenario, (b) proxy-based scenario

average bandwidth does not fall below a critical threshold. Compensation of the fluctuation of the incoming data rate also requires an initial buffering delay, which refers to the time during which incoming data are pre-buffered before the actual playback starts. The required size of the receiver buffer and the initial buffering delay depend on two factors: the encoding of the content and the available bandwidth during the actual transmission of the compressed stream. For mobile streaming, both the buffer size and the initial buffering delay should be as minimal as possible. First, since fast memory is still a relevant cost factor in mobile terminals, and secondly since initial buffering delays of more then 4 to 8 seconds are hardly tolerable if the duration of the streamed clips is below one minute.

The commonly used protocol for streaming media data from the server to the client is RTP [5]. RTP uses UDP as underlying transport protocol and has, unlike TCP, no built-in congestion control. Thus, RTP does not reduce its transmit rate in case of congestion. When TCP and RTP are transported over the same congested link, TCP reduces the rate, while RTP does not. Effectively, RTP gets a higher share of the available link bandwidth. The so-called "TCP-unfriendly" protocol behavior has the potential danger of congestion collapse [6]. Therefore, it is expected that in the near future TCP-friendly rate-adaptation behavior for applications not based on TCP (e.g., streaming) will become an important requirement.

Fig. 1 shows the mobile streaming scenarios we are investigating in this paper. We assume a mobile streaming client connected to the server via a WCDMA [7] streaming bearer and an IP best effort network. Scenario A simply denotes TCP-friendly end-to-end streaming to a wireless client. Scenario B places a proxy between the public Internet and the mobile network. In both cases the wireless part of the connection consists of a WCDMA streaming bearer. This bearer is solely dedicated to RTP traffic and provides a guaranteed quality of service (QoS) for streaming applications in terms of bit-rate and delay jitter [8]. Thus no congestion control for this part of the connection is required. However, over the best effort IP network, the traffic between server and client competes with other applications for available resources and therefore has to be TCP friendly.

In this paper, we investigate two approaches for achieving TCP friendliness over the shared public Internet. First, by tunneling the RTP traffic via a TCP which is TCP-friendly by definition. The TCP tunnel is either terminated at the client (scenario A of Fig. 1), or it is terminated at a proxy (scenario B of Fig. 1). In the second case we assume a TCP friendly rate control at the server which ensures that the RTP packets are sent at a rate no higher than what is regarded as a TCP-friendly share of the available bandwidth. TCP-friendly rate-adaptation is provided in this case by the recently proposed TCP-friendly rate control (TFRC) protocol [9, 10]. To allow a reasonable fair comparison between TCP and TFRC, we assume a simple retransmission mechanism on top of TFRC as explained later in more detail.

Throughout this paper we focus on the on-demand streaming case, which means that the entire video stream is available at the server. Moreover, we assume a priori knowledge of the variable rate constraint imposed by a TCP-friendly rate control. Although this is an unrealistic assumption in real streaming scenarios, it is a valid simplification if we are only interested in comparing best-case scenarios.

Our comparison is mainly based on initial buffering delay and buffer size as the two most important key parameters of a streaming client. We obtain those values from simulated transmissions of a real video stream under the requirement of smooth playout (e.g. no buffer underflows) of the stream at the client. In all cases variable rate constraints derived from simulated TCP and TFRC transmissions are used to compute the resulting initial buffering delay and client buffer size.

The rest of this paper is organized as follows. Section 2 reviews some related work on TCP-friendly streaming, including TCP-friendly rate control, TCP proxy, and smoothing algorithms for variable-bit-rate video. Section 3 describes the algorithms used in this paper for computing minimum client buffer size and minimum initial buffering delay under a variable rate constraint. Finally, section 4 describes the simulation environment we used for obtaining realistic TCP and TFRC throughput curves and presents results for

- streaming under a fixed rate constraint
- end-to-end TFRC / TCP streaming to wired and wireless clients
- proxy-based TFRC / TCP streaming to wireless clients.

Finally, our conclusion and possible future work are given in Section 5.

2. RELATED WORKS

This section gives a brief overview about recent work on TCP-friendly rate control (TFRC), TCP proxies and traffic smoothing for variable-bit-rate video.

2.1 TCP-friendly rate control

The main purpose of TCP-friendly rate control is that they achieve the fairness between TCP and non-TCP flows, and improve the overall network utilization. Besides, they attempt to eliminate some drawbacks of TCP when used in real-time applications like streaming. For instance, in response to a congestion indicated by single packet loss, which is unnecessarily severe for real-time streaming, TCP halves the sending rate, thus could noticeably reduce the user-perceived quality [11].

