CS 641 Project Report

Error resilient video transmission over wireless networks

December 2002

Gang Ding

Abstract

This report proposes a general architecture for error resilient video transmission over mobile wireless networks. Radio link layer, UDP Lite, and error control layer are combined to deal with high error rate in wireless environments. Both sender and receiver algorithms are given. An adaptive algorithm is further presented to automatically adjust parity data length in error control. The performance is analyzed by both theory and simulation results. Future works are also suggested.

1 Introduction

The traditional cellular mobile wireless networks are mainly used for low-rate audio communications. However, new generation of cellular networks is emerging to transfer data traffic at much higher bit-rate and to merge to rapidly growing data networks (e.g. internet). This motivates the demand for multimedia (especially video) communication over wireless networks. The major challenges of multimedia transmission over wireless networks are that wireless links have lower bandwidths, higher transmission error rates (typically time-varying and bursty), in contrast to wired networks. On the other hand, real-time video usually requires much more bandwidth and less response time than text, audio or static image. The stringent quality of service (QoS) requirements of video and the unreliability of wireless links combine to make delivering video over wireless networks a notoriously difficult problem.

In this term project, I am going to address the above problem and propose an error resilient architecture for video transmission over wireless networks, which involves modified radio link layer error control, modified UDP (UDP Lite), and a general frame for error control in application layer. The main objective of these techniques is to combine to overcome the error-prone nature of wireless links. There are actually a lot of research papers addressing such kind of problems. And many new ideas for video source coding, error control coding and network protocols have been proposed. In this project, however, I am not trying to look into a particular technique because currently there is still no single technique which can solve the above problem in any circumstances. Instead, I am more interested in (1) the cooperation among different layers and (2) the performance analysis. Based on an idea that "the higher layer, the more intelligent", we resort to the application layer to coordinate the error control, while lower layers try to provide as much information as possible to the upper layer to make decisions.

This report is organized as follows. Next section will introduce necessary background knowledge. Section 3 will explain the proposed architecture in detail. Then theoretical performance analysis is conducted in section 4. Some simulation experiments are introduced in section 5. Section 6 concludes the report and suggests future works.

2 Background

2.1 Commercial perspective

Due to the great success of mobile cellular phones all around the world, the future global 3G mobile networks are expected to support more and more video applications. However, due to

the delay of deploying of 3G wireless networks. It is still hard to say when the video over wireless will become widely accepted, hopefully as popular as current audio cellular phone. There are actually some commercial products available, for instance, the packet video Corporation (see [9]) develops software for streaming video to mobile devices wirelessly. Packet video's patented video codec plays MPEG-4 video streams in mobile phones. But the quality of current products is still not good enough. This well motivates further research on this difficult topic.

The circuit-switch based cellular network is actually only one of wireless networks. In the recent years, the packet-switch based wireless LAN has become commercially available, mainly motivated by the standardization of IEEE 802.11. A lot of cooperate or campus WLANs have come into being. In this September, ViXS Systems Inc. ([6]) just announced that the industry's first video networking processor "XCode" is sampling to early customers, enabling products to transmit broadcast quality video reliably across wireless LANs. However, the wireless link in WLAN also suffers the unreliability problem which deserves more research efforts on video transmission over wireless IP-based networks. For more details, please refer to [9].

2.2 Wireless link

Wireless link is the lowest layer of a wireless network. No matter it is a cellular network, or a WLAN or Personal Area Network (PAN), the error-prone nature of wireless link is common. Although channel coding is used to protect the transmitted data, there is still high error rate in wireless links in contrast to wired links. MAC and physical layers are usually standardized by some standardization organization, like ISO, CCITT and IEEE. We will not look into the details of these standards. What we are interested in is how to attenuate (ideally remove) the effect of these lower layer errors on the upper layer applications. We will go though all the network layers above MAC and physical (PHY) layers from bottom to the top.

