

Adaptive Streaming of Combined Audio/Video Content over Wireless Networks

Jeffrey Kang, Marek Burza, Peter van der Stok

Philips Research, High Tech Campus 31, 5656 AE Eindhoven, The Netherlands
{Jeffrey.Kang, Marek.Burza, Peter.van.der.Stok}@philips.com

Abstract. This paper describes a method for robust streaming of combined MPEG-2 audio/video content over in-home wireless networks. We make use of currently used content distribution formats and network protocols. The transmitted bit-rate is constantly adapted to the available network bandwidth, such that audio and video artifacts caused by packet loss are avoided. Bit-rate adaptation is achieved by using a packet scheduling technique called I-Frame Delay (IFD), which performs priority-based frame dropping upon insufficient bandwidth. We show an implementation using RTP and an implementation using TCP. Measurements on a real-life demonstrator set-up demonstrate the effectiveness of our approach.

1 Introduction

In-home networks are becoming more and more common in consumer households. They connect together the different electronic devices in the home from the CE, mobile and PC domains, with which different digital content is stored and viewed. Such in-home networks typically consist of wired and wireless segments. Especially wireless networks are popular due to the ease of installation. However, wireless networks cannot always deliver the bandwidth needed for transporting the desired content. This is because wireless networks offer a lower bandwidth than wired networks, and very often this bandwidth has to be shared between multiple streams. Furthermore, the wireless network bandwidth often exhibits an unpredictable and fluctuating behavior. This is mainly caused by reflections and obstacles, roaming portable devices, and disturbances from other devices (e.g. neighboring wireless networks, microwaves). Other network technologies, e.g. power-line, also exhibit similar stochastic behavior under interference. For streams with real-time requirements, such as audio/video (A/V), insufficient network bandwidth causes data loss and the associated artifacts, such as blocky distortions in the image and disturbances in the audio. This is clearly unacceptable for the end user.

The problem of bandwidth sharing can be addressed by several QoS (Quality-of-Service) techniques such as prioritization and bandwidth reservation and access control mechanisms. In this paper, we address the second problem, namely that of fluctuating bandwidth for audio/video streaming. A technique is presented to stream the A/V content over the network, whereby the amount of data transmitted by the sender is dynamically adapted to the available bandwidth by selectively dropping data. In this way the perceived quality of the A/V stream is dynamically adjusted according to the quality

of the network link. Our solution can be implemented using RTP (Real-time Transport Protocol) and TCP (Transmission Control Protocol).

This paper is organized as follows. Section 2 lists some related work. Section 3 describes our adaptive streaming technique and its implementations. Experimental results are presented in Section 4. This paper ends with conclusions and final remarks.

2 Related work

Most work in literature concentrates around streaming of video only, whereas we also consider audio, which is streamed together with the video stream. Most users are more sensitive to audio than to video artifacts, therefore these should be avoided. Different approaches have been proposed for adaptively streaming video over wireless networks. These solutions can be categorized by:

- *Scalable video*. The original video is encoded in different layers, where a *base layer* contains video of acceptable quality, and one or more *enhancement layers* enhance the quality of the base layer. Adaptation is done by dropping enhancement layers in case of insufficient bandwidth. As long as the base layer can be transmitted without loss, no artifacts will occur. Examples of this approach can be found in [1][2][3].
- *Transcoding/transrating*. Here the original bit-rate of the video is adaptively changed to a different bit-rate depending on the available bandwidth, e.g. by dynamically changing the quantization parameters. Examples of such solutions are found in [4][5].
- *Frame dropping*. Complete frames are dropped in case of insufficient bandwidth, examples are found in [6][7].

Our adaptation technique falls into the last category and is called *I-Frame Delay* (IFD), previously reported in [8]. This paper extends this by including audio. IFD is a scheduling technique which drops MPEG video frames when detecting insufficient bandwidth, thereby favoring I- and P-frames over B-frames. Artifacts are avoided as long as only B-frames are dropped. The result is a decreased video frame rate. The perceived quality here is lower than e.g. scalable video [9] or transrating, however IFD has the lowest cost in the implementation, and is able to react quickly to the changing conditions. Most MPEG decoder implementations are able to decode the resulting stream. Furthermore, the decision to perform adaptation is based on send buffer fullness, instead of relying on feedback from the receiver for bandwidth estimation. Therefore no receiver modifications are necessary. The work of [10] resembles ours the most, dealing with adaptive streaming of MPEG-2 Transport Streams by means of frame dropping. However, this approach relies on specific receiver feedback about the network status. Furthermore, since the feedback arrives periodically, the beginning of an abrupt bandwidth drop can often not be handled in time, leading to a short burst of lost packets (and hence frames). Lastly, only RTP streaming is considered, whereas we also consider TCP.

