Link Layer Support for Streaming MPEG Video over Wireless Links

Rohit Kapoor, Matteo Cesana, Mario Gerla

Abstract: Streaming video as a form of media is becoming increasing popular on the Internet. Real-time media such as video requires delay constraints from the network to ensure good quality at the receiver. While watching a video stream on his portable device connected to the Internet through the last-hop wireless link, the mobile user of tomorrow will expect a good experience. But, the time-varying nature of the wireless link can cause video frames to be dropped/delayed, which can affect the quality of video at the receiver. In this paper, we propose a link layer scheme to improve the quality of MPEG video streaming over a wireless link. We use Bluetooth as the wireless technology on which to test our scheme. Our results show that the quality of streaming video can be substantially improved with our scheme, particularly in bad channel conditions.

1. INTRODUCTION

The goal of wireless networks is to replicate the user experience in a wired environment, in terms of connectivity and traffic support. Future wireless technologies need to be able to handle multimedia traffic in an effective way. The scenario we address here is shown in Fig 1 where a user with a wireless (802.11b [1] or Bluetooth [2])-enabled device is streaming video to the device. The access point may be either 802.11 or Bluetooth [3] or both [4] (some companies are developing hybrid 802.11/Bluetooth access points to support both kinds of users). Providing good multimedia support in such wireless systems is a tricky problem since wireless channels are characterized by extreme variability and transmissions can be impaired by phenomena like multipath fading and shadowing.

Fig 1: Streaming multimedia through the access point

Streaming video is one media that is becoming increasingly popular on the Internet. For a good user experience, it is required that video packets reach the user at regular intervals. Video packets have delay constraints, and honoring these delay constraints is generally considered more important than whether the data reaches "uncorrupted" or not. Of the various video codecs, the emerging MPEG-4 [6] video standard is gaining a lot of acceptance for use on the Internet. MPEG-4 uses an inter-frame [5] compression algorithm that exploits temporal correlation between frames to achieve high levels of compression. This algorithm represents most of the frames as differences from reference frames. Though it achieves good compression, it suffers from the 'propagation of errors' [7] effect in which errors in a reference frame propagate to other frames. Thus, a single loss of a reference frame can cause a big drop in the quality of the perceived video. It is also obvious that a loss or error in a reference frame is much more significant than a loss or error of a dependent frame. This points towards the need to protect these reference frames with a higher priority of some kind.

In this paper, we present a simple link layer technique to counter the effects of wireless errors while streaming MPEG-4. We make the link layer application-header aware which enables it to distinguish important video frames from other not so important frames. Armed with this information, the link layer increases the retransmission count of the important frames compared to the less important ones. Note that the technique of differentiating between video frames on the basis of their semantic importance has been explored before [8] [9] [10], but to the best of our knowledge, never at the link layer. The advantage of doing this at the link layer is that the delay of a whole round trip time resulting from retransmitting a packet at the application layer is avoided. We also address how this scheme may be practically implemented. We show that this technique is very effective and increases the PSNR (Peak Signal to Noise Ratio) [11] quality of MPEG video significantly, especially in high error conditions.

Various techniques have been proposed to enhance the support of video over both wired and wireless links. Techniques based on retransmissions [8], FEC [12] [13], layered coding [14] etc have been proposed. The idea of exploiting the information on the type of the video frame to improve video quality has also been proposed in the literature [8][9][10], but only at the application layer. The related work is discussed in more detail later. The novelty of our approach lies in the fact that we apply this differentiation of video frames at the link layer and show its advantage over wireless links.

The wireless technology that we use to evaluate our scheme is Bluetooth, which is being included in more and different kinds of devices. We compare a Bluetooth stack enabled with our scheme with a regular Bluetooth stack. Note that the current Bluetooth specification does not provide any special support for video, though, as we describe later, some API calls are provided to support streaming. The scenario here could be a user with a Bluetooth-enabled PDA, streaming video through his 3G cellphone or a Bluetooth access point. Note that the scheme is wireless technology agnostic and could be as well applied to other technologies such as 802.11.

The paper is organized as follows. Section 2 gives a brief overview of the MPEG4 video flow structure that we exploit in our scheme. In Section 3, we introduce our scheme of frame differentiation and discuss implementation issues. Section 4 contains a performance analysis of the scheme and in Section 5 we give the conclusions of our work.

2. MPEG Basics

In this section, we briefly describe the basics of the video compression used in MPEG and discuss the measure of video quality called PSNR.