Right above MAC and in the same link layer, there is usually a sub-layer protocol, Radio Link Protocol (RLP), which is used to fight for transmission error. The data stream from upper layers is partitioned into multiple RLP units. The size of each unit depends on the MAC and PHY layers. In third-generation (3G) wireless systems, for example, the typical size of a radio unit is from 300 to 600 bits. A unit also contains a 16-bit CRC header in order to detect some error bits in the unit. For GSM, a popular 2G wireless system, a radio unit involves 24 bytes data and 6 bytes header. RLP applies a kind of ARQ to recovery errors. Instead of sending ACK for every unit, the receiver only sends NACK to the sender when it detects error in the received data unit. The sender adopts limited retransmission, which means the link will be reset after several times of unsuccessful retransmission. RLP provides a mechanism for error control at lower layer, but it can not guarantee a complete reliable transmission like TCP. Another drawback of RLP is that it introduces time delay because of ARQ. A RLP has another working mode, called transparent mode, when no error control mechanism is applied. Transparent mode is often employed by time-sensitive applications, such as video streaming, where some data errors may be acceptable. Most previous works ([1], [2]) simply use transparent mode and employ upper layer error control techniques to overcome transmission error. But in this case, the error recovery function of RLP is not fully used.

2.3 Network protocols

Some researchers have investigated the cooperation of RLP and TCP. TCP is a very important network protocol which provides complete reliable transmission of data. However, for most video communications in wireless and wired networks, the application can tolerate data errors to some extent. So another transport protocol, UDP, is widely used for video transmission. UDP is a connection-less, unreliable best-effort transport protocol, which means UDP may lose packets. But UDP has a checksum to verify the integrity of received packet. Therefore, it can detect any error in the packet header or payload. In contrast, wireless networks are characterized as low-bandwidth and unreliable, in which a considerable amount of packet losses are induced by both channel failure and network congestion. Since UDP does not perform any error recovery, it may sacrifice the whole packet only for some minor data errors, which can yield unpredictable degradation and poor video quality

In order to solve the above problem with UDP, a modified version, called UDP Lite is introduced in [12], which is an extension to UDP and allows partial checksums on packet data by enabling application layer to specify, on a per-packet basis, how many bytes of the packet 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 layer. This approach allows the application (e.g. Video decoding) to receive partially corrupted packets which will generate acceptable video quality. Integrating UDP Lite into existing UDP frameworks is quite simple: the length field in the UDP header is replaced by the coverage field, which signifies how many bytes of the packet have been checksummed.

Between UDP and application layer, there might also be a Real-time Transport Protocol (RTP), which provides extra information to application layer in the form of sequence numbers, timestamping, payload type, and delivery monitoring. But RTP itself does not ensure timely delivery or other Quality-of-Service (QoS) guarantees.

2.4 Error control

Error control is employed to reduce the effect of transmission error on applications. Two basic approaches are Forward Error Correction (FEC) (or more generally error control coding) and Automatic Repeat Request (ARQ). FEC adds parity data to the transmitted packets and this redundancy is used by the receiver to detect and correct errors. FEC maintains constant throughput and has bounded time delay. ARQ only provides error detection capability by requesting retransmission when errors are detected by the receiver. ARQ is simple but the delay is variable and unbounded. Many alternatives to FEC and ARQ have also been proposed ([7]).

One of the most popular error coding techniques is Reed-Solomon coding, which can deal with burst error. To our interest in binary data transmission, the field of RS coding is of the form $GF(2^M)$, where M is any positive integer. If the original message length is K, then after adding parity data, the codeword is of length N > K. The basic parameters of RS code are:

- Codeword length: $N = 2^M - 1$
- Number of check symbols: $N - K = 2 * T$

Which means that the redundant codeword of length N can recover errors of length at most $T = (N - K) / 2$. For example, if we choose M=8, then each symbol is one byte, N=255. Then

for any given data of length K < 255 bytes, we add parity bytes in order to get a codeword of length 255 bytes, then the receiver can recover the original K bytes as far as the corrupted data length is not greater than (255-K)/2 bytes.