3 Adaptive streaming of MPEG-2 Transport Streams

In this paper, we focus on the home scenario where content is entering the home from a content provider via ADSL, cable or satellite, and possibly stored on a home storage

server. Here the most common content format is MPEG-2 Transport Streams (TS) [11]. Therefore we will further discuss adaptive streaming of MPEG-2 TS. Section 3.1 describes our IFD adaptive video scheduling technique. Sections 3.2 and 3.3 explain how this technique can be applied on Transport Streams using RTP and TCP.

3.1 I-Frame Delay

I-Frame Delay (IFD) is a scheduling technique for adaptive streaming of MPEG video. The basic idea is that scheduler will drop video frames when the transmission buffer is full and overflow is imminent due to insufficient bandwidth, to reduce the transmitted bit-rate. The algorithm is characterized by the following: 1) buffer fullness is indicated by the *number of frames* currently in the buffer (not the number of bytes), 2) less important frames (B-frames) are dropped in favor of more important frames (I- and P-frames), 3) the transmission of I-frames is delayed when conditions are bad but as little as possible omitted even if they are out-of-date w.r.t. the display time, because they can still be used to decode subsequent inter-predicted frames¹.

The IFD mechanism is based on two parts: 1) during parsing and packetization of the stream into network packets, the stream is analyzed and the packets are *tagged* with a priority number reflecting the frame type (I/P/B)², and 2) during transmission, packets are *dropped* by the IFD scheduler when the bandwidth is insufficient. Non-video packets are given a priority number not recognizable by the IFD scheduler, which therefore will never drop them. The size of the IFD buffer should be big enough to hold a number of frames. The minimum required size is two frames, one to hold the frame currently being sent (denoted as S), and one currently waiting to be sent (denoted as W). Increasing the IFD buffer size permits more intelligent decisions on which frames to be dropped, at the cost of increased latency and memory usage. Figure 1 depicts an example of the buffer filling. The numbers represent the priority numbers of the packets. In this example the IFD-related priority numbers are 10 and higher. The packets with priority number 12 belong to the S frame, and the packets with number 11 belong to the W frame. Currently a packet from an incoming frame C is about to enter the buffer. As can be seen, it is possible to interleave video packets with non-video packets (priority numbers 0 and 2). When a packet belonging to C tries to enter the IFD buffer and both

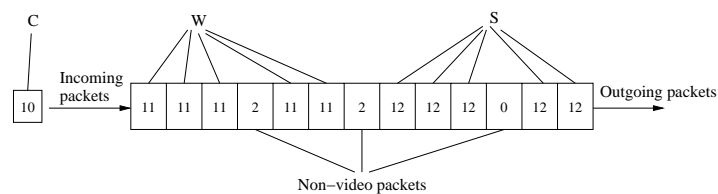


Fig. 1. Network packets in the IFD buffer.

¹ We assume that out-of-date frames are still decoded and only thrown away by the renderer, rather than thrown away already before decoding.

² Since there may be multiple consecutive B-frames between two P-frames, we use different tags to distinguish between them.