TCP-friendly rate control schemes for Internet video proposed to date can be divided into two categories. One is to mimic the TCP congestion control mechanism directly by adopting the AIMD (additive increase multiplicative decrease) [12, 13, 14]. The other one is to utilize the TCP

throughput equation. The sender estimates the current TCP's throughput and also sends its data bounded by this throughput value [11, 15, 9]. The significant difference between them is how to translate the congestion signal and how to behave to congestion. The first approach suffers from unnecessary rate fluctuation, which is not desirable for video streaming applications, since AIMD rule is sensitive to short-term network status. The second approach can prevent this unnecessary rate fluctuation by considering time-average of network status and transmitting in average TCP throughput sense.

The TCP throughput equation is given in Eq. 1 as a function of packet loss rate and round trip time (RTT) for a bulk transfer TCP flow [16]

$$TCPthroughput = \frac{MTU}{RTT\sqrt{\frac{2p}{3}} + T_0\sqrt{\frac{27p}{8}}p(1+32p^2)} \quad (1)$$

where MTU is the maximum transmission unit(packet size), RTT is the round trip time, T_0 is the retransmission time out, and p is the loss rate. This model captures not only the behavior of TCP's fast retransmit mechanism but also the effect of TCP's timeout mechanism on throughput. Eq. 1 can be simplified as

$$TCPthroughput \leq \frac{1.22MTU}{RTT\sqrt{p}} \quad (2)$$

which is the upper bound of TCP throughput [6].

The main issue of the TCP throughput equation-based rate control schemes is how to estimate the loss rate p and round trip time RTT timely and in accurate manner for non-TCP flows (in here, streaming video applications). Among this approach, some schemes [11, 15] utilize the ACKs to estimate the loss rate and RTT, which can be called sender-based scheme. There are potential problems of this approach because not only it might misinterpret the ACKs but also overload the sender side.

Recently, an efficient equation-based congestion control, called TFRC (TCP-friendly rate control) protocol, is proposed in [9]. This scheme assigned the network estimation to receiver, and introduced the weighting method to average the packet loss rate p over a number of loss intervals, so that it prevents the diverge of Eq. 1. On the other hand, this scheme deployed exponentially weighted moving average (EWMA) to calculate the average RTT in Eq. 1, thus prevent the unnecessary oscillation from RTT variance. Based on these measurements, the sender adjusts packet scheduling accordingly by setting the inter-packet spacing as follows:

$$t_{inter-packet} = \frac{s \times \sqrt{R_0}}{Throughput \times M} \quad (3)$$

where R_0 is the most recent RTT sample, and M is the average of the square-roots of the RTTs, calculated using EWMA.

It has been shown that the TFRC protocol achieves the TCP-friendliness while prevents the unnecessary fluctuation of rate prediction, by appropriate estimation of packet loss rate and round trip time. As a promising TCP-friendly congestion control mechanism for Internet streaming applications, it has been in the IETF standardization track [10]. Obviously, the actual packet loss rate and RTT behaviors have important impact on its performance. In this paper, we investigate its performance in wireless streaming environments, where packet loss rate and RTT exhibit different characteristics from those in Internet.

2.2 TCP proxy

The investigation of a proxy scenario in this paper is motivated by previous research work on extending TCP to wireless networks. One outcome of this work was that under certain circumstances TCP has difficulties to fully utilize the capacity of the mobile link. Main reason is the increased end-to-end round-trip time caused by the link layer protocols used over the radio link combined with packet losses that may happen under loaded network conditions in the public Internet. Various TCP improvements have been developed in the past to reduce the impacts of increased delays over wireless links. Just recently, the introduction of a TCP proxy was proposed in [17]. Based on simulations, it was shown that a TCP proxy can provide a significant gain in link utilization and throughput for high data rate bearers, because it shields the wireless link from packet losses in the Internet. Other TCP-friendly rate adaptation applications most likely suffer from the same problem, since the end-to-end rate adaptation mechanism deploys a similar approach to the congestion control used in TCP.

There are two main difference between the work presented in this paper and the previous work on TCP proxies. First, instead of looking just at the throughput, in all our investigations we take into account streaming specific key performance parameters like required buffer size at the client and initial buffering delay. Secondly, in our case we do not employ any further congestion control over the link between the proxy and the client since we assume the existence of an UMTS streaming bearer which carries only streaming media data. Fairness with other TCP traffic over this part of the connection is therefore not required.