2.1 MPEG

The MPEG compression standard (1, 2 and 4) for video makes use of temporal and spatial redundancies in video to achieve substantial compression. Temporal redundancy exists due to a lot of similarity between consecutive video frames. To exploit this redundancy, I (intra-coded) frames are coded independently of other frames, whereas P and B frames are coded using other frames as a reference (P frames are coded from the previous closest frame whereas B frames are coded bi-directionally from the preceding and succeeding frames). Thus, for frames other than I frames, the amount of information to be coded reduces to differences between frames. Fig 2 shows an example of this in which a sequence of I frames followed by P and B frames repeats itself.

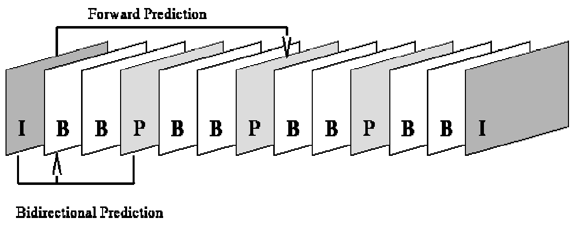

Fig 2: Frame dependencies in MPEG

Note that this differential coding means that I frames are very important since all future frames till the next I frame are coded (directly or indirectly) using this I frame. If an I frame is lost (or too delayed), the next few frames effectively carry no semantic information.

P and B frames, on the other hand, have lesser importance. The loss of a P frame typically shows up as blobs in the video image. This typically affects some part of the image with the rest of the image still being visible. These blobs propagate till the next I frame in the sequence. Thus, the loss of P frames causes a portion of the image to be affected, whereas the loss of I frames renders all P and B frames after it (till the next I frame) completely useless. In [7], Feamster et al have shown that loss of I frames is much worse than that of P frames; our experiments have resulted in similar conclusions.

2.2 PSNR

Though the quality of video can in reality only be judged perceptually, quantifiable measures such as PSNR are widely used. PSNR (Peak Signal to Noise Ratio) is a coarse indicator of video quality that is derived from the Root Mean Square Error. PSNR is expressed mathematically as:

$$PSNR = 20 \log_{10} \frac{255}{\Sigma[f(i, j) - f'(i, j)]^2 / (N1 * N2)}$$

where f' is the degraded image of the N1*N2 8-bit original image f.

Though some other measures such as [15] [16] are supposedly better indicators of video quality since they take into account the behavior of the human visual system, we make use of PSNR in our experiments for its simplicity. Moreover, studies [17] by the VQEG (Visual Quality Experts Group) have shown that PSNR is not really much worse than any other sophisticated measures (particularly for higher bit rates).

3. Our Scheme

Packet errors on wireless links necessitate that the link layer perform some kind of ARQ to protect packets. In wireless technologies such as 802.11b and Bluetooth, in fact, it is possible to specify the retransmission limit (the number of times a packet should be retransmitted before being dropped). In 802.11 b, the retransmission limit[1] can be specified explicitly, whereas in Bluetooth, this can be specified using the Flush Timeout command (we describe this in detail later). These features can be used to provide support for real-time traffic.

As described earlier, I frames in MPEG video are much more important than P and B frames since a number of future frames depend on them. Our scheme prioritizes the transmission of I frames by increasing their retransmission limit (compared to that of P/B frames) at the link layer. Since increasing the retransmission limit of I frames can lead to other frames being delayed at the receiver, which can be as bad as dropping frames, we reduce the retransmission limit of other (P and B) frames. The basic principle behind the scheme is the following:

"If the bandwidth reserved for a video flow allows each frame to be retransmitted 'x' number of times (on the average), then the quality of video can be increased by increasing the number of times I frames are retransmitted and decreasing the number

[1] Retransmission limit is the maximum number of times a packet can be retransmitted.

of times the 'dependent' P/B frames are retransmitted (keeping the total bandwidth the same)". Note the use of the term 'dependent' which means that P/B frames whose reference I frames have been dropped have no chance of being decoded and constitute wasted bandwidth. For such P/B frames, it is better to retransmit the reference I frames a larger number of times at the cost of dropping some of the P/B frames.

In a sense, we are trading off the increase in reliable reception of I frames with a decrease in reliable reception of P/B frames. The increased importance of I frames justifies the use of this technique. In Section 4, we give a simulation result to support this argument.

