# Performance Evaluation of UDP Lite for Cellular Video

Amoolya Singh
agni@cs.berkeley.edu

Almudena Konrad
almudena@cs.berkeley.edu

Anthony D. Joseph
adj@cs.berkeley.edu

Computer Science Division
UC Berkeley
Berkeley, CA 94720

## ABSTRACT

The current generation of wireless links pose a significant challenge for sending video streams, as these links have low bit rate and high error rate compared to wired links. Sending high bit rate delay-sensitive traffic over wireless links requires appropriate error resilient video compression algorithms and transport/link layer protocols. We have built a wireless video system using an off-the-shelf error resilient low bit rate video codec with our implementations of UDP Lite and PPP Lite, which are modifications to transport and link layer protocols respectively. The flexible checksumming schemes in these protocols allow error-resilient application policies to be reflected in the networking stack. We conducted simulations and experimental evaluations of this wireless video system using quantitative performance metrics (i.e., throughput, jitter, packet loss, and end-to-end latency).

## 1. INTRODUCTION

Research on delay-sensitive multimedia applications operating in error prone environments (e.g., wireless links) has thus far focused on how to make such applications adaptive or error-resilient in the wired Internet [4, 16, 22, 24, 28, 29], but most of these approaches operate without corresponding support from the transport and link layers. The basic problem is that while bit-error resilient voice and video codecs (e.g., MPEG-4 [18] and H.263+ [11]) can deliver high-quality audio and video even with corrupted data packets, strict checksumming at lower network layers causes corrupted packets to be dropped rather than passed up to the application, nullifying the codec's error resiliency. This problem is especially acute for wireless networks with high packet loss rates.

In this paper, we propose a wireless network architecture with flexible checksumming schemes that support bit-error resilient codecs. We prototyped a "video over wireless" system that sends real-time video streams over a wireless GSM (Global System for Mobile Communications) circuit-switched data interface and measured network performance and video quality. Challenges in implementing this system included the wireless channel's low bit rate (9.6 Kbit/s) and high bit error rate. In addition, the sensitivity of human perception of multimedia [27] imposes tight requirements on jitter, delay, and image quality. Our system uses off-the-shelf error resilient video coding software [11, 18] to address the latter set of challenges. We modified the transport and link layer protocols to address the former challenge, by implementing UDP Lite [12] and PPP Lite, modifications to transport and link layer protocols respectively. We also implemented a socket interface to the application (`RTPsock`), and various monitoring and evaluation tools (`socket-dump`, `MultiTracer` [14]).

The notion of application-layer control over transport-layer protection is not new: UDP Lite [12] is an extension to UDP that allows partial checksums on packet data by enabling applications to specify, on a per-packet basis, how many bytes of the packet (typically including the header) are "sensitive" and must be checksummed. If bit errors occur in the sensitive region, the receiver drops the packet; otherwise it is passed up to the application through a socket interface. This approach allows the application to receive partially corrupted packets rather than dropping them altogether. We added similar functionality to the datalink layer in the form of PPP Lite, by extending PPP [26] to ignore PPP checksums at the receiver — effectively limiting checksumming to that provided by UDP Lite, which can be controlled at the socket interface.

Our simulations and experimental evaluations show lower packet loss, higher throughput, lower end-to-end delay, and constant packet inter-arrival time — all benefits that improve video quality: image fidelity, distortion, and picture "jerkiness."

## 2. BACKGROUND

This section briefly outlines the protocols used in our wireless real-time video transmission system. Readers familiar with error resilient video coding and UDP/PPP Lite may skip to Section 3.

### 2.1 Application Layer

We used two low bitrate, error resilient video codecs: H.263+ and MPEG-4.


*NOSSDAV'01,* June 25-27, 2001, Port Jefferson, New York, USA.

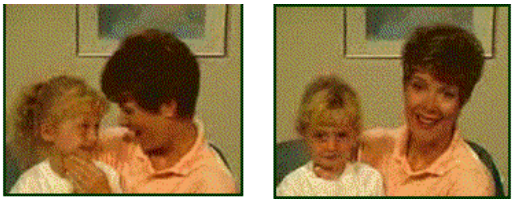


**Figure 1: Sample Screenshots from H.263+ Video Bitstream**

**Figure 2: Sample Screenshots from MPEG-4 Video Bitstream**

### 2.1.1 H.263+ Error Resilient Video Codec

The H.263 video codec standard was developed by the ITU for transmitting video streams at low data rates (< 64 Kbit/s) over the public switched telephone network. It encodes low bit rate video streams using a motion-transform hybrid video coding framework.