2.3 Traffic smoothing of variable-bit-rate streams

For video streaming, the best subjective image quality is achieved by constant quality encoding. Constant quality encoding employs a rate control which tries to encode a video at a fairly constant quality regardless of the scene complexity. Constant quality encoding typically results in variable-bit-rate streams.

Although constant quality encoding is beneficial for the subjective image quality, the burstiness of the resulting bit-stream complicates the provisioning of network resources. For stored video applications, the server can smooth the variable bit-rate stream by transmitting frames into the client playback buffer in advance of each burst. This can be achieved by algorithms which generate a server transmission schedule according to the available knowledge of video source and network conditions, as well as the limitations and requirement of the client. A good scheduling algorithm is characterized by efficient use of available network resource without overflowing or underflowing the client buffer, while at the same time minimizing the required buffer size and buffering delay at the client.

Different packet scheduling algorithms for video streaming applications have been proposed to achieve a variety of objectives. To reduce the network resource required for transmitting variable-bit-rate video traffic, a couple of bandwidth smoothing algorithms [18, 19, 20, 21, 22, 23] generate a transmission plan that consists of a number of fixed-rate runs of transmission. Given a priori knowledge of frame sizes of the entire stored video sequence, these offline scheduling

algorithms aim at different optimization under the buffer size constraints. For examples, [18, 19] minimize the number of bandwidth increases, [20] minimizes the total number of bandwidth changes, [21] minimizes the variability of the bandwidth requirements, while [22] generates periodic bandwidth runs. A comparison of several different schemes can be found in [24]. Online scheduling problems without the complete prior knowledge of the entire video sequence has also been addressed in [25].

In this paper, packet scheduling algorithms are employed to minimize the client buffer size and initial buffering delay subject to a variable rate constraint which represents a fair and TCP-friendly share of the available bandwidth. The scheduling algorithms employed in this paper are based on the fundamental results presented in [26] and will be explained in more detail in the next section.

3. MINIMIZING BUFFER SIZE AND INITIAL BUFFERING DELAY FOR STREAMING UNDER VARIABLE RATE CONSTRAINT

In the following we introduce the terminology required for understanding the computation of optimal schedules for transmitting variable-bit-rate traffic like video without exceeding a variable rate constraint.

We first introduce a general link model as the basis for computing optimal schedules for both the end-to-end and the proxy case.

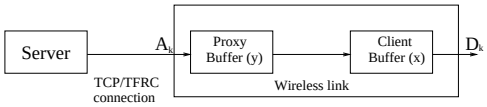

Figure 2: Model of proxy-based streaming system. As explained in the text, end-to-end streaming is contained as a special case.

The general link model consists of an ingress node that performs work-ahead transmission of the arriving video stream into the egress node as shown in Fig. 2. The incoming stream is characterized by an arrival vector $A = (A_0, \ldots, A_N)$, where A_k corresponds to the amount of data that has arrived at the ingress node by time $k = 0, 1, \ldots, N$, where $A_k \geq A_{k-1}, k = 1, \ldots, N$. The stream has a playout vector $D = (D_0, \ldots, D_N)$, where D_k corresponds to the amount of data that must be removed from the x-bit egress buffer by time $k = 0, 1, 2, \ldots, N$, with $D_0 = 0$ and $D_k \geq D_{k-1}$, $k = 1, \ldots, N$. For a valid playout vector of the arriving stream, we assume that $A_k \geq D_k$ for $k = 0, 1, \ldots, N$, with $A_N = D_N$ at the end of the transfer. At time k a total of $A_k - D_k$ bits are distributed between the x-bit egress buffer and the $y = (M - x)$-bit ingress buffer, requiring $A_0 \leq y$ and $A_k - D_k \leq M$ for $k = 0, 1, \ldots, N$. Any valid transmission schedule S must avoid underflow and overflow of both the egress and the ingress buffer.

Fig. 3 illustrates the relationship between transmission schedule S_k, playout vector D_k, initial buffering delay w, and receiver buffer size x.

For our investigations it is necessary to introduce a variable rate constraint R_k, for $k = 0, 1, \ldots, N$ where R_k denotes the maximum amount of data available at the client at time k.

