Adaptive scheduling of MPEG video frames during real-time wireless video streaming

Sergei Kozlov[1], Peter van der Stok[1,2], Johan Lukkien[1]
1. Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
2. Philips Research Laboratories, Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
{s.n.kozlov,p.d.v.v.d.stok,j.j.lukkien}@tue.nl

Abstract

Streaming video over wireless networks can be complicated by a highly varying link capacity. We present an approach that allows close-to-optimal utilization of the link bandwidth. We use a property of the medium access control (MAC) of the 802.11 standard to do instant detection of the bandwidth fluctuations by observing the transmission rate from the sending buffer. The specifics of a general MPEG stream allow classifying video-frames by their importance for the user-perceived quality. We build a scheduling mechanism that favors transmission of more important frames at the expense of the less important ones. We validate our approach by means of test-bed experiments and demonstrate a significant improvement of the end video quality.

1. Introduction

Streaming video over a wireless link is complicated by high fluctuations of the link capacity, especially in the presence of additional interferences (such as an active blue-tooth device or a microwave oven) as shown in Figure 1. Typically, there is a large gap between the worst-case and average-case bandwidth. The effect of bandwidth variations directly concerns the probability that packets are lost dependent on the transmission protocol that is used. With the use of TCP and the ability to stop the video at the source, the video will stop and start with infrequent intervals. When real-time broadcast is used, the video cannot be stopped or when RTP is used for transport of the video, packets will be lost.

A video frame typically is composed of 20 to 60 packets. The loss of one packet leads to the loss of a complete frame. Dependent of the type of frame, the loss leads to perceptible artifacts. Colored squares (blocking) appear in the video, different scenes overlap, videos may become completely unrecognizable, or the movement of objects in the video is irregular. This paper discusses an algorithm that greatly reduces or simply removes the artifacts that occur during video streaming over a wireless medium. A selective control of the packet loss at the sender is the key.

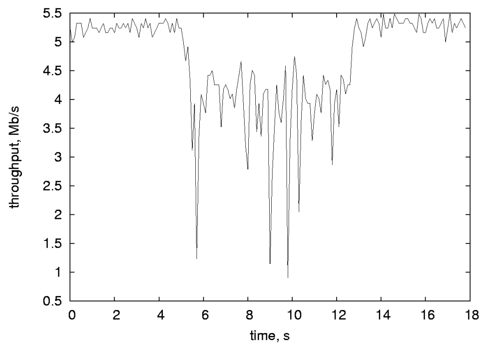

Figure 1. Wireless link effective throughput; additional interference (a microwave oven nearby) introduced for the interval from 5 to 15 seconds

The paper is organized as follows. Section 2 gives an overview of related work. Specifics of an MPEG stream related to our work are discussed in Section 3. In Section 4 we summarize the problems related to sending an MPEG stream over a wireless network and in Section 5 we describe the algorithm to minimize those problems. Section 6 provides measurements and an evaluation of the approach. Conclusions and future results are briefly discussed in Section 7.

2. Related work

The work of Lu and Christensen ([1]) which is an extension to the work of Joe ([2]) is most related to our solution presented in this paper. Both suggest a selective discarding scheme that implements a trade-off between the loss of more important MPEG “I” frames and less important “B” frames. Lu and Christensen ([1]) use simulation to prove that the effect of jitter on a loss-less channel is diminished by selectively removing frames at the receiver side. They focus on wired Ethernet, where limited bandwidth is not a big problem. Our work concentrates on rapid bandwidth variations occurring on wireless links. The algorithm selectively discards frames at the sender before they are sent, thus reducing the bandwidth needs in a controlled manner.

Gürses et al. ([3]) propose a streaming system architecture for stored video which consists of an input buffer at the server side coupled with the congestion control scheme of TCP such that TCP notifies the application about its readiness to transfer next video data. The proposed buffer management scheme selectively discards low priority frames from its head-end dependent on prediction of timeliness of the frame. This approach, however, is only suitable for streaming of stored video due to the usage of large buffers aiming to absorb the fluctuations in the TCP estimation of the available bandwidth.

3. Structure of MPEG video stream

An MPEG video stream consists of frames of different types: I frames are intra-coded frames which are coded independently, without any reference to other frames; P frames are predictive-coded frames which are coded with respect to previous I or P frame in the stream; B frames are bi-directionally predictive-coded frames which are coded with respect to both a previous I or P frame and a next I or P frame in a stream. Frames are combined in subsequent groups called *Group of Pictures (GOP)*, which consist of one I frame and several P and B frames.