The H.263+ standard (1998) improves upon the H.263 standard (1996) in the following ways: it provides enhanced error resilience capabilities (appendices A-F from the standard), offers optional bitstream scalability, and enables better packetization with an underlying protocol such as RTP [5]. The H.263+ standard adds nine new features to the existing suite, including advanced intra coding, reduced block artifacts using a deblocking filter, reference picture selection and resampling, reduced-resolution updates, and modified quantization. The curious reader is referred to the voluminous ITU standard [11] for an exhaustive description of the codec. Two sample screenshots from the *"mom"* bitstream are shown in Figure 1. These screenshots are used as benchmarks to evaluate the performance of our video transmission system, detailed in Section 4.

### 2.1.2 MPEG-4 Video Codec

The MPEG-4 video coding standard was developed by the Moving Pictures Expert Group to code and transmit sub-QCIF video at 5 - 10 Mbits/s. It uses techniques similar to H.263+, with some additional features for error detection and recovery. These include inserting resynchronization markers into the bitstream, partitioning macroblocks within each video packet syntactically, using reversible variable length codes (RVLC) such that data can be decoded in a forward or reverse direction, and using header extension codes (HECs) to optionally repeat important header information describing the video frame. More information can be found at the MPEG-4 website [18] and in Gringeri *et al* [10]. Two sample screen shots from the popular *"carphone"* and *"Suzie"* bistreams are shown in Figure 2.

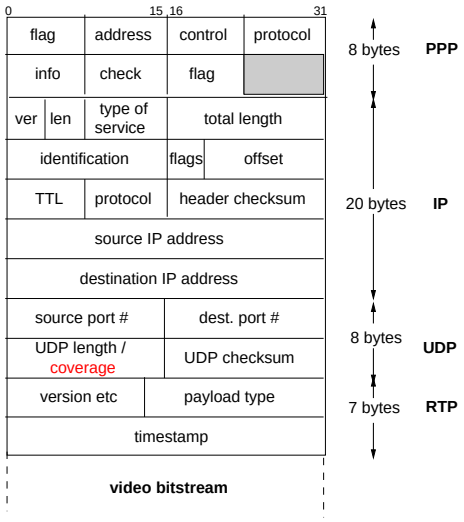



**Figure 3: Packet Header with UDP Lite *coverage* field.**

## 2.2 Transport Layer

### 2.2.1 Real-Time Transport Protocol (RTP).

RTP, the Real-time Transport Protocol [25], provides end-to-end delivery services for real-time data such as interactive audio and video. RTP provides extra information to the application layer in the form of sequence numbers, timestamping, payload type, and delivery monitoring. Depending on the particular real-time application at hand, RTP is either integrated with the application layer or the transport layer (UDP); both RTP and UDP contribute parts of the transport layer functionality.

It is important to note that RTP does not itself ensure timely delivery or other quality-of-service guarantees; it relies on lower-layer services to do so. RTP sequence numbers allow the receiver to reconstruct the sender's packet sequence, but are also used to determine the proper location of a packet (e.g., in video decoding) without necessarily decoding packets in sequence.

**User Datagram Protocol (UDP) and UDP Lite.** UDP is a connection-less unreliable best-effort transport layer protocol. The UDP Lite protocol [12] allows the application to receive corrupted packets instead of dropping them altogether. This is achieved by a partial checksum which only covers a fixed amount of "sensitive" data. Integrating UDP Lite into existing UDP frameworks is simple: the *length* field in the UDP header is replaced by the *coverage* field (see Figure 3), which signifies how many bytes of the packet have been checksummed. With a checksum coverage value replacing the packet length, UDP Lite packets are treated like classic UDP packets with the checksum enabled. To address security concerns and handle the multiplexing of other transport-level flows, the packet header should always be checksummed. If corruption occurs in the sensitive region or in the header, the packet is dropped at the receiver; otherwise the packet is passed up to the application through a socket interface. We used a FreeBSD UDP Lite implementation from Lulea University in Sweden and modified it to run on BSDi.

**Socket Interface: RTPsock.** We implemented a cus-

tom version of a send/receive socket interface in C for several reasons: existing implementations do not have RTP functionality integrated with the socket layer, which we required; most test applications (e.g., Unix `sock`) send random bytes using the socket interface, whereas we wished to send bytes from a video bitstream; and we instrumented the interface to collect statistics (see Section 3). We used standard rate control calculations to determine the socket sending rate for a 9.6 Kbit/s connection.

## 2.3 Link Layer

**Data Link: Point to Point Protocol (PPP) and PPP Lite.** PPP [26] is a data link protocol for point-to-point links. It has two main functions: encapsulation of IP datagrams, and error detection using a polynomial Frame Check Sequence (FCS). The UDP Lite implementation from Lulea University included device driver modifications for implementing a "lite" version of PPP for Unix FreeBSD. We ported these modifications to BSDi adding the ability to simply ignore PPP checksums at the receiver.

### *2.3.1 Radio Link*

The GSM standard allows data traffic in two modes. Non-transparent mode runs a semi-reliable protocol, the Radio Link Protocol, described below. Transparent mode, described in the paragraphs following, simply utilizes the GSM radio link without embedding any additional functionality.