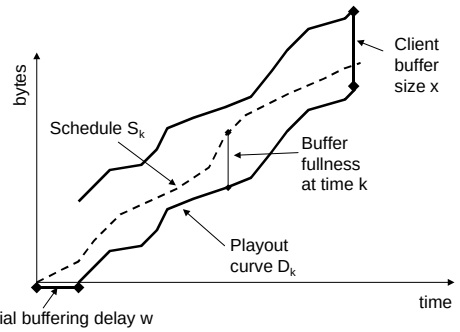

Figure 3: Relationship between transmission schedule S_k, playout vector D_k, initial buffering delay w, and client buffer size x.

For the reasons given in Section 1, we focus on minimizing both the initial buffering delay and the client buffer size. In the following we discuss the necessary optimization steps separately for the end-to-end and the proxy case.

3.1 Minimizing initial buffering delay and client buffer size for end-to-end streaming

Server/client streaming without an intermediate proxy is a special case in the general model depicted in Fig. 3. In this case $y \geq D_N$ and $A_k = D_N$ for all $k = 0, 1, 2, \ldots, N$ holds, since the video stream is completely stored at the server. Under the assumption of stored video it is possible to minimize both the initial buffering delay w and the client buffer size x by generating a schedule S^{late} that transmits frames as late as possible subject to the rate constraint R_k. S^{late} is computed as

$$S_k^{late} = \begin{cases} D_N & k = N \\ max\{S_{k+1}^{late} - R_k, D_k\} & k < N \end{cases}$$

From S^{late} we can compute the minimum client buffer size as

$$x^* = max\{S_k^{late} - D_k\}.$$

and the minimum initial buffering delay as

$$w^* = min\{w | A_k - S_{k-w}^{late} \geq 0\}.$$

3.2 Minimizing initial buffering delay and client buffer size for proxy-based streaming

The general link model described above is also applicable to the proxy case. In this case, A_k is the data arrival vector at the proxy, the ingress buffer corresponds to the proxy buffer and the egress buffer is the buffer in the client as in the previous case.

The feasible transmission schedules in the proxy case are then bounded by the following conditions:

Upper bound (U): As long as there are data in the proxy buffer, transmit them as fast as possible to the client without overflowing the client buffer.

Lower bound (L): Deliver enough data from the proxy to

the client to allow smooth playout but avoid overflow of the proxy buffer.

For computation of the minimum buffer size and the minimum initial buffering delay, we introduce the schedule S^{early}, which transmits video frames as early as possible, subject to a variable rate constraint. S^{early} is then computed as

$$S_k^{early} = \begin{cases} 0 & k = 0 \\ min\{S_{k-1}^{early} + R_k, A_k\} & k > 0 \end{cases}$$

S^{early} minimizes the ingress (proxy) buffer size without any constraints on the egress (client) buffer size.

The procedure for minimizing the client buffer size and the initial buffering delay in the server-proxy-client scenario requires two steps as described in the following:

Step 1. In the first step, we assume that the *proxy* has stored the complete video stream ($A_k = D_N$ for all k). We then compute S^{late} between proxy and client subject to the rate constraint $r^{wireless}$. From S^{late} we obtain the minimum end-to-end start-up delay w^* and the minimum client buffer size x^*.

Step 2. In the second step, we compute the earliest scheduling $S^{early'}$ between server and proxy that sends as early as possible subject to the upper bound condition $U = \min\{D_{k-w^*} + vec(x^*), R_k\}$. From $S^{early'}$ we can compute the required buffer size at the proxy $y'(w^*)$.

By this approach we minimize both the initial buffering delay and the client buffer size without exceeding the upper and lower bounds described above.

4. SIMULATION RESULTS

The simulation results presented in this section are based on a real video stream which was generated from the test sequence "Glasgow". The original sequence was encoded with an H.263 video coder in baseline mode at QCIF resolution with 15 frames per second. Constant quality encoding with a fixed quantizer (Q= 18) was used. The stepsize of the quantizer was set to a value which results in an average bitrate of 64 kbps for the first 30 seconds of the encoded stream. The playout vector D_k for the encoded video stream is shown in Fig. 4.

For this video stream we computed schedules according to the algorithms describes in section 3, minimizing both the initial buffering delay and the client buffer size based on the variable rate constraints obtained from TCP and TFRC throughput measurements in ns-2.

In the following, we first describe the ns-2 simulation environment that we used to generate realistic throughput curves for TCP and TFRC streaming to wired and wireless clients. Then we present the computed initial buffering delay and client buffer sizes obtained for the various streaming scenarios.

