Adaptive streaming of MPEG-based audio/video content over wireless networks

Document status and date:
Published: 01/01/2006

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

Adaptive Streaming of MPEG-based Audio/Video Content over Wireless Networks

Marek Burza, Jeffrey Kang, Peter van der Stok

Philips Research, High Tech Campus 31, 5656 AE Eindhoven, The Netherlands

E-mail: {Marek.Burza, Jeffrey.Kang, Peter.van.der.Stok}@philips.com

Abstract— **This paper describes a method for robust streaming of combined MPEG audio/video content (encoded either with MPEG-2 or MPEG-4/H.264) over in-home wireless networks. We make use of currently used content distribution format (MPEG Transport Stream) and network protocols (either RTP or TCP/HTTP). 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.**

Index Terms— **adaptive video streaming, wireless networks, MPEG, H.264, I-Frame Delay**

I. 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 II lists some related work. Section III describes our adaptive streaming technique and its implementations. Experimental results are presented in Section IV. This paper ends with conclusions and final remarks.

This paper is an extension of earlier publication submitted to IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services [1]. Additional work included here covers extension of the demonstator server and presented algorithm to handle also the streams encoded with MPEG-4/H.264 codec (in Section III-E), results of experiments for streams with H.264 content previously done only for streams with MPEG-2 content (in Section IV) and evaluation of picture quality delivered by our algorithm (in Section IV-C).

II. 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 separate 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 [2][3][4].
- *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 [5][6].
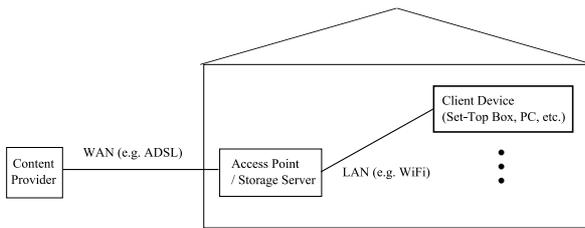
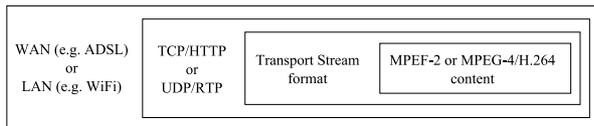Figure 1. Architecture of the streaming use case.

Figure 2. Protocol and format encapsulation.

- *Frame dropping*. Complete frames are dropped in case of insufficient bandwidth, examples are found in [7][8].

Our adaptation technique falls into the last category and is called *I-Frame Delay* (IFD), previously reported in [9]. 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 [10] 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 [11] resembles ours the most, dealing with adaptive streaming of MPEG 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.

III. ADAPTIVE STREAMING OF MPEG 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, then possibly stored on a home storage server and further distributed over a local network within the home (Figure 1). The content can be delivered either with use of HTTP or RTP protocol which transports distribution format. Here the most common distribution

format is MPEG Transport Stream (TS) [12], which can contain video encoded with a variety of different codecs (i.e. MPEG-2 and H.264 that we focus on). Therefore we will further discuss adaptive streaming of MPEG TS. Figure 2 illustrates encapsulation of protocols and formats.

Section III-A shortly introduces the problem addressed later on with our video sheduling technique. Section III-B describes our IFD adaptive video scheduling technique. Sections III-C and III-D explain how this technique can be applied on Transport Streams using RTP and TCP. Section III-E will describe the challenges faced and solutions found when applying IFD to streams with H.264 content.

A. Problem description

Streaming video over a network often faces limited bandwidth as the network link is usually shared between multiple applications. Moreover, if the video is transmitted over wireless networks (e.g. IEEE 802.11), then aside from sharing the logical link, the physical medium that the link is using is also shared between different standards (e.g. DECT phones, Bluetooth, ZigBee) and is subject to interference (such as multipath interference or microwave ovens).

When available bandwidth is insufficient for transmission of a video stream, the viewer will experience a disruption of the viewing experience. Since the default behaviour of a network stack in case of insufficient bandwidth is to drop packets at the moment of congestion, the disruption of frames (artifacts) will be random: missing parts of a frame, mixed frames, misplaced fragments of the picture, etc.