Figure 2 shows a typical structure of a Group-of-Pictures (GOP) in a MPEG stream. The arrows show the dependencies between frames. For example, correct decoding of B_2 is not possible without presence of I_0 and P_3. Similarly, the whole GOP cannot be decoded properly if the corresponding I frame is not present.

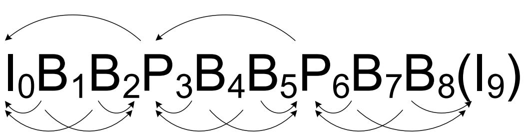

Figure 2. The dependencies between frames in a stream (frames in display order; frame I_9 belongs to the next GOP)

During the periods of low bandwidth, the losses of stream parts are randomly distributed, which makes the probability of losing parts of a given I or P frame higher than those of a given B frame, because I and P frames consist of more packets than B frames, as can be seen in Figure 3.

From the point of view of user-perceived quality, the loss of an I or P frame causes visible artifacts to appear also during rendering of the dependent frames (see Figure 4), while the loss of a B frame does not influence the quality of any other frames. This makes the loss of B frames preferable to the loss of P and I frames.

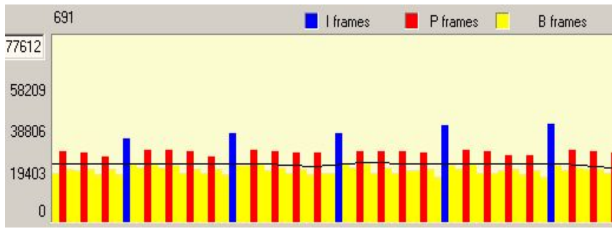

Figure 3. Frame sizes (in bytes) in a 5Mb/s MPEG stream. *Courtesy of Moonlight http://www.moonlight.co.il/)*

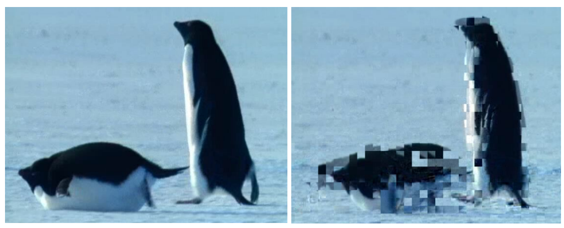

Figure 4. A B frame in the presence (on the left) and absence (on the right) of a reference frame.

In other words: if the bandwidth decreases below the bit-rate of the video stream, we would gain from selectively dropping B frames first, thus enormously reducing the number of blocking artifacts given the available throughput. As Figure 3 shows, the cumulative weight of B frames in this example MPEG stream is more than 50%. This means that the probability that a lost packet belongs to a B-frame is about 50%. More importantly, this means that by only dropping B frames we can make the resulting example video stream fit into a bandwidth that is half the bit-rate of the stream and still see no artifacts at the receiver. The price for this will be a decreased frame-rate, which in the extreme case (i.e. all B frames are dropped) will be 1/3 of the original in the shown example. In general this will depend on the structure of the GOP.

4. Wireless transmission of a MPEG stream

Figure 5 shows a simplified sender-receiver communication scheme. In our work we concentrate on the link layer. We assume that there are no intermediate devices on the way from the video source to the video sink, leaving the problem of data loss caused by congestion in intermediate nodes out of this paper's scope. We focus on the influence of the behavior of the wireless link on the performance of the system.

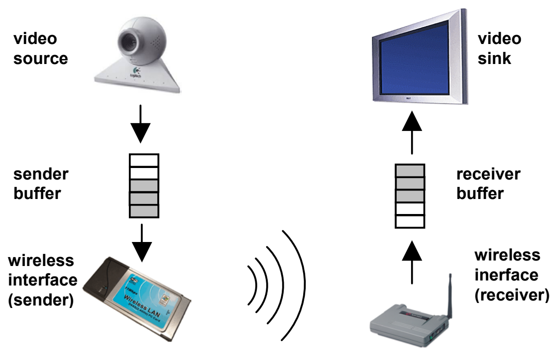

Figure 5. Simplified wireless communication.