S and W are full, the IFD scheduler will decide to drop a frame (in C or W), based on the priority numbers. No artifacts will occur if only B-frames are dropped, because no subsequent frames depend on them. When the network conditions are such that also P-frames are dropped, then the GOP (Group of Pictures) is said to be *disturbed* and the remainder of the GOP is also dropped. This causes the effect of the image being temporarily frozen, the duration of which depends on the GOP size. For an IFD buffer capable of holding two frames, the frame dropping algorithm is shown below.

```

1: procedure BUFFER_ENQUEUE( $C$ )
2:   if Disturbed_GOP == True then
3:     if  $C$  is of type I then                                     ▷ New GOP is encountered
4:       Discard  $C$                                              ▷ Reset disturbed GOP flag
5:     end if
6:   end if
7:   if Disturbed_GOP == True then
8:     Discard  $C$                                                ▷ Discard rest of disturbed GOP
9:   return
10:  end if
11:  if  $W$  is empty then
12:    Store  $C$  in  $W$ 
13:  else
14:    if  $C$  is of type I then
15:      Overwrite  $W$  with  $C$ 
16:    else if  $C$  is of type B then
17:      Discard  $C$ 
18:    else if  $W$  is of type I or P then
19:      Discard  $C$ 
20:      if  $C$  was of type P then                                  ▷ Discarded frame is P-frame
21:        Disturbed_GOP = True                                  ▷ Set disturbed GOP flag
22:      end if
23:    else
24:      Overwrite  $W$  with  $C$ 
25:    end if
26:  end if
27: end procedure

```

3.2 RTP implementation

This section describes our adaptive streaming approach using RTP. We first present a method for encapsulating TS packets into RTP packets, such that they can be used by IFD. This does not come without consequences, as will be explained. We then describe our streamer implementation.

IFD-friendly RTP encapsulation. For RTP encapsulation of TS packets we adhere to RFC2250 [12]. It states that a RTP packet contains an integral number of TS packets (188 bytes each). Each RTP packet does not need to contain an equal number of TS packets, however the common approach is to encapsulate 7 TS packets into one RTP packet. Together with some protocol headers, the packet size stays just below the MTU (Maximum Transmission Unit) of 1500 bytes (Ethernet), which avoids fragmentation. The problem of such RTP packets is that they are dropped by IFD as a whole, and they may contain audio, video and data TS packets, or packets belonging to multiple frames. Therefore we need to make sure that video and non-video packets are split up in different RTP packets, and the same goes for video packets belonging to different frames. Re-ordering of TS packets is unwanted due to timing dependencies and efficiency (e.g. buffer memory usage) considerations, therefore we can achieve the splitting by simply

finalizing packets earlier. During parsing, TS packets are gathered into RTP packets. A RTP packet is finalized if one of the following conditions is met:

1. The RTP packet contains 7 TS packets.
2. The next TS packet has a different TS PID (Packet Identifier) than its predecessor, and one of them has the video PID.
3. The next TS packet and its predecessor both have the video PID, but the next packet starts a new video frame.

Figure 2(a) shows the result of the common encapsulation scheme for an example TS packet sequence. The TS packets are denoted with their type (**A**udio, **V**ideo, **D**ata). For video packets also their frame numbers are mentioned. All packet boundaries are exactly 7 TS packets apart. Figure 2(b) shows the packet boundaries obtained with our encapsulation scheme. Boundary number 2 is the result of the first rule (maximum of 7 packets reached). Boundaries 1, 3, 4, 6 and 7 are the result of rule 2 (switching between video and non-video). Boundary 5 is the result of crossing a frame boundary (rule 3).

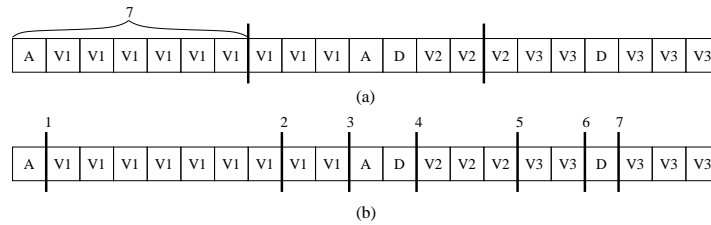


Fig. 2. Example of RTP packet encapsulation: (a) common scheme, (b) IFD-friendly scheme.