B. 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 frames[1].

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

---

[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.
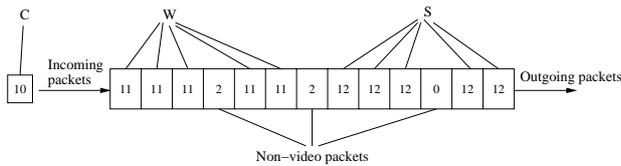
Figure 3. Network packets in the IFD buffer.

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 3 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 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 in Figure 4.

C. 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.

1) IFD-friendly RTP encapsulation: For RTP encapsulation of TS packets we adhere to RFC2250 [13]. It states that a RTP packet contains a 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 packet fragmentation. The problem of such RTP packets is that they are dropped by IFD as a whole, and they may contain

```
procedure BUFFER_ENQUEUE(C)
    if Disturbed_GOP == True then
        if C is of type I then          ▷ New GOP is encountered
            Disturbed_GOP = False       ▷ Reset disturbed GOP flag
        end if
    end if
    if Disturbed_GOP == True then
        Discard C                       ▷ Discard rest of disturbed GOP
        return
    end if
    if W is empty then
        Store C in W
    else
        if C is of type I then
            Overwrite W with C
        else if C is of type B then
            Discard C
        else if W is of type I or P then
            Discard C
            if C was of type P then ▷ Discarded frame is P-frame
                Disturbed_GOP = True  ▷ Set disturbed GOP flag
            end if
        else
            Overwrite W with C
        end if
    end if
end procedure
```

Figure 4. Pseudocode for IFD algorithm with buffer of two frames.

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 5(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 5(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).

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
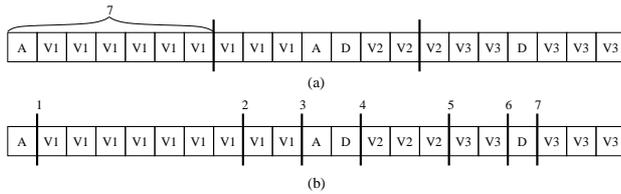
Figure 5. Example of RTP packet encapsulation: (a) common scheme, (b) IFD-friendly scheme.

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.

2) IFD overhead: As can be seen from Figure 5, our IFD-friendly encapsulation scheme results in more and smaller RTP packets. Smaller packets have more over-head, 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
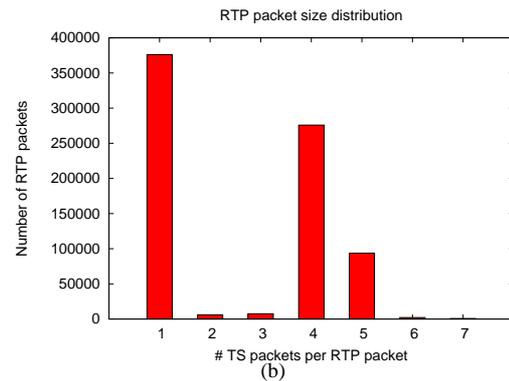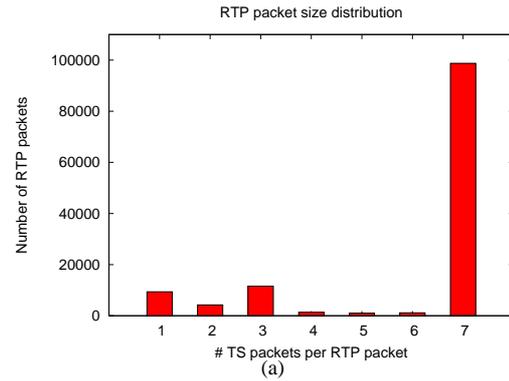

Figure 6. Example RTP packet size distribution for high (a) and low (b) encapsulation efficiency.