We in particular investigate the IEEE 802.11 standard. The IEEE 802.11 standard has a built-in mechanism of retransmissions (MAC-level retransmissions, [8]), which hides high losses occurring on the physical layer from the higher layers. In fact, transport protocols do not see most of the losses happening at the physical layer because of this mechanism – instead, they see it as a link with a fluctuating throughput. Only when the number of MAC-level retransmissions reaches a predefined limit (set up in the wireless interface driver), data is considered to be lost and there are no further attempts to retransmit the given packet in the link layer. This fact is then reported to the upper layers, leaving it up to transport protocols to decide if a retransmission should take place there (as an example, TCP [6] would retransmit and UDP [7] would not). In this work we use the Real-time Transport Protocol (RTP) designed for video streaming over Internet ([9]) which is based on UDP and implements no retransmission mechanism.

Transmitted data can be lost in 3 ways:

1. *During physical transmission.* If the interference of the RF-carrier increases such that the MAC-level retransmission limit is reached, the data is discarded and the loss is reported to higher layers.
2. *At the sender.* If the video source is producing data with a higher bit-rate than the throughput of the wireless link, the sender buffer overflows.
3. *At the receiver.* This can happen due to two reasons: 1) if the video sink is consuming data with a lower bit-rate than the sender is sending, the receiver buffer overflows; 2) if the buffer at the sender is larger than at the receiver and the transmission bandwidth is temporarily much higher than the consumption rate at the receiver, then the data accumulated in the sender buffer may flood the receiver buffer later and cause overflow loss.

We do not consider the third case, assuming that 1) the computational power of the receiving device is always sufficient to process all the data arriving from the network; 2) the size of the sender buffer is less-than-or-equal to the size of the receiver buffer. We can minimize the probability of the first case by increasing the retransmission limit to its maximum (usually 255). As work in [12] shows, reaching this limit would mean unsuccessful attempts during approximately 5.67 seconds, which we may consider as a broken link. Thus we assume that the retransmission is eventually successful.

However, long attempts to retransmit can result in an overflow of the sender buffer (case 2). Increasing the buffer decreases this probability (at least if the average video bit-rate stays lower than the average available bandwidth), but this will come at the cost of an increased end-to-end latency of the video and an increase in the costs of the sender and receiver devices.

We focus on the 2nd case – losses at the sender – because precisely the overflow of the sender buffer would indicate that the data is being produced at a higher bit-rate than it can momentary be delivered via the wireless link. In other words, this would be the indication of insufficient bandwidth (further in this section we only deal with the sender buffer simply referring to it as "buffer").

The usual buffer management technique is Tail Drop, in which the new data gets dropped when the buffer is full. In order to clearly explain the dynamics of the buffer's occupancy, we take an example of an MPEG stream with the characteristics shown in Table 1.

Network bandwidth	5Mbps
Video stream bit-rate	4Mbps
Video frame rate	25fps
GOP structure	IBBPBBPBBPBBPBB
I-frame size	31KB
P-frame size	20KB
B-frame size	14KB

Table 1. A model of a MPEG stream: fixed sizes for each type of frame.

Thus, we assume that a new frame is delivered to the sending buffer every 40ms and that frames of the same type have the same fixed size. We choose the buffer size equal to 33 Kbytes such that the buffer is at least sufficient to accommodate a complete I frame. Figure 6 depicts the dynamics of the buffer for the first few frames for the numbers shown in Table 1.

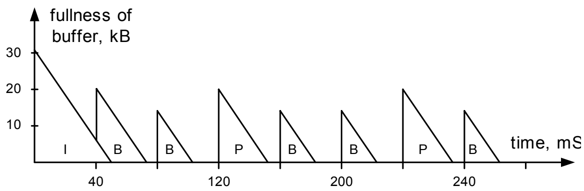

Figure 6. Occupancy of the buffer; average video bit-rate is lower than the average bandwidth

The vertical axis denotes the number of non-transmitted bytes in the sender buffer. The horizontal axis represents time. The points where the diagonal lines cross the time axis are the moments when a corresponding frame has arrived at the sender (note that at time 40, the B frame is delivered to the buffer while the previous I-frame has not been fully transmitted). We see that the occupancy of the sender buffer does not grow above 31KB and that all the frames are delivered to the receiver.

Now let us consider how the situation changes if after 50ms the link throughput drops to 1.6Mbps and after 200ms it comes back to 5Mbps (Figure 7).

We see that after 50 ms the occupancy of the buffer starts growing with every new coming frame and after 120ms tails of frames start being lost because the amount of offered but non-transmitted bytes is higher than the buffer capacity. This situation continues until the link throughput returns to the original value that is sufficient to carry the stream.