2.5 Video source coding

ITU-T H.263 recommendation is the first standard to offer a solution for very low bit-rate (<64 Kbit/s) teleconferencing applications. The recently adopted H.263+ improves coding efficiency of H.263 in many ways. It provides enhanced error resilience capabilities, offers optional bitstream scalability, and enables better packetization with an underlying protocol such as RTP. H263+ 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.

Similarly, the recently adopted ISO MEPG-4 standard is robust in error-prone environments, which is achieved by inserting resynchronization markers into the bitstream, partitioning macroblocks within each video packet syntactically, using header extension code (HEC) to optionally repeat important header information describing the video frame, and using reversible variable-length coding (RVLC) such that data can be decoded in a forward or reverse direction.

Many novel error resilient video codec's are kept being invented. But I will not introduce the details and I prefer to use the standard codec.

3 An error resilient video transmission architecture for wireless networks

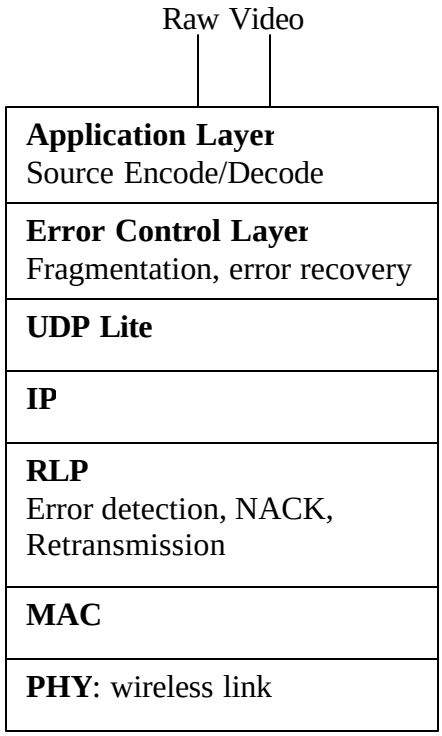


Figure 1. Protocol stack

I here propose a general protocol stack for video transmission over wireless links. See Figure 1. The algorithms for sender and receiver are given as follows.

Sender's Algorithm:

1. *At application layer, when the raw video comes, appropriate source codec is applied to get encoded video bitstream. We hereby don't specify a particular source coding algorithm, but an error resilient algorithm is preferred.*
2. *At error control layer, the encoded video data is fragmented to a fixed size of M2=255 bytes, including partial data of size M3 bytes. Then the actual data length is M2-M3.*
3. *Add UDP Lite header where checksum only covers the header.*
4. *Add IP header with IP checksum.*
5. *At RLP layer, fragment packet to equal length radio units and add CRC for error detection. Set timer for sent unit and retransmit the unit if timeout or NACK is received from the receiver. If still gets NACK after several times of retransmission, reset the link.*
6. *Send radio units through MAC and wireless link to receiver.*

Receiver's algorithm:

1. *At RLP layer, detect error of received packet units, assemble units and send up to IP layer. If error is detected, send a NACK to sender and set timer for retransmission of NACK.*
2. *At IP layer, calculate checksum of IP header, if checksum error, discard the packet, otherwise forward it up to UDP layer.*
3. *At UDP layer, calculate checksum of UDP header, if checksum error, discard the packet, otherwise forward it up to error control layer.*
4. *At error control layer, apply error coding algorithm to correct errors, assemble fragments to a complete video frame and buffer it until its time is up and retrieved by application layer. Notice that if transmission error size at receiver is not greater than the error control capacity, for example $M_3/2$ for RS codes, the entire real video packet can be recovered. Otherwise, signal RLP layer to send one more NACK immediately. So we still have a chance to receive the correct data from retransmission before playing. But when this frame's time is up, before forwarding it to application layer, make sure to clear all corresponding timers in the RLP layer because we do not need the data any more. We refer to the source decoding algorithm in application layer to deal with remaining errors and the received video quality might be degraded.*
5. *At application layer, decode the received video packet for playing.*