*Radio Link Protocol (RLP).* The GSM Radio Link Protocol [2] is a full-duplex reliable link layer protocol derived from the HDLC logical link layer protocol. Fixed-size data frames of 30 bytes (with 6 bytes of header information) are sent/received every 20 ms; the overall bandwidth is therefore 1200 bytes/s or 9.6 Kbit/s. All frames are strictly aligned to the GSM radio block size used as a basis for channel coding. The protocol performs error recovery through *selective reject* and *checkpoint recovery*.

The ARQ (Automatic Repeat reQuest) mechanism of RLP may introduce delays that are not tolerable when sending video streams. GSM allows us to choose between transparent mode (without ARQ) and non-transparent mode (with ARQ). Turning on RLP increases reliability but also increases delay. On the other hand, turning off RLP decreases delay but also decreases reliability. By experimenting with these two modes, we gain a better understanding of the role of RLP in overall system performance.

*GSM Transparent Mode.* Since transparent mode uses the GSM radio link without any additional ARQ functionality, it provides consistent delay for data transfers. For this reason, we used GSM primarily in transparent mode for our experiments.

# 3. DESIGN OVERVIEW

We present a brief overview of our network stack's layers and the challenges and solutions posed at each layer. Figure 4 illustrates the different parts of our design. Our setup consists of a mobile host communicating with a fixed host (both running UNIX BSDi 3.0) over a GSM [17, 23] circuit-switched connection (either transparent or non- transparent mode) using a mobile phone with a wireless modem. Going end to end, we have RTP using UDP or UDP Lite (UDP Lite always runs over PPP Lite) at the transport layer, IP at the network layer, and PPP at the data link layer. The wireless link from the mobile host to the GSM base station runs in either transparent or non-transparent mode.

At the sender (mobile host), video streams are encoded, using either H.263+ or MPEG-4 low bit rate error-resilient video codecs, at a QCIF resolution of 176 x 144 pixels, bit rate of 10 Kbit/s, and frame rate of 10 Hz, then packetized into RTP packets using a recommended packetization algorithm [30], and finally fed into a socket connection. At the receiver (fixed host), RTP packets are read from the socket, stripped of headers, reassembled into a contiguous stream that is fed to the decoder in real time, and displayed at the receiver and analyzed qualitatively.

The protocol stack with appropriate interfaces is shown in Figure 5.

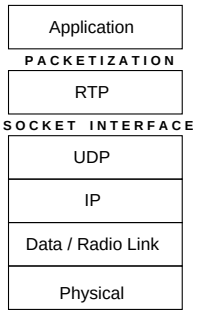



**Figure 5: Network Protocol Stack**

To trace RTP/UDP sender and receiver information, we implemented `socket-dump`, to generate a single log file with timestamps and byte numbers specifying when a packet was placed in the sender's buffer. From the data generated by `socket-dump`, we graphed time/sequence plots and calculated statistics. In particular, we analyzed end-to-end delay, packet inter-arrival time, throughput information, and also some exceptional conditions (e.g., retransmissions, detailed in Section 2.3). We visualized the data generated by the socket interface using `MultiTracer` [14], a set of Perl scripts developed in previous work that converts the trace data into the input format required by plotting tools such as Matlab and `xgraph`.

# 4. PERFORMANCE EVALUATION

We conducted wireless video simulations to isolate the error resilience effects of the H.263+ and MPEG-4 video codecs, as well as to understand the viability of UDP Lite for wireless real-time multimedia streaming. We also ran experiments using the testbed described in Section 3 to collect video traffic and calculate statistics (e.g., end-to-end delay, inter-arrival time, and packet loss).

## 4.1 Channel Simulation

Performance studies of multimedia applications over wireless networks require intense analysis of real video transmissions over an existing network infrastructure. In addition to being very time-consuming, it is oftentimes difficult to isolate specific problems. To understand the impact of high bit error rate wireless channels on error resilience codecs, we designed a wireless simulator, *WSim*, which takes advantage of wireless error traces collected in previous work [13]. A wireless error trace consists of a binary sequence where each