Compared to the common encapsulation scheme, parsing of the TS packets down to the video elementary stream level is required for finding the picture start codes in video packets to identify the start of each frame and the frame type. In many cases the start of a new frame is aligned with the start of a TS packet. This is the ideal situation and a clean separator can be placed between two TS packets to separate two frames. However, there are cases where the frame start and TS packet start are not aligned:

1. In some cases, the picture start code is located halfway a TS packet, which means that this packet belongs to two different frames. In this case there are a number of options for setting the RTP packet boundary: 1) split up this packet into two TS packets and add stuffing, 2) keep this packet with the previous frame, 3) keep this packet with the new frame. For efficiency reasons (no increased bit-rate and no copying/modification of TS headers) we did not go for option 1. Options 2 and 3 differ in which part of a frame will be dropped. With option 2 the start of the new frame will be lost if the previous frame is dropped, while with option 3 the last part of the previous frame will be lost if the new frame is dropped. In our experience, most decoders handle the latter situation better, therefore we choose option 3.
2. Sometimes the picture start code is split over two TS video packets. Detecting this situation entails a higher complexity, because two video packets have to be scanned at a time, with the additional possibility that they are interleaved with audio and data

packets. Failing to detect this situation may result in two frames being tagged as if they are one. If a B-frame is followed by a P-frame, then also the latter is tagged as a B-frame, and may be dropped as well.

The occurrence of the above situations depends on how the video stream is packetized into Packetized Elementary Stream (PES) packets, before it is multiplexed in the Transport Stream together with audio data. From our experience with broadcast streams two methods are most often used: 1) each PES packet contains a single frame, 2) each PES packet contains a GOP. The above miss-alignment situations do not occur when a PES packet contains a frame, because the start of a PES packet is always at the start of a TS packet. They might occur often, though, when each PES packet contains a GOP.

IFD overhead. As can be seen from Figure 2, our IFD-friendly encapsulation scheme results in more and smaller RTP packets. Smaller packets have more overhead, which means that the wireless network is used less efficiently. Considering an average size of a RTP packet consisting of $N_{TS_per_RTP}$ TS packets, we define the *encapsulation efficiency* E_{encap} as a measure to express how well a stream can be encapsulated, that is, how close $N_{TS_per_RTP}$ is to the optimal number of 7:

$$E_{encap} = \frac{N_{TS_per_RTP}}{7} \times 100\% \quad (1)$$

Our experience with typical Transport Streams yields values for E_{encap} ranging from 37.4% to 85.8%. Figure 3 shows the distribution of RTP packet sizes for example sequences with a high and a low encapsulation efficiency.

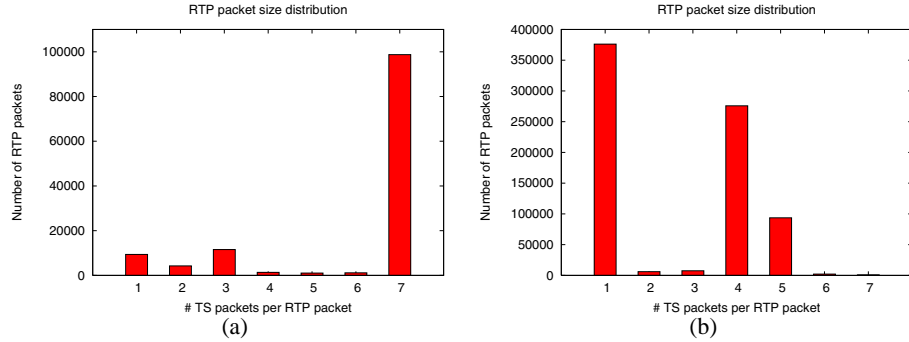


Fig. 3. Example RTP packet size distribution for high (a) and low (b) encapsulation efficiency.

One source of overhead of having smaller packets is the protocol headers (e.g. RTP, IP), as the packet payload is relatively smaller. An analysis of the header overhead resulted in numbers ranging from 0.92% to 8.42% for our example streams. Another source of overhead comes from the fact that the maximum theoretical throughput at the 802.11 MAC layer degrades with decreasing packet size (a theoretical analysis can be

found in [13]). Figure 4 depicts the relation between $N_{TS_per_RTP}$ and the maximum throughput. We define the throughput penalty as the relative drop in throughput with an average RTP packet size $N_{TS_per_RTP}$ compared to the throughput with the maximum packet size of 7 ($E_{encap} = 100\%$). For the example sequences we used, a throughput penalty was found ranging from 6.1% to 47%. It is therefore imperative to obtain a high encapsulation efficiency. This depends on how the packets have been multiplexed. The best is to have as few transitions between video and non-video packets as possible. Since video packets make up most of the stream, we can then achieve the highest possible number of RTP packets of the maximum size (such as in Figure 3(a)).