Followings are several comments to the above algorithm

First, by error control coding, i.e. adding extra parity bytes of size M3, at least part of errors can be recovered by receiver. Obviously, the larger M3 is, the more errors will be corrected. But on the other hand, parity data introduces more traffic to the limited network bandwidth and may even cause packet loss due to congestion. So M3 should not be large. Hense how to choose an appropriate M3 to trade-off the error correction and network traffic should be considered carefully. In the above algorithms, we assume the partial byte size M3 is given and fixed. A better way is to adapt M3 based on the available information of networks and errors. To this end, we further introduce a M3 adaptation algorithm as follows.

Recall that in step 4 of receiver's algorithm above, the receiver sends one more NACK back to the sender when it can not correct all errors. So At the sender's side, this second NACK can be regarded as an indication of failure of error correction. So upon receiving the second NACK, sender increased M3 in order to improve the error correction capacity:

$$M3 = M3\_initial + 2^k$$

Where M3_initial is the initial value for M3 which can be arbitrarily chosen. k is increased by 1 for each following NACK.

For regular case when there is no second NACK, however, sender keeps deceasing M3 in order to reduce the extra traffic to the network by

$$M3 = M3 - M3step$$

Where M3step is the decreasing step.

The above adaptation algorithm can help us to choose an appropriate parity data size M3. But it may also introduce oscillation of M3. So we can run this adaptation algorithm at the beginning, after certain amount of time, we can use the average value of M3 as a constant for M3 and turn off adaptation algorithm. Adaptation can be started whenever necessary.

Second, when designing the above algorithm, we are following an important idea borrowed from intelligent control theory: *the higher layer, the more intelligent*. For example,

- The error control layer between UDP and application is responsible for recovering errors, adjusting M3, requesting to transmit NACK and clearing timers at RLP layer.
- But at RLP layer, it only conducts error detection and semi-ARQ. Its timer is cleared by signals from error control layer.
- At both BLP and UDP layer, data is forward to the upper layer even if there is some error in the payload, they just simply resort to the upper layers to make decision.

The reason to follow the above idea is that at higher layers, there is more information collected from below so that better decisions can be made. For the video transmission over wireless network, the final decision is made by the highest layer: the human being who watches the video. So what the lower layers do is just try to collect as much as information as possible instead of discarding it.

Third, we are trying to make the proposed algorithm general, for instance,
Instead of thinking of one particular wireless network, we only take the common characteristic of all wireless networks: unreliability. In this perspective, we take the wireless link as just an error generator which may generate different kinds of errors, such as constant rate error, variable rate error, burst error, etc. So the above algorithm applies to any kind of error-prone wireless networks.
We also do not specify a particular error control codes, we just use the common function of all blocking error codes: using parity data to correct errors.
We do not specify a particular video source coding algorithm. But we prefer to use and investigate standardized codec which has been tested for times.

4 Performance Analysis

In order to analyze the above algorithm in theory, however, we should first specify the error model for wireless link. There is actually no all-recognized model for the underlying wireless link, mainly due the highly time-varying and non-stationary nature of wireless networks. It involves both fast channel fading and slow channel fading, as well as the mobility pattern, the location of the mobile node, and so on. But in all cases, the wireless link can just be modeled as an error generator. In our analysis, for the sake of convenience, we just assume the bit error probability is given by p_b. Then we will give theoretical expressions for the error probability and efficiency at RLP, UDP and error control layer, respectively.

4.1 RLP layer analysis

Since the bit error probability is p_b, it is easy to find out the RLP radio unit error probability is

$$p_R=1-(1-p_b)^{M1+H1} \sim (M1+H1)p_b$$

Due to retransmission, the actual error probability of a radio unit is probably less that the above result.