4.1 Obtaining TCP-friendly rate constraints

Fig. 5 shows the ns-2 simulation setup we used for generating throughput curves for TCP and TFRC streaming to wired and wireless clients. Note that the simulation set-up used in this work is very similar to the one used by Floyd *et al.* for the performance comparison between TCP and TFRC presented in section 4 of [9].

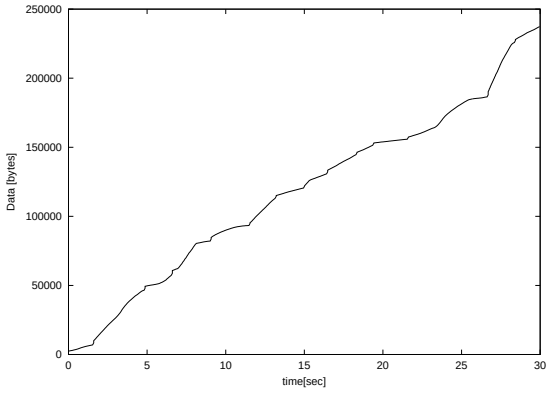

Figure 4: Playout vector D_k of video stream used in our experiments. The horizontal axis denotes time, the vertical axis denotes the overall amount of data played out at a specific time.

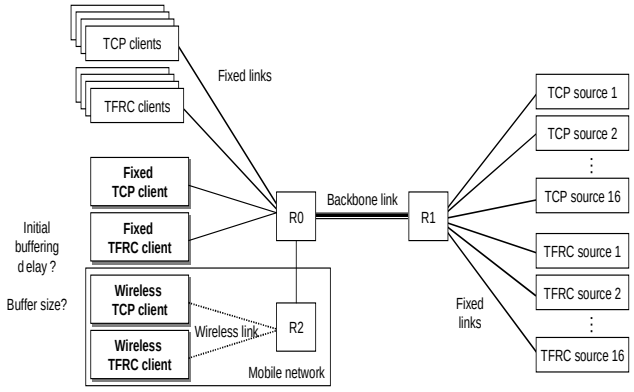

Figure 5: ns-2 simulation set-up for obtaining throughput curves for TCP and TFRC streaming to wired and wireless clients. The configuration and parameter settings of the ns-2 simulation environment are explained in the text. They are based on the settings used in section 4 of [9]

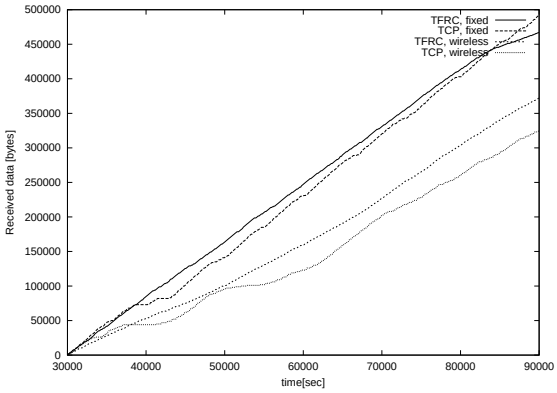

Figure 6: TFRC and TCP transmission behavior for wired and wireless clients. Note that the TFRC curves shown here were generated from deducting the actually measured curves by 5% to allow for one retransmission attempt in case of a packet loss.

In our simulation we used 16 TFRC and 16 TCP connections. Source and client nodes are connected via a backbone link with a maximum bandwidth of $32*64$ kbps $= 2048$ kbps. That means, if all flows fairly share the backbone capacity, each connection will get on average 64 kbps of the overall available bandwidth. The backbone delay is set to 75 ms.

All wired access links are configured with a bandwidth of 64 kbps and a delay of 2 ms. The wireless access link is modeled by an additional router with its own queue and packet dropping strategy connected to the backbone router. The wireless link has a bandwidth of 64 kbps, and introduces an additional delay of 200 ms.

The backbone load was controlled by inserting additional background traffic into the backbone. In our experiments we adjusted the background traffic load such that it results in an average packet loss rate of 5% over the backbone link.