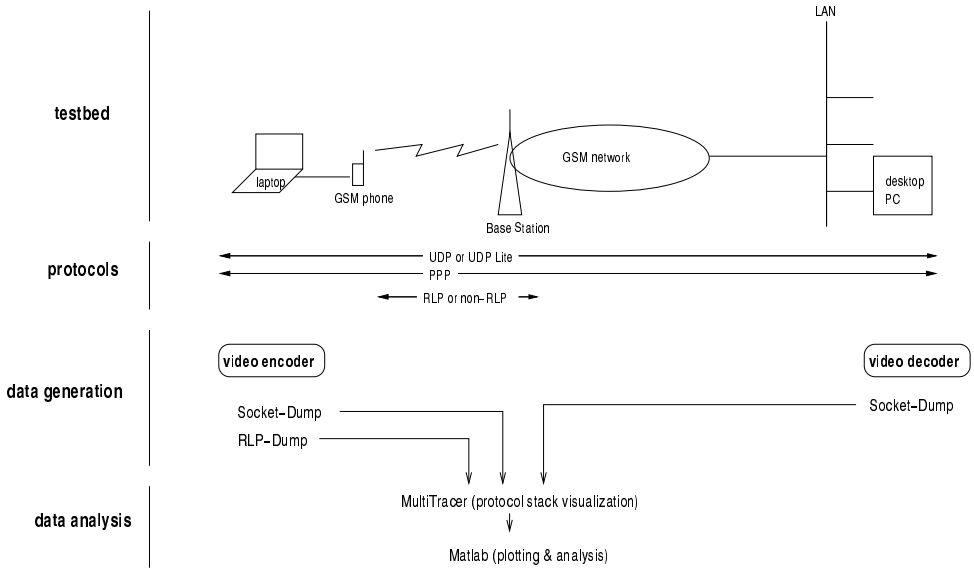



**Figure 4: Cellular IP Testbed**

element represents the state of a radio frame[1] in the GSM wireless channel. There are two states. A *one* represents a corrupted radio frame, while a *zero* represents a correct radio frame. These wireless error traces are specific to the particular FEC and interleaving schemes implemented by GSM circuit-switched data connections. The frame error rate denotes the average rate of corrupted radio frames.

The wireless error traces were collected in different channel environments such as stationary, walking, and driving using the testbed illustrated in Figure 4. Traces collected in stationary environments are either termed *bad* or *good* depending on the signal strength at the mobile phone [13].

*WSim* (see Figure 7) takes as input a video stream, a wireless error trace, and a protocol configuration. The input video stream is fragmented into application data units (ADUs). Figure 6 shows the ADU encapsulated in a PPP frame. In our simulation, the PPP frame consists of 512 bytes of ADU, 7 bytes of RTP header, 8 bytes of UDP or UDP Lite header, 20 bytes of IP header, and 7 bytes of PPP header. *WSim* implements the radio link in transparent mode. Furthermore, the PPP frame is fragmented into 24-byte radio frame, where each data byte takes up 1 'start' bit, 8 data bits and 1 'stop' bit.

Our simulator generates an output video stream with corruption according to the error occurring at the radio frames. However, we did not know the distribution of errors *within* a radio frame. Therefore, we chose to randomly corrupt 20% of the data bytes in the radio frame. The algorithm for *WSim* is presented in Figure 8.

*WSim* is based on 215 minutes of traces collected in a *bad* stationary environment. We simulate two protocol configurations, UDP and UDP Lite, running over the radio link in transparent mode. Our goal was to compare the two protocol configurations and determine which one offers better video quality. We ran *WSim* on the *"mom"* video stream using a wireless trace with error rate 1.58%. Although we had traces with frame error rates ranging from 0.5% to 4%, we chose 1.58% because this high error rate is difficult to obtain in a real measurement, yet not so high that it causes the video codec to crash from unrecoverable corruption in the bitstream. Figure 9 shows representative screenshots of H.263+ video streams produced by *WSim* for both UDP and UDP Lite. Observe that the video quality with UDP is worse than with UDP Lite owing to the fact that UDP drops corrupted packets while UDP Lite transmits them. For the sake of brevity, we are omitting additional results from the MPEG-4 video streams since they are qualitatively quite similar.

[1]In current circuit-switched GSM systems, a radio frame contains 30 bytes.

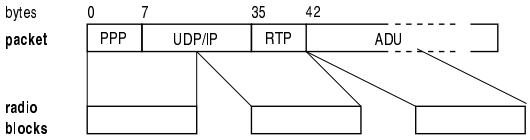



**Figure 6: Radio Frame Fragmentation**

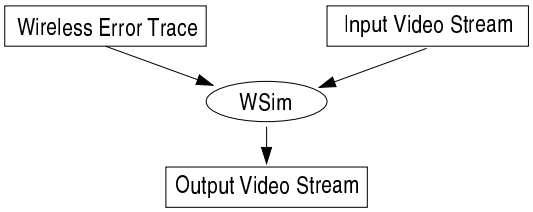



**Figure 7: *WSim* block diagram**

## 4.2 Experimental Results

We ran video over wireless experiments for three combinations of transport and radio link protocols: UDP with the radio link in non-transparent mode; UDP and UDP Lite