We also define a transmission efficiency parameter for radio unit as the ratio of times transmitting a radio unit without loss and with loss-and-retransmission.

$$g_R = \frac{(M1+H1)}{\sum_{n=1}^{n_{\max}-1}(nM1+(2n-1)H1)P_{n0} + (n_{\max}M1+(2n_{\max}-1)H1)(1-\sum_{k=1}^{n_{\max}-1}p_{n0})}$$

where n_{max} is the maximal time to retransmit before reset, $P_{n0} = p_R^{n-1}(1-p_R)$. We assume the transmission rate is constant during retransmission and the size of an NACK packet is just the header length.

4.2 UDP layer analysis

Similarly, for UDP, considering both the header and payload, error probability is

$$p_{UDP} = 1-(1-p_R)^{\frac{M2+H2}{M1}}$$

efficiency is

$$g_{UDP} = \frac{(M_2+H_2)}{\sum_{n=1}^{\infty}[(\frac{M_2+H_2}{M_1}(M_1+H_1)+(n-1)(M_1+H_1)]P_n}$$

where $P_n = p_{UDP}^{n-1}(1-p_{UDP})$ means probability of *n-1* UDP packets get errors before a success (re)transmission. We assume only one radio unit in a UDP packet is retransmitted.

For UDP Lite, only considering UDP header, error probability is

$$p_{UDPLite} = 1-(1-p_R)^{\frac{H2}{M1}}$$

and efficiency is the same as that of UDP.

$$g_{UDPLite} = g_{UDP}$$

4.3 Error Control layer analysis

If there is no error control layer above UDP layer, all error data is forwarded to the application layer, where the packet error rate is

$$p_{APP1} = 1-(1-p_b)^{\frac{M2+H2}{M1}(M1+H1)} \approx \sum_{n=1}^{M_2-1}\binom{M_2}{n}p_b^n(1-p_b)^{M_2-n}$$

When error control is added, it can recovery M2-M3 bytes error out of a packet of size M2. In this case, the error probability is represented as

$$p_{APP2} = \sum_{n=1}^{M_2-M_3-1}\binom{M_2}{n}p_b^n(1-p_b)^{M_2-n}$$

So we can easily know that error probability is decreased when error control is added. But we should notice that in this case, the actual throughput is also decreased because after adding error control, M2-M3 data instead of M2 data is transmitted every time.
For a video packet of size M2, the efficiency is represented as follows.

$$g_{APP} = \frac{M_2}{\sum_{n=1}^{\infty}[(\frac{M_2+H_2}{M_1}(M_1+H_1)+(n-1)(M_1+H_1)]P_n}$$

5 Simulation

I simulated the proposed algorithm and tested it by a benchmark MPEG-4 trace. The video trace file comes from the movie "Jurassic Park" of 60 minutes long and is publicly available

for the test of algorithm's network performance, especially for wireless networks ([10], [11]). Some important parameters for the video are as follows.

- Resolution: QCIF 176*144
- Frame rate: 25 frames/sec
- Frame sequence: IBBPBBPBBPBB
- Compression ratio: YUV:MP4 49.96
- Video run time: 3.6e+6 msec
- Min frame size: 26 bytes
- Max frame size: 8154 bytes

In the simulation, we choose our parameters M1 = 50, M2 = 255, H1 = 2, H2 = 28. Due to the fact that the simulation of long video is quite time-consuming, we only use the first 1080 frames of the trace video for our simulation.