Fig 3 (a) illustrates the technique used in our scheme. The figure shows packets from a video flow arriving at a link layer. A certain amount of bandwidth is reserved for the video flow. The first I frame is corrupted in transmission and is retransmitted twice. This causes the last four P/B frames to be dropped. For comparison purposes, Fig 3 (b) shows the situation without our scheme, in which the I frame is dropped after one retransmission but more P/B frames are transmitted than in Fig 3 (a).

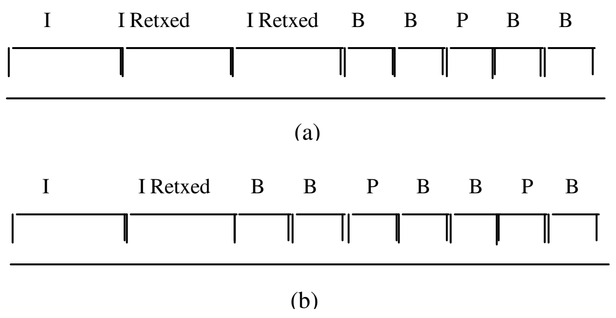

Fig 3: Illustration of the scheme

3.1 Related Work

Previous work has addressed support of video over the Internet. Various techniques based on retransmissions, FEC, layered coding etc have been proposed. FEC techniques add redundant data which is used by the receiver to reconstruct the original data in case of network losses. In [18,17], the authors have shown that FEC bandwidth overhead required to support burst losses may be as much as 30%. Layered coding techniques have also been used by various authors. QAL (quality assurance layering), in which each frame is temporally dependent only on essential parts of previous frames, has been studied in [19] [20]. The video-conferencing tool vic [21] encodes each frame as an intra-frame, at the cost of a smaller compression ratio.

Since our work falls in the retransmission techniques category, we now describe such related work. Retransmission techniques can provide error resilience for video traffic without incurring too much bandwidth overhead. Yet, such techniques, in general, have been considered unsuitable for real-time applications since retransmission will require at least one additional round-trip delay, which may be unacceptable. Wah et al argued that error recovery via retransmission schemes are not suitable for real-time video due to the imposed delay [22]. On the other hand, various techniques have been proposed which can potentially make retransmission work. Papadopolous et al [23] discussed techniques such as playout buffering, gap-based loss detection etc and showed that retransmission-based techniques can be applied to scenarios where round-trip delay is not too large. Kleinrock et al [10] proposed a scheduling scheme that decides which packets to transmit based on predictions about how layers in future frames may be delivered. Note that this list of references is in no way exhaustive.

Our scheme is inspired by some of the retransmissions-based schemes proposed in the literature. The biggest criticism of these techniques is the extra round-trip delay involved; we offset this disadvantage by applying such techniques at the link layer. Giving different priorities to I, P and B frames has been studied before, but to the best of our knowledge, never at the link layer.

3.2 Reading the Application Header

One question that arises is: how does the Link Layer identify whether the received packet received contains an I, P or B frame? There could be two approaches for this:

a) *Interaction between Application and Lower layers:* Cross-layer optimization for wireless networks has been proposed earlier [24]; future systems could have API calls between applications and lower layers.

b) *Reading Application Header Information:* The Link Layer could be provided with the information required to read and understand application layer headers. We adopt this second approach in our experiments.

3.3 Feasibility

In traditional wired networks, packet losses are mostly due to congestion. Thus, retransmissions at the link layer are not beneficial. Also, "core" routers will need to perform high-speed routing functions, and such a link layer scheme could add huge processing overhead. The feasibility of our scheme arises from the fact that it is specifically applicable to wireless networks, which exist at the edge of the Internet, where high-speed 'routing' functions are not necessary. The scenario we address is like the one shown in Fig 1, where the wireless access point (base station) performs such simple functions to improve video quality over the last wireless hop.

Another objection to our scheme could be the breaking of the layering protocol, i.e., the link layer needs to have knowledge of application layer headers. Use of cross-layer optimization has been proposed before in the context of wireless networks [24] [25]. We believe that such techniques can prove to be very useful for wireless links and in fact, may be the only way to counter their highly unpredictable nature.