how well a stream can be encapsulated, that is, how close $N_{TS\_per\_RTP}$ is to the optimal number of 7:

$$E_{encap} = \frac{N_{TS\_per\_RTP}}{7} \times 100\% \qquad (1)$$

Our experience with typical Transport Streams yields values for E_{encap} ranging from 37.4% to 85.8% for MPEG-2 and 59.9% to 76.5% for H.264. Figure 6 shows the distribution of RTP packet sizes for example sequences with a high and a low 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% (MPEG-2) and from 1.63% to 3.56% (H.264) 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 [14]). Figure 7 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 47.0% for MPEG-2 and from 15.7% to 28.5% for H.264. It is therefore imperative to obtain a high encapsulation efficiency. This depends on how the packets have been multiplexed. The best is
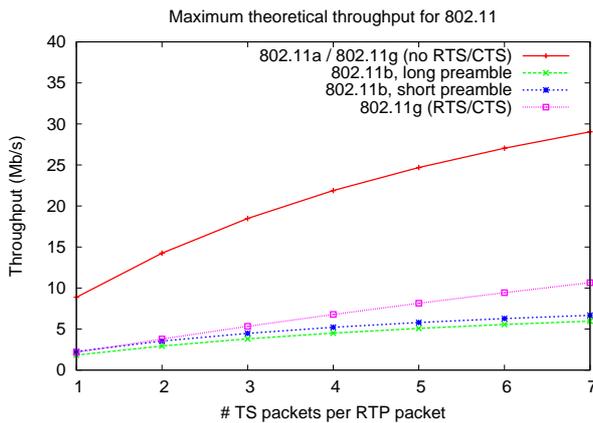
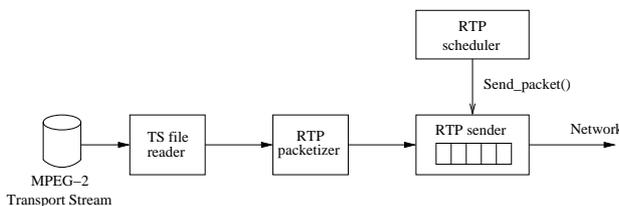Figure 7. Maximum theoretical throughput for 802.11.

Figure 8. RTP sender architecture.

Figure 9. TCP sender architecture.

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 6(a)).

3) 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 8. The *TS file reader* reads TS packets from a MPEG 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 [15]). 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 correct 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.

D. 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
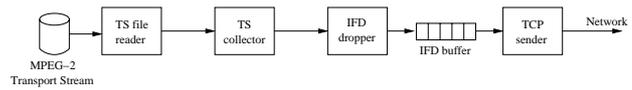
is because with RTP, only the bandwidth of the wireless link on which transmission takes place (e.g. between the sender and the access point) can be monitored (there is no feedback from the receiving peer). If the path to the receiver passes another wireless link (e.g. from access point to the receiver), then the quality of the second link is visible for the access point, but not for the sender[3]. 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 9. 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 III-C.3, 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.

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.

E. H.264 considerations

To further explore applicability of IFD to video streaming both TCP and RTP implementations were extended to handle streams with H.264 content. Like in the case of

---

[3]This could potentially be tackled by detecting ICMP Source Quench messages, however, not every router implements this feature.

MPEG-2, the H.264 content was encapsulated in MPEG Transport Stream together with audio. The choice of the codec was dictated by the high interest with H.264 in both scientific comunity and industry. Its features and the recognition it gets suggest that it is a viable successor to MPEG-2. For this reason it was important to investigate the performance of IFD with H.264 streams.

Although there are differences between the two standards, the H.264 standard keeps many of the MPEG-2 notions: it still distinguishes I, P and B frames, and encoding of the frames is in principle the same.

Prior to implementation it was necessary to perform a simple sanity check if IFD can be at all applied. Major benefit of IFD is that it allows to cope with bandwidth drop by dropping B-frames without introducing artifacts. Therefore, applying IFD depends on the contribution of B-frames to the size of the stream - if it is significant then it is reasonable to apply IFD to use the bandwidth gain. After checking our example streams with MPEG-2 content it was concluded that the contribution of B-frames is between 30% and 55% (which means that it is possible to save the same amount of bandwidth in case of network congestion). The same check for our example streams with H.264 content yielded results between 20% and 35% for SD streams and 40%-48% for HD content. This is still significant, even though it is somewhat less than in MPEG-2.