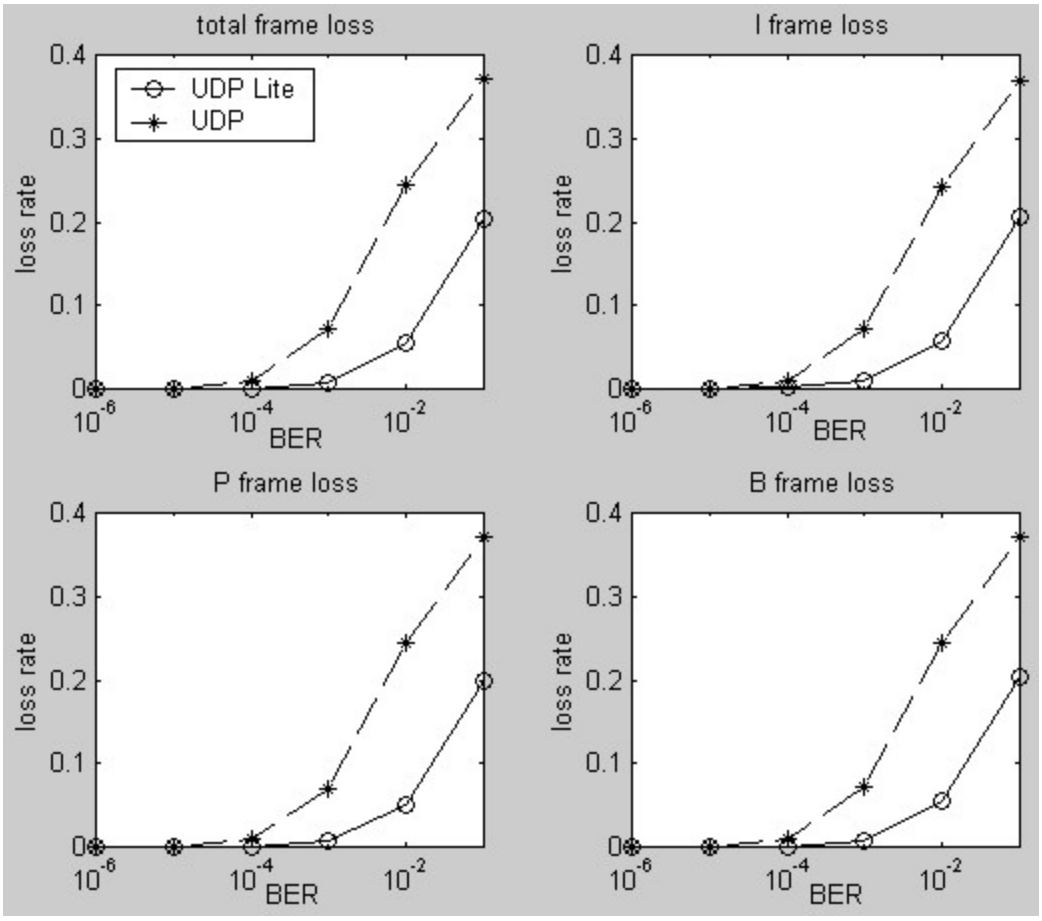
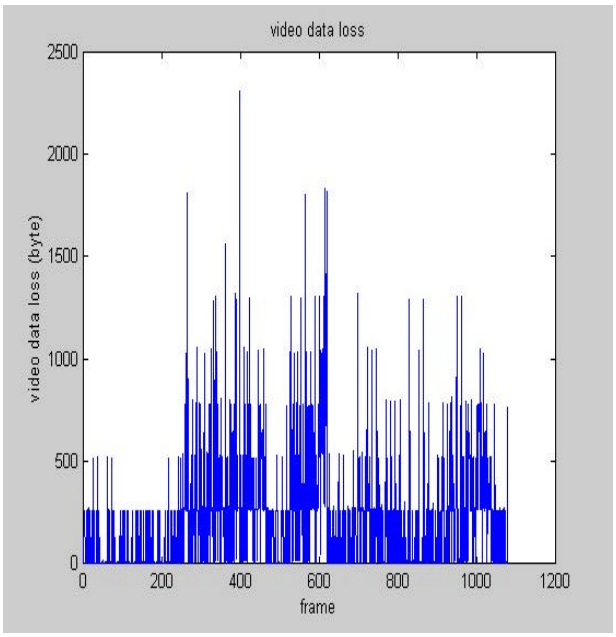