3.4 Implementation of the Scheme

We used Bluetooth as the wireless technology on which to test our scheme. We have a detailed simulator of Bluetooth (described in the next section). Fig 4 shows the layers of the Bluetooth stack. Note that the baseband and radio layers are hardware, whereas the other layers are usually software. The Bluetooth specification [2] defines a set of API commands to get/set state in the baseband. One set of these API commands is related to 'flushing' an L2CAP (L2CAP is the link layer of Bluetooth) packet from the Baseband (flushing means that the packet is flushed from the baseband queue, i.e., dropped). There are three commands in the set of 'Flush' API:

Flush: This command is used to discard all data that is currently pending for transmission in the Baseband for the specified connection handle, even if there are currently chunks of data that belong to more than one L2CAP packet in the Baseband.
Write Flush Timeout: This commands is used to write the value for the Flush_Timeout parameter for the specified connection handle. The Flush_Timeout parameter defines the amount of time before all chunks of the L2CAP packet, of which a baseband packet is currently being transmitted, are automatically flushed by the Host Controller. The timeout period starts when a transmission attempt is made for the first baseband packet of an L2CAP packet. This allows packets to be automatically flushed without the Host issuing a Flush command. The Flush Occurred event occurs when the Flush_Timeout for an L2CAP packet has expired and the packet is flushed.
Read Flush Timeout: This command is used to read the value of the Flush_Timeout parameter.

For our purposes, we use the Write Flush Timeout command to assign different Flush Timeouts to different video frames (I, P and B). Note that the Flush command can also be used but this would entail maintaining state regarding the arrival time of each packet at the L2CAP/HCI layers.

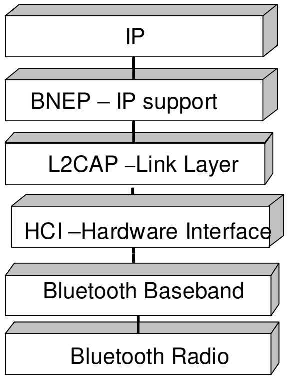

Fig 4: Layers of the Bluetooth stack

4. Simulation Results

In this section, we evaluate the improvement achieved by our scheme over the Bluetooth wireless link (the scheme could work in a similar manner over 802.11 b). We have a detailed simulator of Bluetooth that contains most of the standard features of Bluetooth like Frequency Hopping, Multi-Slot Packets, Fast ARQ (Automatic Retransmission Query). The Bluetooth model also defines a channel model, which is defined below. We also enhanced the Bluetooth link layer (L2CAP/HCI) to incorporate our scheme.

4.1 Channel Model

In order to evaluate the performance of the proposed scheme we developed a channel error model based on a three-state Markov chain. Each state in the chain is characterized by a certain Packet Error Probability (PER): the Good state has PER=0.028, the Medium state with PER=0.13 and a Bad state with PER=0.32. By setting appropriate transition probabilities among the states, we are able to simulate different channel conditions. We report in Fig 5 the transition matrices used to simulate a Fair, Medium and Poor channel respectively, where the first row/column represents the Fair state, the second the Medium state and the third the Bad state.

$$H_{Fair} = \begin{bmatrix} 0.99 & 0.008 & 0.002 \\ 0.23 & 0.7 & 0.07 \\ 0.1 & 0.25 & 0.65 \end{bmatrix} \quad H_{Medium} = \begin{bmatrix} 0.99 & 0.008 & 0.002 \\ 0.23 & 0.7 & 0.07 \\ 0.1 & 0.25 & 0.65 \end{bmatrix}$$

$$H_{Poor} = \begin{bmatrix} 0.9 & 0.08 & 0.02 \\ 0.08 & 0.85 & 0.07 \\ 0.09 & 0.11 & 0.8 \end{bmatrix}$$

Fig 4: Transition probabilities for Fair, Medium and Bad states

Table 1 reports the average PER in the three simulated wireless channels.

Fair	0.028
Medium	0.13
Poor	0.32

Table 1: Packet Error Rates

4.2 Simulations

In order to model MPEG 4 traffic, we used traces of Starship Troopers from [26]. These traces contain information about the frame number, frame type (I, P or B), time of generation and size of each video frame. The traces are coded from the video source at different bitrates, ranging from about 40Kbps to about 3Mbps. We fed these traces into our Bluetooth simulator.

The simulation topology consisted of a Bluetooth piconet consisting of a master and a variable number of slaves (equal to the number of video flows). We considered different MPEG4 flows with bitrates varying from 64Kbps to 512Kbps.