The high compression factor of the H.264 codec can be attributed to several features it supports. Many of these features are new in H.264 and were not included in MPEG-2 before. Because of these differences between these two standards, it was necessary to look closer at those features as they could have a negative impact on the possible application of IFD to H.264.

IFD relies only on extraction of frames (beginning and end or length of a frame) and detection of their type. That, in turn, depends on features related to the structure of the bitstream with frame-wise granularity and classification of the frames, which limits the scope of the features considered to the following list (for a complete list of features one can refer to [16] or [17]):

- *Indication of frame boundaries*. There are two ways to detect a beginning of a new frame in H.264. One way is to scan bitstream and detect an Access Unit Delimiter (AUD) indicating a beginning of a new frame. However, AUD is an optional element of a sequence. Another way is to parse headers of all top-level parsing units (Network Abstraction Layer units, NAL units) in the bitstream and checking values from the headers against a set of conditions defined by the standard.
- *Variable Length Compression*. If parsing of headers is necessary then this can introduce overhead through keeping storage and the analysis of the values in the headers. H.264 uses Exponential-Golomb encoding for these values, which requires more computing than for the encoding used in MPEG-2.
- *Less strict inter-frame dependencies*. Often, B-frames are a closer approximation to neighbouring B-frames than more distant I- or P-frames. Therefore to enhance video quality and compression ratio it was allowed to make B-frames dependant on other B-frames. The result of this is that some B-frames are more important than others.
- *New frame types*. H.264 introduces new frame types - SI and SP, which allow to introduce custom entry points for the purpose of random access and enhanced resilience. They basically duplicate the content of another frame - for example an SI frame duplicates another SP frame but allows the decoder to recover in case preceding frames were lost, or, in case we have two SP frames they introduce switching points between two streams (more information can be found in [18]).
- *GOP-related issues*. H.264 does not have an explicit counterpart of Group-of-Pictures structure known from MPEG-2. Instead, it has a concept of Instantaneous Decoding Refresh (IDR) picture, which can be considered as a counterpart of a beginning of a GOP.

One challenging feature is related to the detection of frames. In our research we are using Transport Streams and the standard defines that if H.264 bitstream in encapsulated in TS then AUD units are mandatory. In this case frame detection is simple and requires only a detection of the AUD units, it requires only scanning for a particular four byte long sequence in the bitstream (the test set used during this research consisted only of sequences with AUD units). We are also able to handle a situation where AUD units are absent. In this case the server has to do more detailed parsing of the sequence (it has to be robust enough for sequences with arbitrary slice ordering, consecutive frames of the same type etc.) and has to manage necessary data for parsing and detection.

Parsing of a H.264 bitstream and frame detection in absence of the AUD units can be a resource consuming process. One reason for this would be storage and lookup of values from previously parsed headers, but a more significant reason is the encoding used in the H.264. To avoid sending any unnecessary data even in the control part, it was decided to use variable length encoding (Exponential-Golomb) with bit alignment (not a byte alignment) and not to use any leap values. The fact that no leap values are used (like length of a header or a parsing unit) makes it impossible to skip values that are of no interest - everything needs to be parsed. This can of course require some CPU cycles, but by an experiment it was established that the cost is minimal. For Pentium 4 1.7GHz with Windows XP the detection of frames by using the slice encoded info adds only about 0.192% processing time per frame as compared to the detection based on the Access Unit Delimiters and for MIPS in a set-top-box running Linux the difference is 0.032% (both values measured for a 2 Mb/s stream).