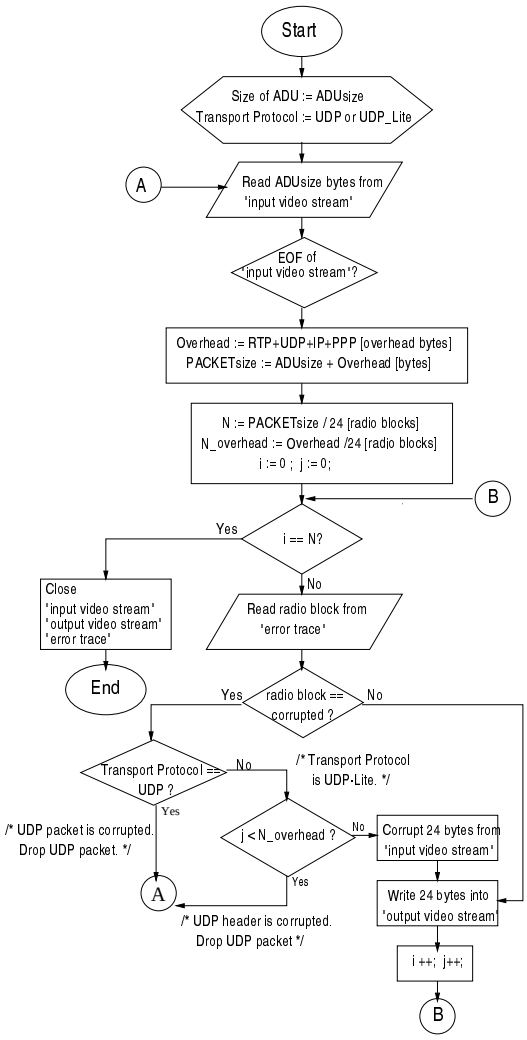



**Figure 8: *WSim* Algorithm.**

with transparent mode.

UDP offers a connection-less best-effort datagram service with strict error detection in the transport layer, while UDP Lite offers the same service with flexible error detection which enables corrupted data packets to be passed up to the application layer. The radio link in non-transparent mode provides reliability at the radio link layer at the cost of end-to-end delay.

We collected a total of 4480 minutes of wireless video traces, corresponding to 1120 four minute video transmission runs. We repeated each protocol combination for ten runs of good and bad channel conditions. Thus, each data point presented below corresponds to 40 minutes of transmission. Good channel conditions correspond to a signal strength between -93dB and -87dB, while bad channel conditions correspond to a signal strength between -105dB and -99dB. For each run, we collected traffic information at the transport and radio link layers and chose to use these statistics to compute four key metrics: end-to-end delay, inter-arrival time, % packet loss, and throughput. The choice is based on the effects of end-to-end delay, inter-arrival time,

UDP UDP Lite

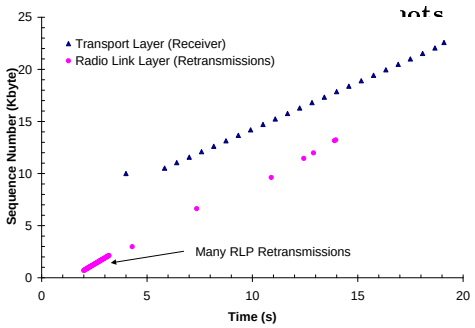



**Figure 10: RLP Impact on Delay**

and packet loss on perceived quality of real-time video. High delay causes a long lag between when an image is sent and when it is received. High variation in packet inter-arrival times results in "shaky" or jerky video playback. Finally, packet loss causes image blurring or severe distortion.

The mean value for each metric and protocol combination is plotted as a single point, with Y error bars denoting standard deviation. We also include screenshots for quality comparisons between UDP and UDP Lite in transparent mode.

### 4.2.1 *Delay*

Figure 10 shows the impact of RLP retransmissions on end-to-end delay for streaming video over UDP, with non-transparent mode. Using `MultiTracer`, we plot sequence numbers for packets at the receiver's transport layer along with any retransmissions at the radio link layer. Notice that the first two received packets experience larger end-to-end delay due to radio link layer retransmissions.

The end-to-end delay is greatest for UDP non-transparent at 1.73 seconds, while for UDP, transparent and UDP Lite, transparent it is 0.51 and 0.38 seconds respectively. Figure 11 shows the delay of the connection from end to end for all three protocol configurations. Notice that the mean delay for the transparent connection, for both UDP and UDP Lite, is smaller than the mean delay for the non-transparent connection. These results are as expected: a reliable link-layer protocol like RLP introduces end-to-end delay due to retransmissions at the link layer, even though the overlying transport protocol (UDP or UDP Lite) is only best-effort.

### 4.2.2 *Inter-Arrival Time*

The packet inter-arrival time is plotted in Figure 12. The main result in this graph is that packet inter-arrival time is remarkably constant across all combinations of transport and link-layer protocols (0.60, 0.59, 0.60 for UDP, non-transparent; UDP, transparent; and UDP Lite, transparent respectively). We observed that the range of inter-arrival times is slightly larger for non-transparent (introduced by retransmissions)

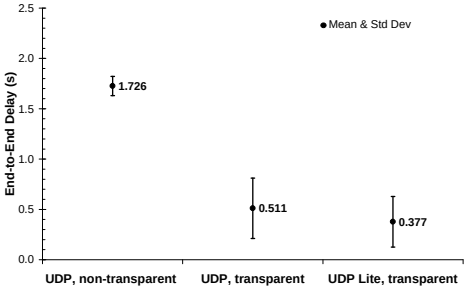



**Figure 11: End-to-End Delay (seconds)**

than for transparent mode. There are also small differences in the results' standard deviations that are likely caused by slight variations in the wireless channel environment rather than by any significant protocol effects.