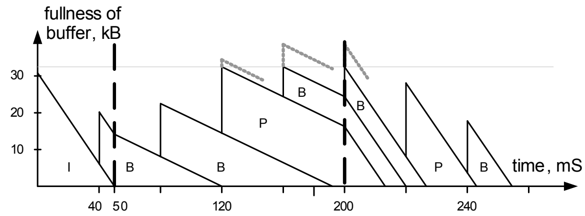

Figure 7. From 50ms to 200ms the throughput of the link drops to 1.6Mbps.

The chosen buffer size of 33 Kbytes (taken for illustration purposes) does not break the generality of the problem, originating in too small buffers to cover the bandwidth variations (see [1]), which actually means that at some point of time there will inevitably be losses as just illustrated.

Generally, MPEG decoders (also the one we use in our testing setup) do not handle incomplete frames, so they are normally thrown away at the receiver. Hence, when all tails of the video frames are dropped due to overflow of the sender buffer, no video will be displayed to the end user. This can be improved by a strategy to drop the frames which would not completely fit into the buffer beforehand. Figure 8 depicts this strategy for the case of the same stream transmitted over a 2.5Mbps link.

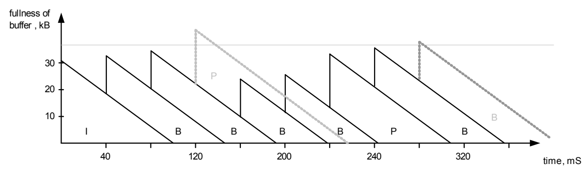

Figure 8. Link throughput of 2.5Mbps; frames that do not fit into the buffer are dropped.

This strategy allows transmitting complete frames which is made possible by not wasting the bandwidth for the frames which would not get transmitted completely anyway.

However, dependencies between video frames introduce an additional problem described in the previous section. In Figure 8, the loss of the P frame causes the rest of the GOP to be displayed in a distorted fashion. We illustrate the impact of random frame skipping by the following example. Let us consider that due to the decreased bandwidth we face a drop of every seventh frame. In Figure 9 we mark the frames that are lost with a double strikethrough, and the ones that are displayed with artifacts due to the absence of the dependent frames with a single strikethrough.

Figure 9. Impact of random frame losses.

We see that out of 45 video frames we have actually lost only 6, though 22 more are being displayed with artifacts, which are caused by the losses of P frames.

A significant improvement to this result could be made if we would favor the transmission of P frames at the cost of previous B frames (see Figure 10).

Figure 10. Favoring transmission of P frames at the cost of previous B frames.

In the next section we introduce a simple scheduling scheme that makes a selection of dropped video-frames based on their importance for the consistency of the GOP.

5. Scheduling algorithm

To reduce delays and reduce costs we try to minimize the number of buffers. In accordance with the current practice in Consumer Electronic devices, a maximum of two frames can be buffered at the sender: the frame which is currently being transmitted, *S*, and the frame which is waiting for transmission, *W* – see Figure 11

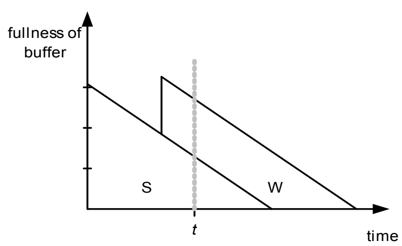

Figure 11. At time *t* frame *S* (partially transmitted) and *W* are simultaneously in the buffer

The frame which is offered for transmission by the application will be marked as *C*. When *W* is present, which means that we cannot buffer any more frames, and *C* is arriving from the application, the scheduling algorithm decides which of the two frames (C and W) is the least important one for the end-quality in order to decide which one to discard. For efficiency reasons, we decided to continue the transmission of the S frame.

The following algorithm favors transmission of I and P frames over B frames:

```
1 PROCESS buffer_enqueue
2 BEGIN
3   WHILE (TRUE) DO
4       WHILE (C is empty) DO
5           Nothing
6       IF (W is empty) THEN
7           Store C in W
8       ELSE
9           IF (C is of type I) THEN
10              Overwrite W with C
11          ELSE IF (C is of type B)
12              Discard C
13          ELSE IF (W is of type I or P)
14              Discard C
15          ELSE
16              Overwrite W with C
17 END
```

We will demonstrate the effect of the algorithm with several examples.

In Figure 12 we see that at 80 ms an incoming B frame is dropped due to the fact that previous B and I frames are occupying the buffer. This action corresponds to line 12 of the algorithm. The same thing happens at 200ms.