Note that unlike TCP, TFRC does not provide any kind of retransmission mechanism. Since retransmission mechanism on top of TFRC are beyond the scope of this paper, we made some simplified assumptions. We assume that in case of a packet loss, only one retransmission attempt is made. If the retransmitted packet is lost again, it is regarded as lost for the application. If we assume a 5% packet loss rate, the final packet loss after one retransmission attempt is 0.25%. At such low loss rates appropriate error concealment techniques are sufficient to ensure that the degradation of audiovisual playback quality is almost unnoticeable to the end-user. To allow for at least one retransmission, we deducted 5% from the TFRC throughput curves obtained from the ns-2 simulations. We furthermore assume that the retransmission of a lost packet can be accomplished within one round-trip time. This assumption can be taken into account by increasing the initial buffering delay by one round-trip delay in case TFRC is used. All simulation results presented in the following take into account the simplified retransmission model described above.

Fig. 6 shows examples for the transmission behavior of TFRC and TCP to wired and wireless clients. The horizontal axis denotes time, the vertical axis denotes the overall amount of data received at the client. The figure show the last 60 seconds of a 90 second simulation run. Therefore, slow-start effects are not taken into account.

Case	R [kbps]	w^*[sec]	x^*[bytes]
TCP, wired	65.6	1.7	26567
TFRC, wired	62.2	2.8	32016
TCP, wireless	43.3	13.9	76840
TFRC, wireless	49.7	9.0	58320

Table 1: Minimum initial buffering delay and minimum buffer size for transmission of the test video stream subject to a fixed rate constraint.

Case	w^*[sec]	x^* [bytes]
TCP	4.1	43810
TFRC	2.1	30724

Table 2: Minimum initial buffering delay and minimum buffer sizes for transmission of the test video stream subject to a TCP-friendly rate constraint.

As can be seen, both TCP and TFRC achieve higher throughput over wired links. In this specific example the throughput was 62.2 kbps for TFRC and 65.6 kbps for TCP. When used over the wireless link, the throughput for both TCP and TFRC is lower. In the example it was 43.3 kbps for TCP and 52.3 kbps for TFRC. Fig. 6 also shows that compared to TCP, TFRC results in a smoother almost linear transmission curve which helps reducing both the initial buffering delay and the minimum client buffer size.

4.2 Streaming under a fixed rate constraint

As a reference for the other transmission scenarios discussed below, Table 1 summarizes the results for streaming subject to a fixed rate constraint. The fixed rate constraint was chosen as the average throughput of the curves shown in Fig. 6. As expected, the initial buffering delay and the minimum buffer size increases with decreasing average rate.

4.3 TCP-friendly streaming to wired client

Table 2 shows the results obtained for streaming to wired clients. Here the variable rate constraint was provided by the throughput curves from Fig. 6. Note that compared to TCP, TFRC results in a lower initial buffering delay and a smaller client buffer size.

4.4 TCP-friendly streaming to wireless client

Table 3 summarizes the results obtained for the investigated wireless streaming scenario. For the proxy case, the optimal schedule was computed according to the algorithm described in section 3.2. The rate constraint used in Step 1 is the fixed rate constraint of the dedicated streaming wireless link. The variable rate constraint used in Step 2 is given by the variable rate curve obtained for TCP (TFRC) streaming to a wired client (see Fig. 6). In other words, the receiving side of the proxy is treated like an ordinary wired client, since the TCP (TFRC) connection is terminated at the proxy.

Fig. 7 shows the transmission schedules for wireless streaming without proxy. Fig. 8 shows corresponding transmission schedules for wireless streaming with proxy.

Table 3 compares initial buffering delay and client buffer size for wireless streaming with and without proxy. As can be seen, introduction of a proxy leads to significant gains in

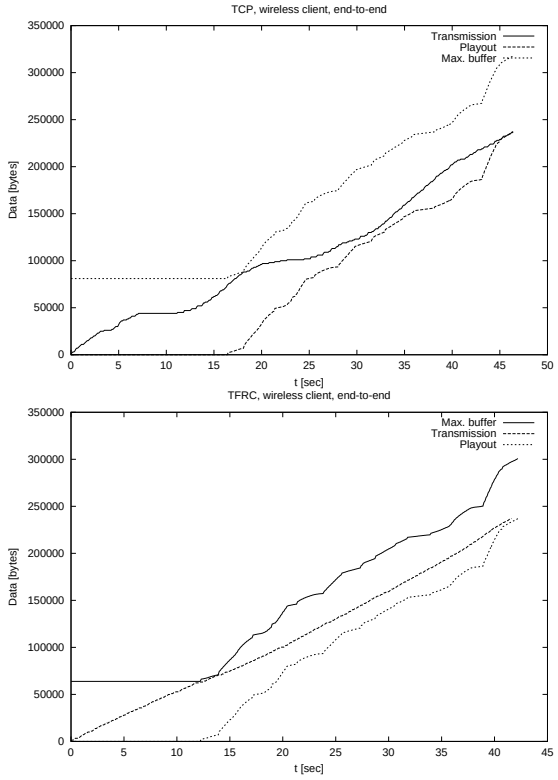

Figure 7: End-to-end streaming to a wireless client over TCP (top) and TFRC (bottom).