The next feature that might pose problems is using B-frames as reference frames. In this case it is necessary

to distinguish between "more" and "less" important B-frames. Tagging a priority to frames is no longer equivalent to their type only. In general, that could require parsing and analysis of a sequence to be even more detailed. Fortunately, the parsing unit (NAL) header holds a field which indicates whether given unit is referenced or not - this can aid the detection of referenced B-frames without adding much complexity to the decision algorithm. The way the field is used, reflects the level of dependency on given unit, for example - I-frame receives value 2, P-frame 1 and unreferenced B-frame 0. Again, our test set included sequences without B-frames as reference frames.

Another new feature is the introduction of new frame types, SP and SI. Although their role is slightly different then of the P- and I-frame, their priority can remain the same, which allows using the same tags and leaving decision algorithm unchanged.

In MPEG-2, it used to be possible that B-frames at the beginning of GOP were referencing frames from previous GOP (so-called open GOP). This, however, was limited to a GOP boundary. In H.264, to enable management of reference buffers a concept of Instantaneous Decoding Refresh (IDR) picture was introduced. Only I-frames can be IDR frames, and - when encountered - such frame indicates that all reference frames should be marked as unused. Therefore, even though H.264 does not recognize formally a notion of GOP, it still keeps this functionality.

The IFD also handles situations where GOP is disturbed by dropping P- or I-frame. I this case all following frames are dropped until beginning of the next GOP. The reason for this is to drop frames that depend on a missing frame (these frames would be displayed with artifacts). H.264 does not have a notion of GOP, but fortunately it is possible to detect a boundary beyond which there is no reference made. The boundary is an IDR-frame. Thanks to that, also this IFD feature remains valid for H.264 - IFD algorithm only requires different source of information used in the decision. Long distances between IDR-frames might be then the only problem. However, for a broadcast content the distance is supposed to be short to avoid sparse recovery points; trick-play for stored content requires that as well. Otherwise, a mechanism analysing references between frames would have to be in place in order to detect when a dropped frame is not referenced anymore. A trade-off is possible: dropping can simply be continued until next I-frame or IDR-frame, which, in case of sparsely distributed IDR-frames, would mean that dropping I- or P-frame would result in longer periods when the picture is frozen.

IV. 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 IV-A), the wireless network is disturbed by turning on a microwave. The second scenario (Section IV-B) involves a mobile receiver, which is moved away from the access point, causing the bandwidth to drop. We used a 802.11g wireless access
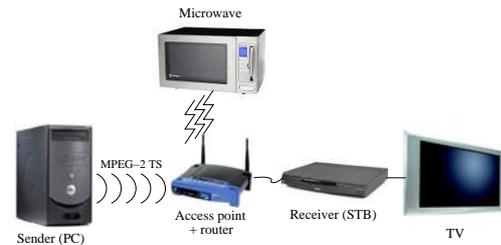

Figure 10. Experimental set-up with stationary receiver.

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.

A. Stationary receiver with microwave disturbance

The set-up for this scenario is shown in Figure 10. 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 MPEG-2 sequence (with phy-rate set to 54 Mb/s) and 4 Mb/s H.264 sequence (with phy-rate set to 24 Mb/s). The GOP size in both cases is 15 and the GOP structure is IBBP. During the 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. For both types of streams (MPEG-2 and H.264) we first ran an experiment with the TCP sender (Figure 9) 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 11 (a) and (c). The observations for MPEG-2 and H.264 were similar. 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 11 (b) and (d)). Most of the

Figure 12. Dropped frames with IFD under microwave disturbance (RTP): (a) MPEG-2, (b) H.264.

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 12. As can be seen, in this experiment only B-frames were dropped.

B. Mobile receiver

In this experiment we used a 8 Mb/s MPEG-2 sequence (with phy-rate set to 54 Mb/s) and 4 Mb/s H.264 sequence (with phy-rate set to 24 Mb/s). The sequence was streamed along two wireless links from the sender PC to the laptop (see Figure 13). 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.