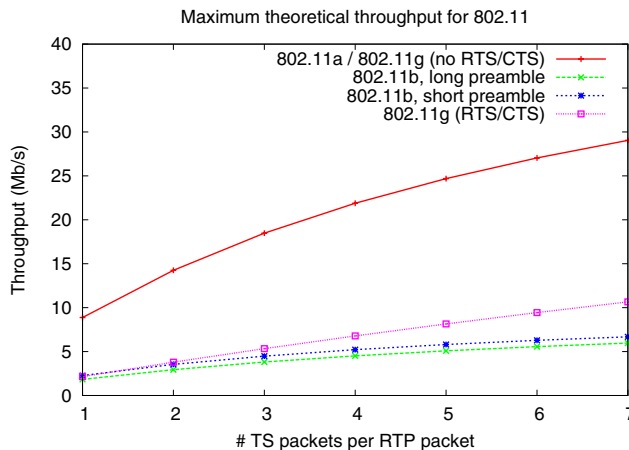


Fig. 4. Maximum theoretical throughput for 802.11.

Implementation. We implemented IFD as part of a RTP/RTSP server application running on a Linux PC, the streaming part of which is shown in Figure 5. The *TS file reader* reads TS packets from a MPEG-2 Transport Stream file and sends it to the *RTP packetizer*, which encapsulates the TS packets into RTP packets according to the method described earlier. The target transmission times (TTT) of the packets are determined by the file reader from the PCR (Program Clock Reference) values (following the DLNA guidelines [14]). The RTP packetizer then translates the TTT to RTP timestamps and inserts the proper RTP headers. The resulting RTP packets are given to the *RTP sender*. This component is responsible for buffering the RTP packets for transmission. For the right pacing of the transmission, the *RTP scheduler* examines the RTP timestamps of the RTP packets and tells the RTP sender to send the packets at the right time. IFD packet tagging is done by the RTP packetizer, and the dropping is implemented in the RTP sender. Audio and data packets are tagged with a priority number which is not recognized as video frames by IFD; they thus will never be dropped. In this way the audio is almost never interrupted at the receiver output.

3.3 TCP implementation

The main drawback of using RTP streaming is that IFD must be supported by each wireless node (e.g. an access point) in the network, not only at the sender. This is because

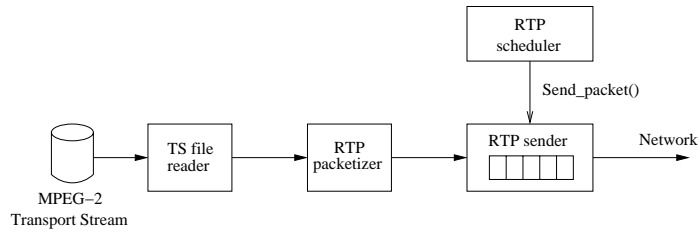


Fig. 5. RTP sender architecture.

with RTP, only the bandwidth of the wireless link on which transmission takes place (e.g. between the sender and the access point) can be monitored (there is no feedback from the receiving peer). If the path to the receiver passes another wireless link (e.g. from access point to the receiver), then the quality of the second link is visible for the access point, but not for the sender. Therefore the sender can only adapt to the bandwidth conditions on the first link but cannot prevent artifacts caused by packet losses on the second link, and therefore also the access point should implement IFD. This is commercially highly unattractive. TCP provides reliable data transmission using its acknowledge and retransmission mechanism. Since the acknowledgments are sent between the receiver and sender, it is possible to observe the end-to-end bandwidth on the network path, independent of the (number of) wired/wireless links on that path. Thus it suffices that only the sender supports IFD.

The main issue of TCP is that the retransmission mechanism may delay the streaming such that the real-time requirements are not met. We can solve this by applying IFD. Our TCP sender architecture is shown in Figure 6. The file reader component is similar to the RTP implementation. The TS collector component packs TS packets together in a similar way as the RTP packetizer described in Section 3.2, with a slight difference that the TS collector never applies encapsulation rule 1. This is because packets offered to TCP may be bigger than 7 TS packets; TCP will automatically split up such big packets into smaller chunks. The IFD algorithm is implemented by the *IFD dropper* component. It writes packets into the IFD buffer according to the pacing of the stream. The *TCP sender* task tries to send the packets in this buffer as fast as possible. When the network conditions deteriorate, TCP stalls and the TCP sender cannot empty the IFD buffer fast enough. This will trigger the IFD dropper to apply the dropping algorithm.