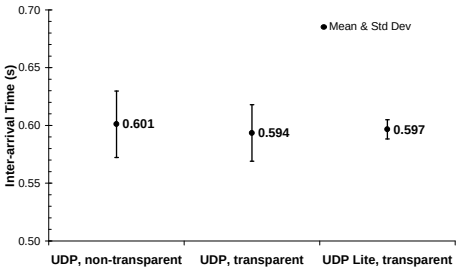



**Figure 12: Packet Inter-Arrival Time (seconds). Note that the Y-axis does not begin at zero.**

### 4.2.3 *Packet Loss*

Packet loss is plotted as a percentage of packet drops in Figure 13. These results are simple to understand. Packet loss is 0% for the non-transparent connection, because RLP uses ARQ to retransmit corrupted data. Loss is slightly higher for transparent connections because neither the link or transport layers provides error detection. As expected, packet loss is much lower for UDP Lite than UDP (1.1% versus 2.1%), because UDP Lite ignores errors in data blocks.

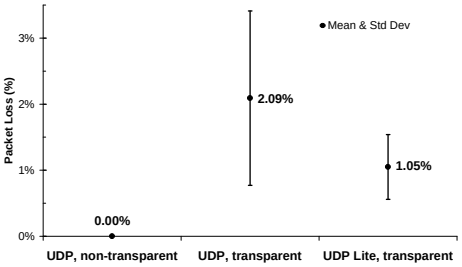



**Figure 13: Packet Loss (%)**

### 4.2.4 *Throughput*

Throughput is a useful overall performance metric as it indicates both end-to-end delay and packet loss. We measured throughput as the amount of data passed up to an application (corrupted or intact), divided by the total connection time. Figure 14 shows the throughput obtained for the three different protocol configurations. Since packet loss is significantly lower for non-transparent than for transparent mode, we expect to see lower throughput for transparent mode connections (6.73 Kbit/s for UDP and 6.79 for UDP Lite) versus the non-transparent mode connection (6.83 Kbit/s). The difference in throughput between UDP and UDP Lite is explained by UDP's higher packet loss than UDP Lite, which in turns drives its throughput down.

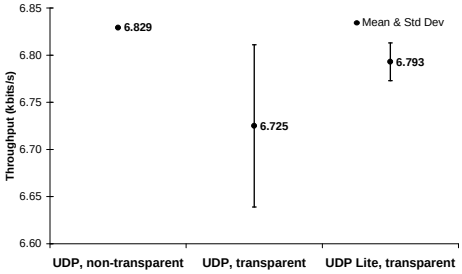



**Figure 14: Throughput (Kbit/s). Note that the Y-axis does not begin at zero.**

### 4.2.5 *Perceived Video Quality*

The pictures in Figures 15 and 16 are from the H.263+ and MPEG-4 video bitstream sent over UDP and UDP Lite in transparent mode. We omit transmissions in non-transparent mode since the high end-to-end delay caused by ARQ cannot be depicted in static screenshots. These figures show UDP Lite's advantages: notice the distortion near the girl's ear in the UDP picture due to packet losses. The MPEG-4 screenshots show similar problems with UDP. Ongoing work involves conducting a qualitative analysis of video based on the Mean Opinion Scoreand other methodologies.

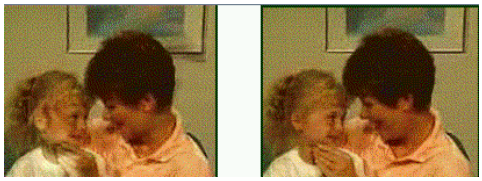
## 4.3 Discussion

In summary, UDP Lite provides 26% less end-to-end delay, constant inter-arrival time, slightly higher (1%) throughput, 50% less packet loss, and significantly better video quality than UDP, when both are run over a transparent radio link. These results allow us to comment on what protocol combinations may be used for particular applications. For example, a delay-sensitive application that does not have stringent error-free requirements might perform better using UDP Lite and the low delay characteristics of transparent mode. On the other hand, an interactive error-intolerant application would perform better using the high reliability of the non-transparent mode. It should be noted that our approach is applicable strictly to UDP flows running over a transparent radio link. In the event of competing TCP and UDP flows expecting reliable transmission, Ludwig and Raythoni's technique of interleaving flows at the radio link layer [15] can be applied since it ensures differing levels of reliability from end to end.