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
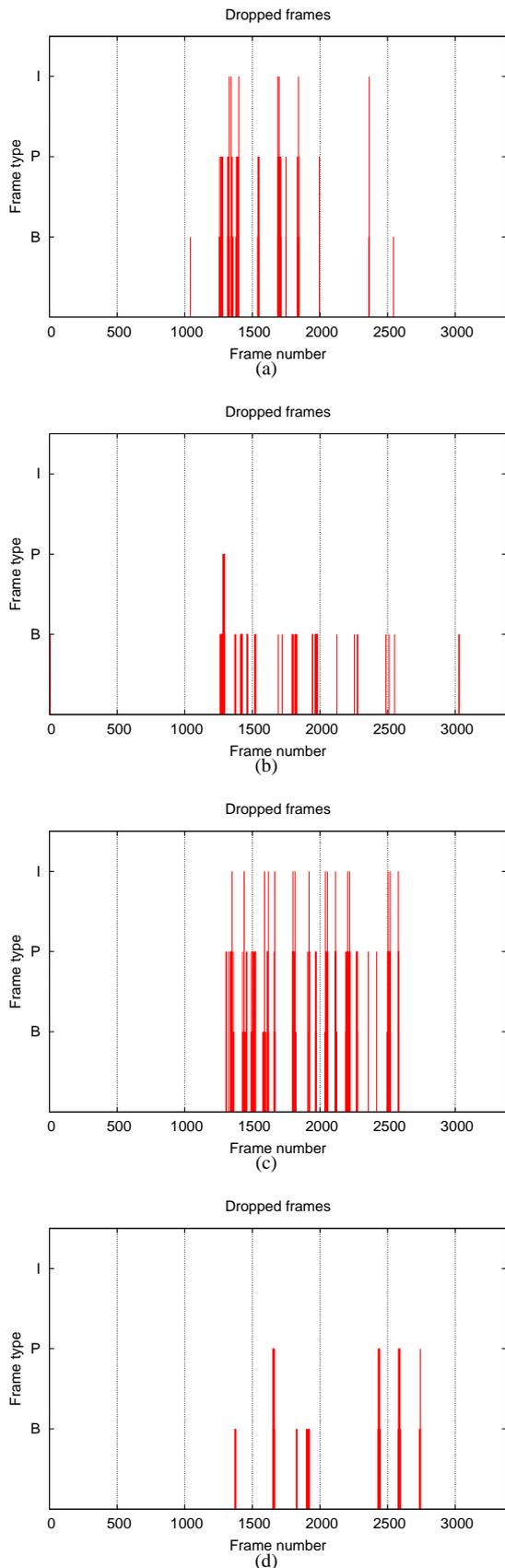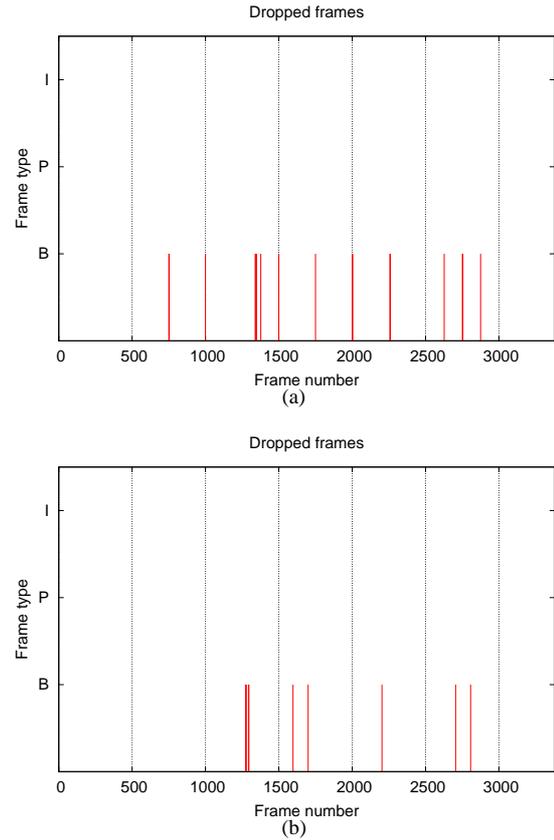