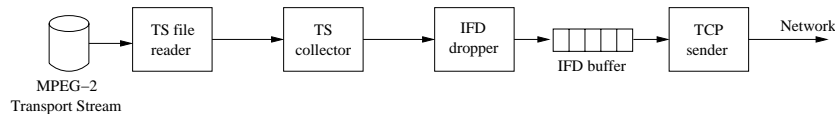


Fig. 6. TCP sender architecture.

The overhead introduced by our packetization scheme for TCP is smaller than for RTP. This is because TCP will automatically split up big packets into smaller chunks, and merge smaller packets into bigger chunks for transmission. The average packet size at the IP level is therefore bigger than for RTP. This comes at a cost of a bigger receiver buffer to deal with the higher jitter.

4 Experimental results

This section presents some experimental results in two scenarios. The first one involves a stationary receiver attached to a TV. In this scenario (Section 4.1), the wireless network is disturbed by turning on a microwave. The second scenario (Section 4.2) involves a mobile receiver, which is moved away from the access point, causing the bandwidth to drop. We used a 802.11g wireless access point and PCMCIA wireless adapters. IFD is implemented in the sender. The experiments were conducted in an office environment with a WLAN infrastructure, which causes some additional interference with our own network.

4.1 Stationary receiver with microwave disturbance

The set-up for this scenario is shown in Figure 7. The sender PC has a wireless connection to the access point, which is connected via wired Ethernet to a set-top-box. The nearby (at appr. 2 m) microwave introduces the disturbance. For this experiment we used a 10 Mb/s sequence. The GOP size is 15, the GOP structure is IBBP. During the

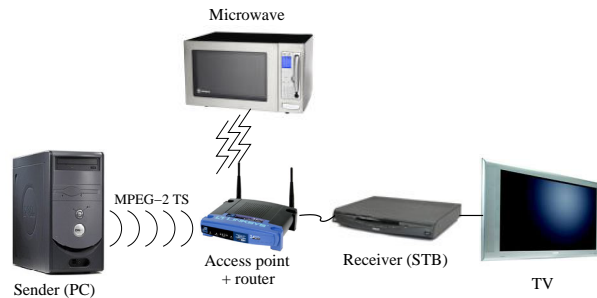


Fig. 7. Experimental set-up with stationary receiver.

experiment, the microwave was turned on 45 seconds after the stream started, for a period of 1 minute, and the output video and audio quality was observed on the TV. With the microwave off, the audio and video were streamed without problems. During the period when the microwave was on, the experiment without IFD exhibited artifacts in the video and frequent interruptions in the audio. With IFD turned on, the audio was never interrupted, and the video frame rate was observably reduced, however no artifacts were seen. Similar observations were made both for the RTP and the TCP implementation. Since the wireless network conditions may vary over time, the experiments were repeated a number of times.

Measurements were performed to determine the dropped frames and their types. We first ran an experiment with the TCP sender (Figure 6) without IFD, where the IFD dropper was configured to drop all incoming packets when the IFD buffer is full (tail-drop). The dropped frames at the sender side are shown in Figure 8(a). As can be seen, almost no frames were dropped when the microwave was off. When the microwave was on, frames from all types were dropped (including I- and P-frames, consistent with the observed artifacts). Note that actually only some packets belonging to these frames were

dropped, and not the complete frames. We consider a frame with one or more packets missing as dropped because typical decoders will throw away incomplete frames anyway (including I- and P-frames, leading to artifacts). With IFD turned on, it can be seen that no I-frames were dropped (Figure 8(b)). Most of the dropped frames were B-frames. It can be seen that also some P-frames had to be dropped, causing the dropping of the rest of the disturbed GOPs. Considering our GOP size, dropping a disturbed GOP may cause the image to be frozen up to roughly half a second. This effect is reduced if the GOP size is smaller. The results for the RTP implementation are shown in Figure 9. As can be seen, in this experiment only B-frames were dropped.