We first performed a simple experiment to validate our scheme. We considered a 256 Kbps video stream being streamed from a Bluetooth master to a slave. The video stream was allowed to use a fixed reserved bandwidth of about 170 Kbps. The size of an I frame in the video was approximately equal to 1/3 of the size of a complete GOP; the I frames, thus, needed a bandwidth of about 85 Kbps and the rest was needed by P/B frames. We gave a certain percentage of the bandwidth to I frames and distributed the rest randomly among P and B frames. Frames that did not receive any bandwidth were dropped at the link layer. We increased the bandwidth given to I frames; this decreased the bandwidth given to P/B frames. Fig 6 shows the PSNR values of the video quality as the percentage of video frames accepted for transmission is varied. The increase in video quality as I frames are given a higher percentage of the bandwidth is clearly visible. This validates the basic premise of our scheme of supporting I frames with higher 'priority' at the link layer.

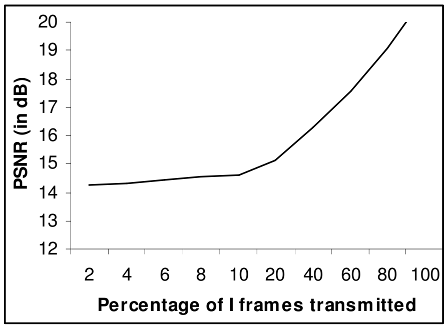

Fig 5: PSNR values versus percentage of I frames

In the next experiment, four 128 Kbps video flows were downloaded (streamed) by four slaves from the master (access point) and the channel quality was varied as good, medium and bad. The ratio of the maximum number of retransmissions for I frames compared to that for other frames was varied, keeping the total bandwidth used the same (some P/B frames needed to be dropped for this). Fig 7 shows the PSNR quality of the video flows. As I packets are given a higher priority (higher ratio of retransmissions), the PSNR value increases. In fact, the increase is very significant for medium and low quality channels, being almost equal to 5dB. Note that a simple scheme like ours leads to significant increase in quality of video flows, especially as channel conditions become bad. This clearly points to the usefulness of cross-layer optimization schemes to support real-time traffic in wireless environments.

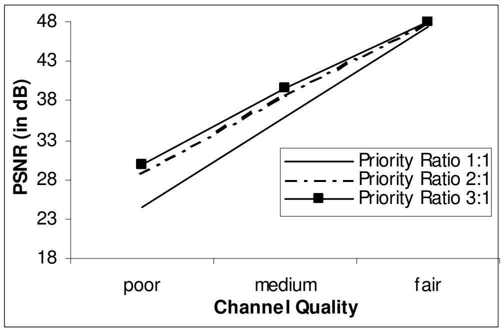

Fig 6: PSNR values for four 128Kbps video flows

Finally, one 512 Kbps flow was downloaded by a slave from the access point. The PSNR values for video quality are shown in Fig 8. Again, the increase in video quality can be seen.

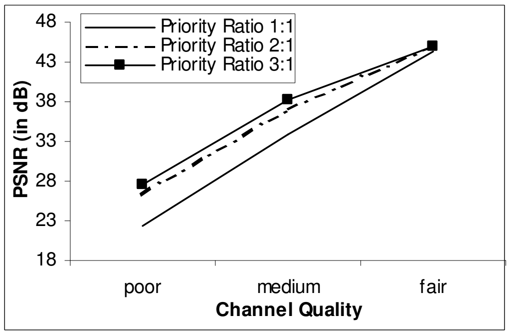

Fig 7: PSNR values for one 512Kbps video flow

5. Conclusions and Future Work

We presented a simple, yet effective Link Layer scheme to improve the quality of MPEG video flows over wireless links. We discussed implementation issues of the scheme and gave arguments to support its feasibility in last-hop wireless links. We used the Bluetooth technology as the wireless technology to test our scheme, enhancing the L2CAP/HCI layers of Bluetooth with our scheme. Simulation results showed a significant improvement in video quality when using a Bluetooth L2CAP/HCI layer enhanced with our scheme compared to a standard Bluetooth stack. In bad channel links, in particular, the improvement in video quality was appreciable.

The improvements obtained by using this scheme have encouraged our belief that cross-layer optimizations can be useful in wireless environments. The quick response time at the Link Layer make it an ideal place to incorporate such techniques. Moreover, as we argued earlier, such a technique could easily be deployed in wireless base stations (access points) operating at the edge of the network since, unlike core routers, these are not constrained by needing to perform high-speed functions.