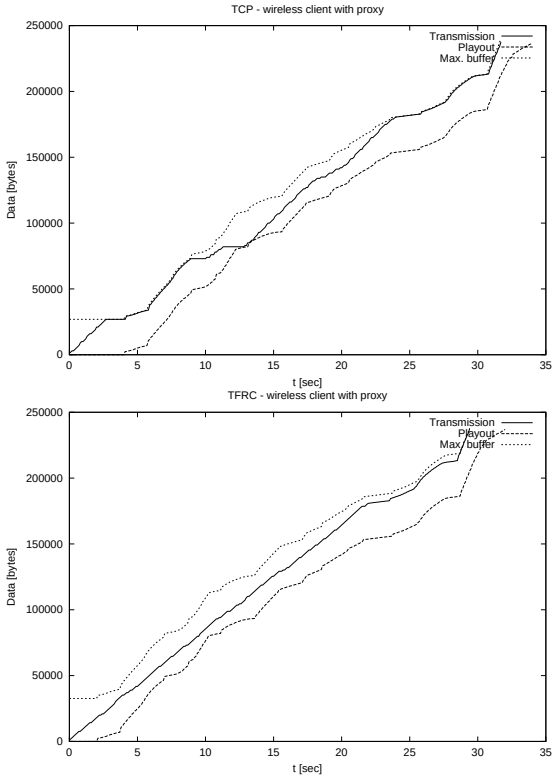

Figure 8: Proxy-based streaming to a wireless client over TCP (top) and TFRC (bottom).

both the initial buffering delay and the client buffer size. In the TCP case, the initial buffering delay is reduced from 16.5 seconds down to 4.1 seconds. At the same time the client buffer size is reduced from 81030 bytes down to 26924 bytes. In the TFRC case the initial buffering delay is reduced from 12.2 seconds down to 2.2 seconds. At the same time the client buffer size is reduced from 63833 bytes down to 32602 bytes.

5. CONCLUSION AND FUTURE WORK

In this paper we investigated TCP-friendly streaming over wired and wireless links. TCP-friendliness was achieved by tunneling RTP packets over TCP and by assuming an RTP server rate control which does not exceed a variable rate constraint derived from the recently developed TFRC protocol. Although TFRC in contrast to TCP does not provide a retransmission mechanism, retransmission mechanism on top of TFRC was taken into account by some simplified assumptions.

Without proxy			
Case	w^*[sec]	x^*[bytes]	y'[bytes]
TCP	16.5	81030	n.a.
TFRC	12.5	63833	n.a.
With proxy			
TCP	4.1	26924	26924
TFRC	2.2	26924	32602

Table 3: Minimum initial buffering delay and minimum buffer sizes for wireless streaming with and without proxy. y' denotes the required buffer space in the proxy.

Furthermore, we considered a network architecture consisting of both wired and wireless clients. In both cases the server is connected to the public Internet. For the wireless case we assumed a client which is connected to the public Internet via a dedicated UMTS streaming bearer.

The ns-2 simulation results show that end-to-end TCP and TFRC can not fully utilize the wireless UMTS link for packet loss rates around 5% over the shared public Internet link. Wireless streaming in this scenario requires high resource in terms of client buffer size, and results in a poor service quality in terms of initial buffering delay.

To solve those problems, we propose a proxy that splits the connection and uses TCP-friendly congestion control only over the shared backbone. Simulation results showed significant improvements in average throughput and link utilization. For a realistic test video stream the initial buffering delay and the client buffer size could be decreased by a factor of three to four due to the introduction of a proxy with appropriate re-scheduling mechanisms.

Interesting future work includes consideration of

- other load situations in the backbone, resulting in less or higher packet losses
- more realistic retransmission mechanisms on top of TFRC
- appropriate rebuffering strategies in case of limited client buffer size

- more realistic model of the wireless link
- radio access technologies other than WCDMA, e.g. GPRS.