These tradeoffs are summarized in the table below.

| *Application Type* | *Example* | *Protocol Choice* |
|---|---|---|
| intolerant & rigid* | batch *email, ftp* | TCP/RLP |
| | interactive *telnet, web* | UDP/RLP |
| tolerant & adaptive* | hard real-time `wb` | UDP/RLP |
| | adaptive real-time `vic, vat` | UDP Lite/ non-RLP |

*Adapted from [7].



UDP UDP Lite

**Figure 15: Experimental Screenshots: H.263+ "mom"**

## 5. RELATED WORK

Several efforts related to this research exist both in industry and academia. Here we will outline the most prominent and attempt to compare and contrast them with our work.

PacketVideo [21] and ActiveSky [1] Corporations both develop software for streaming video to mobile devices wirelessly. First demonstrated in November 1999, PacketVideo's patented video codec plays MPEG-4 video streams on the application-specific integrated circuits used in mobile phones.

NEC Corp. [19] in September 1999 announced a prototype of a two-piece third generation cellular video handset which transmits images and sound simultaneously. The phone and viewer are connected using Bluetooth short distance radio technology while the video is compressed using the MPEG-4 standard. NTT DoCoMo [9] has also produced a prototype video phone similar to NEC's. However, these prototypes are aimed at the W-CDMA (wideband code division multiple access) network system, supporting data rates of up to 384 Kbit/s. The high bandwidths distinguish NEC's and NTT's products from our research.

Researchers in Sweden have profiled the performance of multimedia streams over network nodes running UDP Lite in the kernel [12]. Their simulations of real-time network traffic (with and without partial checksums on the transport and link layers) show that the decision to turn checksumming on or off must be a function of the particular network environment rather than a function of the particular multimedia stream. This decision concurs with our results.

Wendi Heinzelman has developed unequal error protection schemes for MPEG-4 [18] video which adapts the level of correction across a packet to the importance of the corresponding bits [6]. Her work is distinguished from ours in that her application-specific communication protocols for wireless networks focus mainly on energy-efficient cluster-based routing and media access.

At UC Berkeley, Avideh Zakhor's group has developed hierarchical FEC, subband coding, and other video compression techniques in the context of their Matching Pursuits [20] video codec. They are also examining TCP-based bandwidth-scalable techniques.

**Figure 16: Experimental Screenshots: MPEG-4 "carphone"**

Finally, the Comet group at Columbia University [8] is examining adaptive mobile networking issues through *Mobiware*, a CORBA- and Java-based networking middleware toolkit which adapts to time-varying network conditions. Although the thrust of their work sounds quite similar to the ideas presented here, the primary difference is in (important) details: all their experiments are based on local-area (2 Mb/s WaveLAN) and ATM networks, and not the wide-area IP-based Internet. The extremely high bandwidth of their air interface implies a significantly different set of research problems than ours.

## 6. CONCLUSION

In this paper, we explored techniques for improving multimedia content delivery over error-prone wireless links. In doing so, we have demonstrated the significant benefits of application-level adaptation schemes coupled with network-level protection schemes that allow applications to determine what data they consider "sensitive" to corruption or loss over aggressive network-level protection of multimedia traffic. Through simulation of the wireless channel and experiments transferring live video over a radio interface, we showed how the use of flexible checksumming schemes, such as UDP Lite and PPP Lite, yields markedly less delay, less packet loss, and higher throughput than a traditional protocol stack, while keeping the jitter and video quality constant. In addition, implementing these protocols is not difficult: UDP Lite involved changes to only a few hundred lines of kernel code, and at the user level, it is possible to invoke UDP Lite on a per-socket basis. As video coding technology, such as MPEG-4, improves and wireless packet data networks, such as GPRS [3] are deployed, multimedia application developers will be able to build rate-adaptive systems that utilize the radio backchannel for network-level feedback. This simplistic, but powerful networking approach, when combined with the appropriate low bit rate, error-resilient video codecs, will enable truly error-resilient mobile multimedia communication over wireless links.

## 7. FUTURE WORK

There are several possible extensions to this work. At the application layer, one might incorporate unequal protection schemes which would work in concert with UDP Lite.

At the socket layer, a better alternative would be to provide a more dynamic feedback mechanism between the application and link layers. For example, by providing real-

time feedback on radio channel conditions, the encoding application can adapt its transmission rate and encoding accordingly. A preliminary implementation of such a feedback algorithm has just been completed, and is under submission.

In the wireless simulator, we plan to collect byte-level wireless error traces (in place of the packet-level error traces) which we will feed into *WSim* as the channel coding model.

## Acknowledgments

We would like to thank Lars-Ake Larzon and Mikael Degermark for providing us with an UDP Lite implementation for Unix FreeBSD-3.0. Special thanks to Keith Sklower for implementing UDP Lite and PPP Lite in BSDi3.0 UNIX, and for all manner of help with systems programming and kernel debugging. We also thank Randy Katz, Mark Handley, and Larry Rowe for guidance and feedback. Reiner Ludwig, Bela Raythoni and Stephan Baucke of Ericsson were a ready wealth of ideas and technical knowledge.