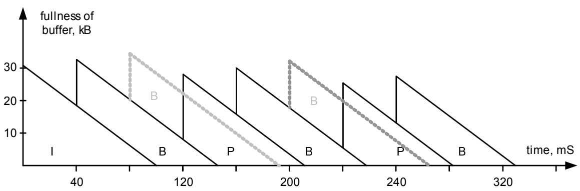

Figure 12. Effect of the scheduling algorithm, example 1 (throughput 2.5Mbps)

For another example we chose the network bandwidth to be around 1,6Mps (which is 2,5 times lower than the bit-rate of the stream). The result is shown in Figure 13.

At 80 and 200ms an incoming B frame is dropped, as before. At 120ms and 220ms the B-frame in W is overwritten, which corresponds to line 16 of the algorithm, where we choose to remove the waiting B frame in order to store the incoming P frame. As the final result, we lose all B frames from the stream, thus assuring the delivery of all complete I and P frames.

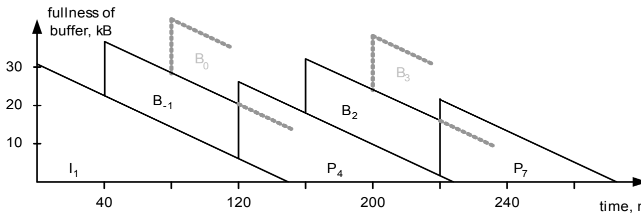

Figure 13. Effect of the scheduling algorithm (same stream, bandwidth is around 1,6 Mbps)

6. Testing and evaluation

6.1. Testing setup

1) Sender. We used a Dell Inspiron 4150 laptop running Debian Linux, kernel 2.4.22. We implemented the scheduling algorithm in the framework of LARTF ([4]). For setting up a video stream over the network we used streaming software called GStreamer ([5]).

2) Receiver. We used a Linksys 802.11b wireless router connected to the MT3 set-top box running receiving software built with Philips proprietary libraries (Ludite-II) for RTP video decoding.

6.2. Measurements

We created several 1-minute video-samples of the same scene, encoded with different bit-rates. The frame-rate is 25 frames/sec. After 20 seconds we switched on a microwave oven located near the receiver and sender for 20 seconds. Accordingly, approximately 500 frames were transmitted while strong interference was present. The results are shown in Figures 14, 15, 16 and 17. The horizontal axis shows the progress of time expressed as the frame number. The loss of B (I,P) frames is denoted with a cross at the moment of loss with value B (I,P) frame on the vertical axis.

Figure 14 shows that a 3Mbps video doesn't suffer from the additional interference. Consequently, the bandwidth of the wireless link doesn't drop below 3Mbps even while the microwave oven is on (which is in accordance with the picture shown in the introduction). Only three B-frames were dropped during this test.

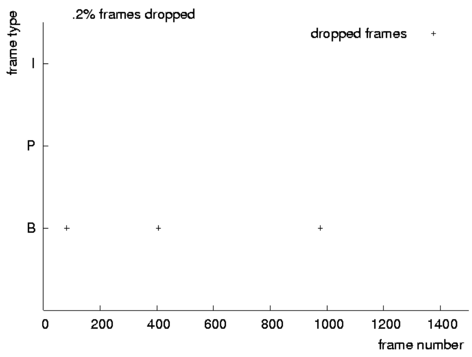

Figure 14. 3Mbps video is delivered almost undisturbed.

Figure 15 shows the results obtained with a 4Mbps stream. We see that between frame numbers 500 and 1000 we have an increased loss of B frames. Still, no P frames are lost, which provides 0% of disturbed frames.

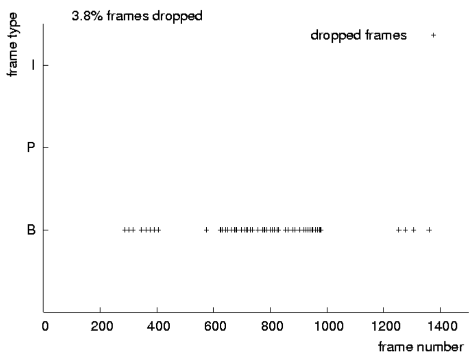

Figure 15. 4Mbps transmission is more sensitive to additional interference.

Figure 16 shows the situation when the link bandwidth is constantly lower than the stream bit-rate (5Mbps). We see an increased B frame loss during the activity of the microwave oven. Still no P frames are lost.