6. REFERENCES

[1] D. Wu, Y. T. Hou, W. Zhu, Y. Zhang, and Z. M. Peha, "Streaming video over the internet: Approaches and directions," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, Mar. 2001.

[2] 3GPP, "PSS general description," *Technical Specification 26.233*, Mar. 2001.

[3] 3GPP, "PSS protocols and codecs," *Technical Specification 26.234*, Mar. 2001.

[4] I. Elsen, F. Hartung, U. Horn, M. Kampmann, and L. Peters, "Streaming technology in 3G mobile communication systems," *IEEE Computer*, vol. 34, no. 9, pp. 46–52, Sep. 2001.

[5] Schulzrinne, Casner, Frederick, and Jacobson, "RTP: A transport protocol for real-time applications," *Internet-Draft ietf-avt-rtp-new-01.txt (work inprogress)*, 1998.

[6] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Trans. on Networking*, vol. 7, no. 4, pp. 458–472, Aug. 1999.

[7] E. Dahlman et al., "WCDMA - the radio interface for future mobile multimedia communications," *IEEE Trans. on vehicular Technology*, vol. 47, no. 4, pp. 1105–118, Nov. 1998.

[8] 3GPP, "QoS concept and architecture," *Technical Specification 23.107 v4.2.0*, Oct. 2001.

[9] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. of the SIGCOMM*, Aug. 2000.

[10] M. Handley, J. Pahdye, S. Floyd, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol specification," Internet draft draft-ietf-tsvwg-tfrc-00.txt, Nov. 2000.

[11] W. Tan and A. Zakhor, "Real-time Internet video using error resilient scalable compression and tcp-friendly transport protocol," *IEEE Trans. on Multimedia*, vol. 1, no. 2, pp. 172–186, June 1999.

[12] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet," in *Proc. of the IEEE INFOCOM*, Mar. 1999, vol. 3, pp. 1337–1345.

[13] D. Sisalem and H. Schulzrinne, "The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme," in *Proc. of the NOSSDAV*, Jul. 1998.

[14] J. Bolot and A. Vega-Garcia, "Control mechanisms for packet audio in the internet," in *Proc. of the IEEE INFOCOM*, 1996, vol. 1, pp. 232–239.

[15] F. Toutain, "TCP-friendly point-to-point video-like source rate control," in *Proc. of the Packet Video Workshop*, May 1999.

[16] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," Tr98-008, UMASS CMPSCI Tech Report, Feb. 1998.

[17] M. Meyer, Joachim Sachs, and Markus Holzke, "Performance evaluation of a TCP proxy in WCDMA networks," in *Proc. of ACM MobiCom 2002 (submitted)*, Sep. 2002.

[18] W. Feng and S. Sechrest, "Smoothing and buffering for delivery of perrecorded compressed video," in *Proceedings of the IS&T/SPIE Symp. on Multimedia Comp. and Networking*, Feb. 1995, pp. 234–242.

[19] W. Feng and S. Sechrest, "Critical bandwidth allocation for delivery of compressed video," *Computer Communications*, vol. 18, pp. 709–717, Oct. 1995.

[20] W. Feng, F. Jahanian, and S. Sechrest, "Optimal buffering for the delivery of compressed prerecorded video," *Multimedia Systems Journal*, pp. 297–309, Sep. 1997.

[21] J. D. Salehi, Z.-L. Zhang, J. F. Kurose, and D. Towsley, "Supporting stored video: Reducing rate variability and end-to-end resource requirements through optimal smoothing," *IEEE/ACM Trans. on Networking*, vol. 6, pp. 397–410, Aug. 1998.

[22] J. M. McManus and K. W. Ross, "Video on demand over ATM: Constant-rate transmission and transport," *IEEE J. Select. Areas Commun.*, pp. 1087–1098, Aug. 1996.

[23] Z. Jiang and L. Kleinrock, "General optimal video smoothing algorithm," in *Proceedings of the IEEE INFOCOM*, Apr. 1998, pp. 676–684.

[24] W. Feng and J. Rexford, "A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video," in *Proc. IEEE INFOCOM*, Apr. 1997, pp. 58–66.

[25] S. Sen, J. Rexford, J. K. Dey, J. F. Kurose, and D. Towsley, "Online smoothing of variable-bit-rate streaming video," *IEEE Trans. on Multimedia*, vol. 2, no. 1, pp. 37–48, Mar. 2000.

[26] J. Rexford and D. Towsley, "Smoothing variable-bit-rate video in an internetwork," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 202–215, April 1999.