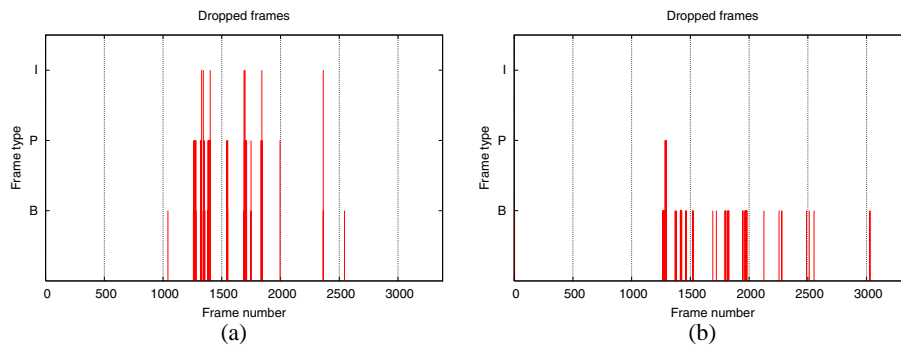


Fig. 8. Dropped frames under microwave disturbance (TCP): (a): without IFD, (b) with IFD.

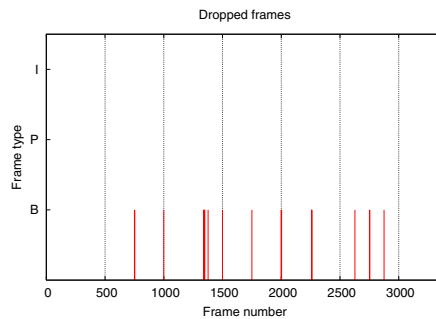


Fig. 9. Dropped frames with IFD under microwave disturbance (RTP).

4.2 Mobile receiver

In this experiment we used a 8 Mb/s sequence. The sequence was streamed along two wireless links from the sender PC to the laptop (see Figure 10). After the stream started, we walked away with the laptop from the access point until halfway the sequence, then turned around and walked back to the starting position. This experiment was only done with the TCP implementation, because for RTP we lacked IFD on the access point for adapting the stream on the second link.

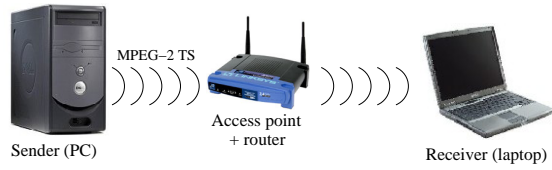


Fig. 10. Experimental set-up with mobile receiver.

With IFD off, it was observed that while walking away, the audio and video were first streamed with no problems, followed by a short period with artifacts, then followed again by a period without problems. This is caused by the automatic reduction of the transmission rate by the access point when the signal strength decreased. Hence the artifacts were caused during the transitions. This effect can also be seen in Figure 11(a) from the short bursts of dropped frames. When we reached a certain distance where the required bandwidth could not be sustained anymore, the receiver output exhibited continuous artifacts. With IFD on, we observed that during transitions the image showed a short freeze (caused by dropping a P-frame and the subsequent frames in the disturbed GOP). There were no blocking artifacts and no audio interruptions. As can be seen from Figure 11(b), the dropped frames were limited to B-frames and P-frames.

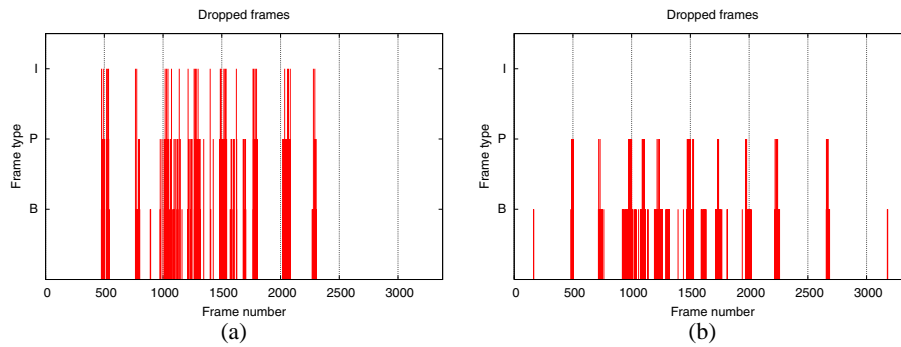


Fig. 11. Dropped frames with roaming receiver: (a) without IFD, (b) with IFD.