Finally, we try to stream a 8Mbps stream, which is significantly beyond the capacity of the 802.11b link (Figure 17). Still we do not see I or P frames dropping when the microwave is off, which fits into our conclusion made in Section 3 (throwing away B frames gives us a stream size reduction of more than

50%, which turns the 8Mbps stream into a stream of less than 4Mbps, which fits into the link capacity). But the interference brought by a microwave oven decreases the throughput of the link, hence we start losing P frames and even some I frames, which results into the artifacts shown in Figure 4.

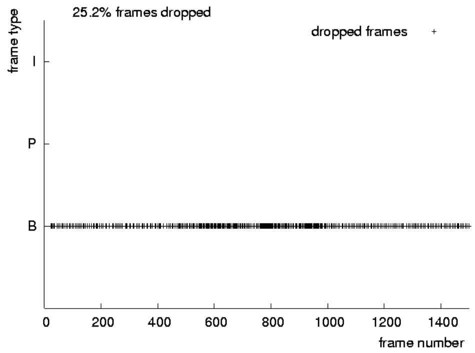

Figure 16. Periodic losses of B-frames in the case when the stream bit-rate (5Mbps) is higher than the bandwidth.

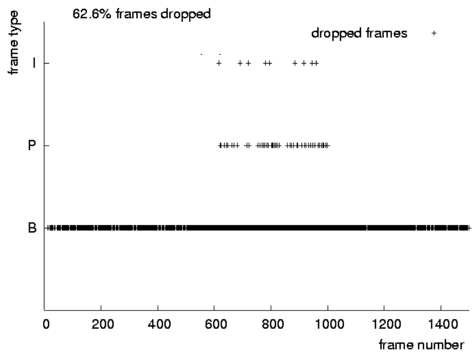

Figure 17. 8Mbps stream; during the additional interferences P and I frames start getting lost.

7. Conclusions and future work

In this paper we proposed and tested a simple and effective solution for streaming MPEG video over wireless links using a kind of temporal scalability . We have shown its efficiency in real life tests. This approach provides minimum user-perceived quality losses when the network link capacity drops below the video-stream bit-rate.

The advantages of this approach are:

- Only sender modifications are needed. Most of the widely used MPEG decoders will handle the resulting stream. The only requirement for the decoder is the ability to recover from lost frames (which is common for decoders used in an RTP streaming system)
- It is very responsive to network variations (<40 ms) due to the fact that the scheduling is done at the link layer.
- The available bandwidth is used to a maximum.

Our future work includes:

- combining this approach with other scalable techniques – for example the one described in [11], or in combination with dynamically tunable trans-coders
- streaming two and more video streams from the same sender
- modification of the algorithm when more buffering takes place (for example, preserving P frames from being dropped by the scheduler).
- providing dynamic adaptation of the scheduling scheme according to long-term bandwidth variations
- combining this technique with traffic shaping approaches

References

[1] Yantian Lu, Kenneth J. Christensen. Using selective discard to improve real-time video quality on an ethernet local area network. *Int. Journal of Network Management* 1999, pp. 106-117

[2] Joe I. Packet loss and jitter control for real-time MPEG video communications. *Computer Communications* 1996, pp. 901-914

[3] Eren Gürses, Gözde Bozdaği Akar, Nail Akar, A Simple and Effective Mechanism for Stored Video Streaming with TCP Transport and Server-Side Adaptive Frame Discard, submitted to *Elsevier Science* 2003

[4] Linux Advanced Routing & Traffic Control. http://lartc.org/

[5] GStreamer, Open Source Multimedia Framework. http://gstreamer.freedesktop.org/

[6] Transmission control protocol. RFC 793

[7] User Datagram Protocol. RFC 768

[8] IEEE 802.11 Standard. http://grouper.ieee.org/groups/802/11/main.html

[9] A Transport Protocol for Real-Time Applications. RFC1889

[10] Ralf Steinmetz, Klara Nahrstedt. *Multimedia: computing, communications, and applications.* ISBN 0-13-324435-0, pp 151-156

[11] D. Jarnikov, Towards Balancing Network and Terminal Resources to Improve Video Quality, *Eindhoven University of Technology*, SAI Technical Report, August, 2003 (ISBN 90-444-0317-6)

[12] F.H.G. Ogg, P.D.V. van der Stok, M.A. van Hartskamp, Smoother Streaming over Wireless Networks, *Nat.Lab. Technical Note PR-TN-2003/00545*