Our simulations have shown that quality of video flows can be improved significantly with our scheme. Work is now on to implement this scheme in a Linux Bluetooth testbed. Our testbed uses the Bluez [27] open-source Bluetooth stack and consists of various Bluetooth PCMCIA and Compact Flash cards.

In addition to video, another useful medium that could benefit from such Link Layer support is audio streaming. For example, consider a user with a Bluetooth-enabled MP3 player and headset. If good quality MP3 streaming could be supported over the MP3 player-headset Bluetooth link, it could enable the user to listen to songs without being constrained to be in proximity of the MP3 Player. Work is on to identify techniques that can be used at the Link Layer to enhance support of MP3 streaming audio in an error-prone wireless environment.

References

[1] The Working Group for Wireless LANs, http://grouper.ieee.org/groups/802/11/

[2] Bluetooth Specifications, www.bluetooth.org

[3] Axis Communications - AXIS 9010 Bluetooth Access Point, www.axis.com/products/axis_9010/

[4] Combo Bluetooth-802.11b Access Point, www.pico.net

[5] http://www.mpeg.org/MPEG/index.html

[6] MPEG4 home page, http://www.mpeg4.net/

[7] N. Feamster, H. Balakrishnan, *Packet Loss Recovery for Streaming Video*, in Proceedings of 12th International Packet Video Workshop, April 2002.

[8] Injong Rhee, *Error control techniques for interactive low-bit rate video transmission over the Internet*, SIGCOMM 1998.

[9] W. Tan and A. Zakhor, *Packet Classification Schemes for Streaming MPEG Video over Delay and Loss Differentiated Networks*, Proc. Packet Video Workshop, 2001.

[10] Zhimei Jiang, Leonard Kleinrock, *A Packet Selection Algorithm for Adaptive Transmission of Smoothed Video Over a Wireless Channel*, Journal of Parallel and Distributed Computing.

[11] Video Quality Experts Group, http://www.its.bldrdoc.gov/vqeg/results-psnr.html

[12] J-C. Bolot, T. Turletti, *Adaptive error control for packet video in the Internet*, Proc. ICIP '96.

[13] J-C. Bolot, T. Turletti, *Experience with rate control mechanisms for packet video in the Internet*, Computer Communication Review, January 1998.

[14] M. Normura, T. Fujii, N. Ohta, *Layered Packet- Loss Protection for Variable Rate Coding Using DCT*, Proceedings of InteTnational Workshop on Packet Video, 1988.

[15] S. Wolf, M. Pinson, S. Voran, and A. Webster, *Objective Quality Assessment of Digitally Transmitted Video*, in Proceedings of IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing, 1991.

[16] S. Voran, *The development of objective video quality measures that emulate human perception*, GLOBECOM 1991.

[17] The Video Quality Experts Group, http://www.its.bldrdoc.gov/vqeg/

[18] N.C. Oguz, E. Ayanoglu**,** *Performance analysis of two-level forward error correction for lost cell recovery in ATM networks***,** Infocom 1995.

[19] Andres Albanese, Johannes Blömer, Jeff Edmonds, Michael Luby, Madhu Sudan, *Priority Encoding Transmission*, IEEE Symposium on Foundations of Computer Science 1994.

[20] K. Shinamura, Y. Hayashi, and F. Kishino. *Variable bitrate coding capable of compensating for packet loss*, in Prou. SPIE Cortf Visual Commun. and Image Processing, Nov. 1988.

[21] Video Conferencing Tool, vic, http://www-nrg.ee.lbl.gov/vic/

[22] B. W. Wah, X. Su, and D. Lin. *A survey of error-concealment schemes for real-time audio and video transmissions over the internet*. In Proc. Int'l Symposium on Multimedia Software Engineering, December 2000.

[23] C. Papadopoulos and G. Parulkar, *Retransmission-based error control for continuous media applications*, In Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video, 1996.

[24] Bhaskaran Raman, Pravin Bhagwat, and Srinivasan Seshan, *Arguments for cross-layer optimizations in Bluetooth scatternets*, In Proceedings of Symposium on Applications and the Internet 2001.

[25] Hari Balakrishnan, Srinivasan Seshan, and Randy H. Katz, *Improving reliable transport and handoff performance in cellular wireless networks*, ACM Wireless Networks, December 1995.

[26] Video Traces, Starship Troopers, http://www.acticom.info/1466.html

[27] Bluez Linux Bluetooth Protocol Stack, bluez.sourceforge.net