5 Conclusions

In this paper, we have shown a method for adaptive streaming of audio/video content over wireless networks while using standard content distribution formats and network protocols. With I-Frame Delay as the underlying bit-rate adaptation mechanism, we are able to stream artifact-free video even under degrading network conditions. Our solution is implemented at the sender only, no modifications are needed at the receiver side. It is possible to apply IFD in combination with RTP as well as TCP. We proposed a packet encapsulation scheme which makes it possible to separate video TS packets from non-video packets and packets belonging to different frames, such that the resulting packets can be fed to the IFD scheduler. The effectiveness of IFD in case of network bandwidth fluctuations was shown by means of measurements on a real-life demonstrator set-up.

Our RTP packet encapsulation scheme results in on average smaller network packets, which entails some overhead in network efficiency. This overhead is dependent on the encapsulation efficiency of the streams. For TCP this overhead is considerably smaller.

IFD is effective against bandwidth fluctuations, which may be severe but only last a short period of time. In case of long-term bandwidth drops (e.g. due to multiple contending streams), the perceived quality of the IFD solution seriously degrades because frames are constantly dropped. Such bandwidth problems are better handled by solutions such as transcoding.

References

1. D. Wu, Y. Hou and Y.-Q. Zhang. Scalable Video Coding and Transport over Broadband Wireless Networks. In *Proceedings of the IEEE*, vol. 89, no. 1, 2001.
2. M. Domanski, A. Luczak and S. Mackowiak. Spatio-temporal Scalability for MPEG Video Coding. In *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, 2000.
3. H. Radha, M. van der Schaar and Y. Chen. The MPEG-4 Fine-Grained Scalable Video Coding Method for Multimedia Streaming over IP. In *IEEE Transactions on Multimedia*, vol. 3, no. 1, 2001.
4. Kwang-deok Seo, Sang-hee Lee, Jae-kyoon Kim and Jong-seog Koh. Rate-control algorithm for fast bit-rate conversion transcoding. In *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, 2000.
5. P. Assuncao and M. Ghanbari. A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams. In *IEEE Transactions on circuits and systems for video technology*, vol. 8, no. 8, 1998.
6. Yantian Lu and Kenneth J. Christensen. Using Selective Discard to Improve Real-Time Video Quality on an Ethernet Local Area Network. In *International Journal of Network Management*, Volume 9, Issue 2, 1999.
7. Ricardo N. Vaz, Mário S. Nunes. Selective Discard for Video Streaming over IP Networks. In *Proceedings of the 7th Conference on Computer Networks (CRC2004)*, 2004.
8. Sergei Kozlov, Peter van der Stok, Johan Lukkien. Adaptive Scheduling of MPEG Video Frames during Real-Time Wireless Video Streaming. In *Proceedings of WoWMoM*, 2005.
9. Reinder Haakma, Dmitri Jarnikov, Peter van der Stok. Perceived quality of wirelessly transported videos. In *Dynamic and Robust Streaming in and between Connected Consumer-Electronic Devices*, Philips Research Book Series, vol. 3. Springer, 2005.
10. SangHoon Park, Seungjoo Lee, JongWon Kim. Network-adaptive high definition MPEG-2 streaming over IEEE 802.11a WLAN using frame-based prioritized packetization. In *Proceedings of the 3rd ACM international workshop on Wireless mobile applications and services on WLAN hotspots (WMASH)*, 2005.
11. ISO/IEC 13818-1:2000(E). Information technology – Generic coding of moving pictures and associated audio information: System.
12. D. Hoffman, G. Fernando, V. Goyal and M. Civanlar. RFC2250: RTP Payload Format for MPEG1/MPEG2 Video.
13. Jangeun Jun, Pushkin Peddabachagari and Mihail Sichitiu. Theoretical Maximum Throughput of IEEE 802.11 and its Applications. In *Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications (NCA'03)*, 2003.
14. Digital Living Network Alliance. Home Networked Device Interoperability Guidelines, Version 1.5, 25-10-2005.