Figure 11. Dropped frames under microwave disturbance (TCP): (a) MPEG-2 without IFD, (b) MPEG-2 with IFD, (c) H.264 without IFD, (d) H.264 with IFD.
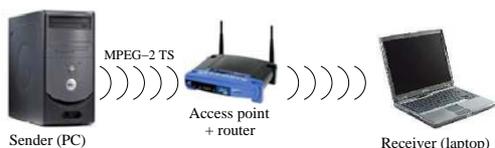
Figure 13. Experimental set-up with mobile receiver.

access point when the signal strength decreased. Hence the artifacts were caused during the transitions. This effect can also be seen in Figure 14 (a) and (c) 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 14 (b) and (d), the dropped frames were limited to B-frames and P-frames.

C. Picture quality delivered with IFD

Additionally, to confirm that IFD delivers better quality than standard scheduling mechanism we performed an experiment using scenario from Figure 10 where the sequences were recorded as delivered on receiver side and the Peak Signal-to-Noise Ratio (PSNR) was calculated against the original sequence.

The experiment was done for both MPEG-2 and H.264. The bandwidth was continuously suppressed to a value just above the bitrate of the sequence to ensure a big enough measurement sample (total number of frames sent was 3370). Limitation of bandwidth was done with use of the Token Bucket Filter (TBF) - a queuing mechanism available on Linux, which can be used for traffic control (the reason for using it, is to be able to arbitrarily configure available bandwidth). For MPEG-2, a 10 Mb/s sequence was used. Bandwidth was set to 10.5 Mb/s and no additional interference was introduced. Similarly, for H.264 we used a 4 Mb/s sequence with bandwidth limited to 4.2 Mb/s.

The PSNR values were obtained by comparing YUV files decoded from original and recorded sequences with ffmpeg decoder. In case of missing frames this transcoder copies most recent one so the number of frames and timing still matches the original - which is consistent with observations we made during playback of the streamed video. The results of these experiments are presented in two sets - one presents how many frames were disturbed during transmission (Table I) and another presents average PSNR (Table II). Note: The average PSNR was calculated for disturbed frames only.

In both cases - MPEG-2 and H.264 - we can observe that the number of disturbed frames is reduced by similar factor when IFD is used. Also, in both cases the quality of the disturbed frames is better - this can be attributed to
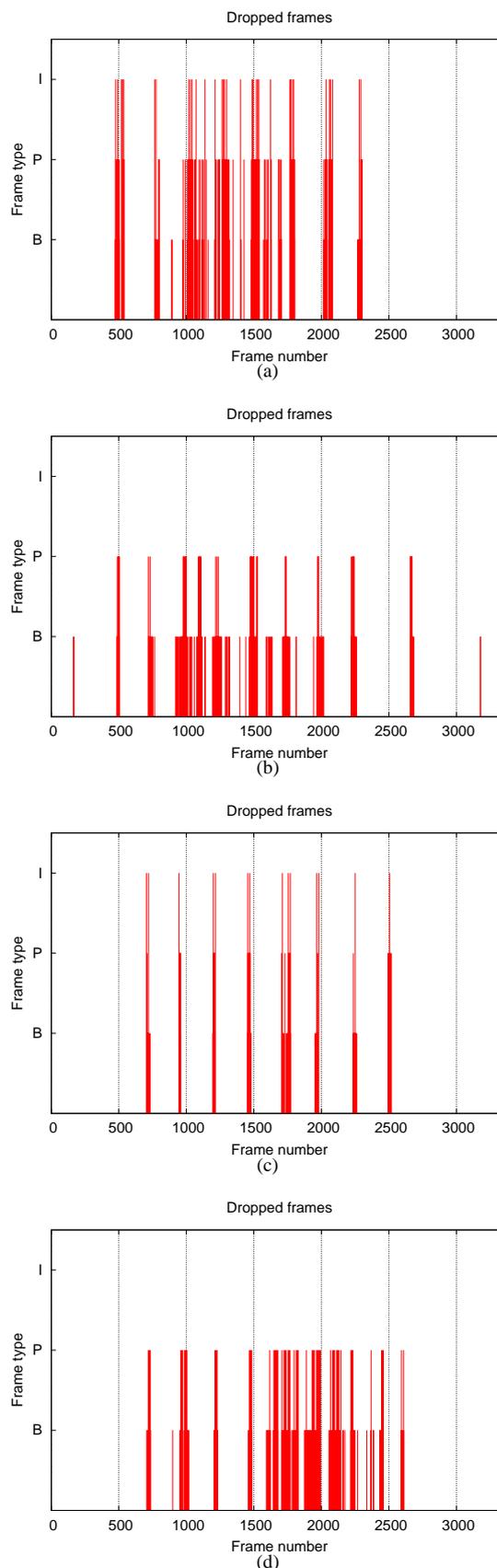