## 8. REFERENCES

[1] *ActiveSky Corp.* `http://www.activesky.com/`.

[2] E. G. T. S. 04.22. GSM Radio Link Protocol for data and telematic services, Dec 1995.

[3] C. Bettstetter, H.-J. Vögel, and J. Eberspächer. GSM Phase 2+ General Packet Radio Service GPRS: Architecture, Protocols, and Air Interface. *IEEE Communications Surveys*, 2(3):2–14, 1999.

[4] J.-C. Bolot, T. Turletti, and I. Wakeman. Scalable Feedback Control for Multicast Video Distribution in the Internet. In *Proceedings of ACM SIGCOMM*, 1994.

[5] C. Bormann, L. Cline, G. Deisher, T. Gardos, C. Maciocco, D. Newell, J. Ott, G. SUllivan, S. Wenger, and C. Zhu. RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H.263+). *RFC 2429*, Oct 1998.

[6] M. Budagavi, W. R. Heinzelman, J. Webb, and R. Talluri. Wireless MPEG-4 Video Communication on DSP Chips. *IEEE Signal Processing Magazine*, January 2000.

[7] D. Clark, S. Shenker, and L. Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. In *Proceedings of ACM SIGCOMM*, 1992.

[8] Comet Group. Columbia University. `http://comet.columbia.edu/mobiware/overview.html`.

[9] M. T. *et al.* DoCoMo's Exhibits for Telecom '99. *NTT DoCoMo Technical Journal*, 1(3):32–40, Mar 2000.

[10] S. Gringeri, R. Egorov, K. Shuaib, A. Lewis, and B. Basch. Robust Compression and Transmission of MPEG-4 Video. In *Proceedings of ACM Multimedia*, 1999.

[11] I.-T. R. H.263. Video Coding for Low Bit Rate Communication, Feb 1998.

[12] L. Larzon, M. Degermark, and S. Pink. UDP Lite for Real-Time Multimedia Applications. *Proceedings of the Sixth IEEE International Workshop on Mobile Multimedia Communications*, 1999.

[13] R. Ludwig, A. Konrad, and A. Joseph. Optimizing the End-to-End Performance of Reliable Wireless Links. In *Proceedings of ACM Mobicom*, 1999.

[14] R. Ludwig, B. Rathonyi, A. Konrad, K. Oden, and A. Joseph. Multi-Layer Tracing of TCP over a Reliable Wireless Link. In *Proceedings of ACM SIGMETRICS*, 1999.

[15] R. Ludwig and B. Raythoni. Link Layer Enhancements for TCP/IP over GSM. In *Proceedings of IEEE Infocom*, 1999.

[16] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-Driven Layered Multicast. In *Proceedings of ACM SIGCOMM*, 1996.

[17] M. Mouly and M. B. Pautet. *The GSM System for Mobile Communications*. Cell and Sys, France 1992.

[18] MPEG4 Overview. `http://drogo.cselt.stet.it/mpeg/standards/mpeg-4/mpeg-4.htm`.

[19] NEC Corp., Japan. `http://www.nec.com/`.

[20] R. Neff and A. Zakhor. Very low bit rate video coding based on matching pursuits. *IEEE Transactions on Circuits and Systems for Video Technology*, 7.1:158–171, 1997.

[21] PacketVideo Corp., San Diego, CA. `http://www.packetvideo.com/`.

[22] R. Puri, K. Ramchandran, and A. Ortega. Joint source channel coding with hybrid FEC/ARQ for buffer constrained video transmission. In *Proceedings of Multimedia Signal Processing*, 1998.

[23] M. Rahnema. Overview of the GSM System and Protocol Architecture. *IEEE Communications Magazine*, pages 92–100, April 1993.

[24] R. Rejaie, M. Handley, and D. Estrin. Quality Adaptation for Congestion Controlled Video Playback over the Internet. In *Proceedings of ACM SIGCOMM*, 1999.

[25] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *RFC 1889*, Dec 1995.

[26] W. Simpson. The point-to-point protocol. *RFC 1661*, Jul 1994.

[27] R. Steinmetz and C. Engler. Human perception of media synchronization. Technical Report 43.9310, IBM European Networking Center, Oct. 1993.

[28] Y. Wang and Q.-F. Zhu. Error Control and Concealment for Video Communication: A Review. *Proceedings of the IEEE*, 86(5):974–997, May 1998.

[29] Q. Zhang and S. A. Kassam. Hybrid ARQ with Selective Combining for Video Transmission over Wireless Channels. In *Proceedings of IEEE International Conference on Image Processing*, 1997.

[30] C. Zhu. RTP Payload Format for H.263 Video Streams. *RFC 2190*, Sep 1997.