Figure 14. Dropped frames with roaming receiver (TCP): (a) MPEG-2 without IFD, (b) MPEG-2 with IFD, (c) H.264 without IFD, (d) H.264 with IFD.

	Without IFD	*With IFD*
MPEG-2	54.58%	8.21%
H.264	83.26%	39.85%

TABLE I.
PERCENTAGE OF DISTURBED FRAMES.

	Without IFD	*With IFD*
MPEG-2	22.40	24.14
H.264	19.56	25.60

TABLE II.
AVERAGE PSNR OF DISTURBED FRAMES.

the fact that for IFD there are no artifacts in the disturbed frames - they are simply repeated frames.

V. 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. Moreover, it allows to reduce the number of affected frames and this together artifact-free transmission it results in a higher picture quality. Our solution is implemented at the sender side only, no modifications are needed at the receiver side. We also found out that IFD is flexible enough to be applied to both MPEG-2 as well as H.264 (and possibly also to other video codecs when necessary adaptation is made). 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 into 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 transrating.

ACKNOWLEDGMENT

REFERENCES

[1] J. Kang, M. Burza, P. van der Stok. Adaptive Streaming of Combined Audio/Video Content over Wireless Networks. In *Proceedings 9th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services*, 2006

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] ISO/IEC 13818-1:2000(E). Information technology – Generic coding of moving pictures and associated audio information: System.

[13] D. Hoffman, G. Fernando, V. Goyal and M. Civanlar. RFC2250: RTP Payload Format for MPEG1/MPEG2 Video.

[14] 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.

[15] Digital Living Network Alliance. Home Networked Device Interoperability Guidelines, Version 1.5, 25-10-2005.

[16] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra. Overview of the H.264 / AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, July 2003

[17] H.264: Advanced video coding for generic audiovisual services. ITU-T, March 2005

[18] E. Setton, B. Girod. Rate-Distortion Analysis and Streaming of SP and SI Frames. *IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 6, pp. 733-743,* 2006

Marek Burza was born in Tarnów, Poland. He received his M.Sc. degree in Computer Science from the Wrocław University of Technology in 2002. He then joined Siemens Mobile Communications Dept. as a SW developer and a project leader. In 2005 he moved to Philips Research and had worked on streaming platforms for multicore processors and location services in wireless sensor networks. His recent research is devoted to Quality-of-Service for networked multi-media streaming.

Jeffrey Kang was born in Taipei, Taiwan. He received his M.Sc. degree in Electrical Engineering from the Delft University of Technology in 1999. He then joined Philips Research as a research scientist, where he has worked on various projects related to hardware/software communication, streaming models and interfaces, and rapid system prototyping. His recent research interests include (wireless) networked multi-media streaming and Quality-of-Service. He currently works at the Innovation Lab of Philips Customer Electronics.

Peter van der Stok holds a M.Sc (1973) and a Ph.D. (1982) degree in physics from the University of Amsterdam. He has worked from 1973 till 1989 at CERN where he became responsible for the operating system software of the distributed control system of the LEP accelerator. He worked from 1989 at Eindhoven University where he is associate professor. His main interests are distributed fault-tolerant real-time systems. Since September 2000 he works for 90% of his time at Philips Research on the subject of In-Home Digital Networks and more recently on Wireless Sensor Networks.