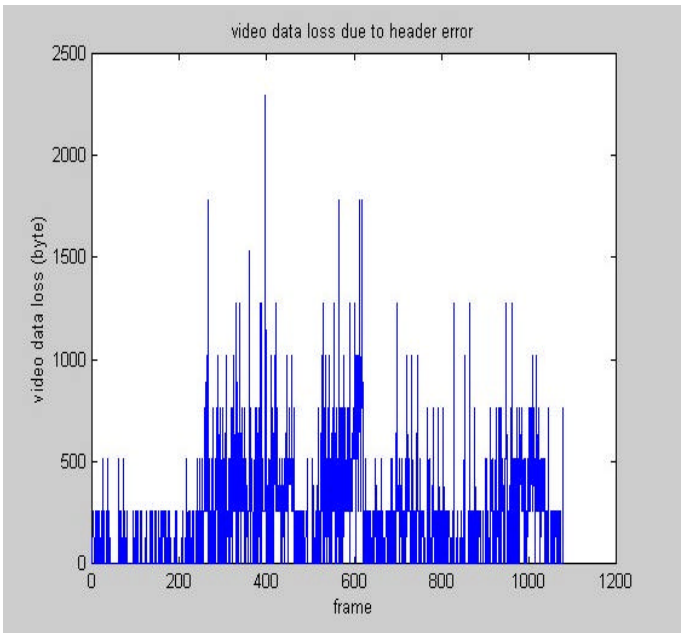
Figure 2. Video frame loss rate (UDP: dash, UDP Lite: solid)

5.1 UDP vs. UDP Lite

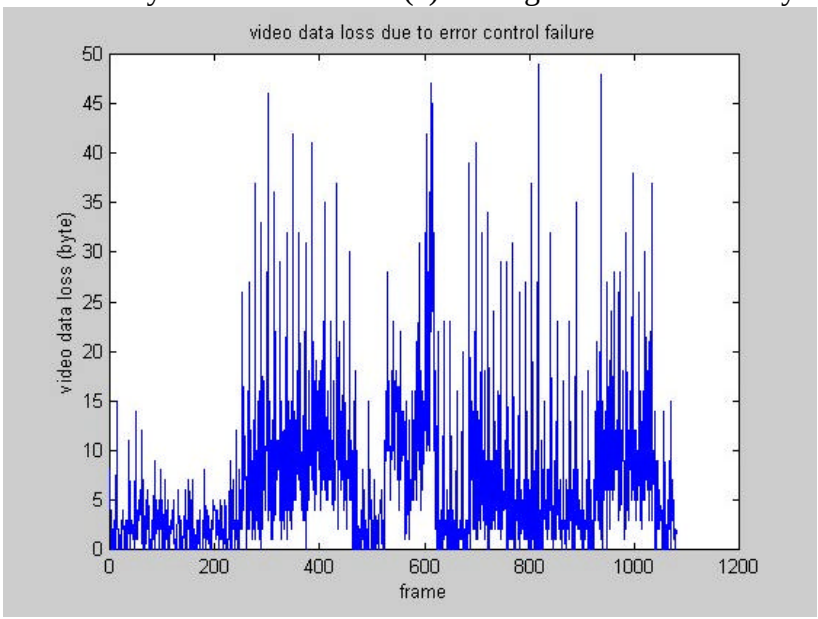
We first compare packet loss rate for UDP and UDP Lite. The results are show in Figure 2, where total frame loss, I frame loss, P frame loss and B frame loss are all displayed for Bit Error Rate (BER) p_b from 10^{-6} to 10^{-1}. It is obvious that UDP Lite (solid) performs much better than UDP(dash). This is actually also proved by the theoretical expressions for p_{UDP} and $p_{UDPLite}$ in last section.



(a) Average data loss = 225.7 bytes/frame



(b) Average data loss = 219 bytes/frame



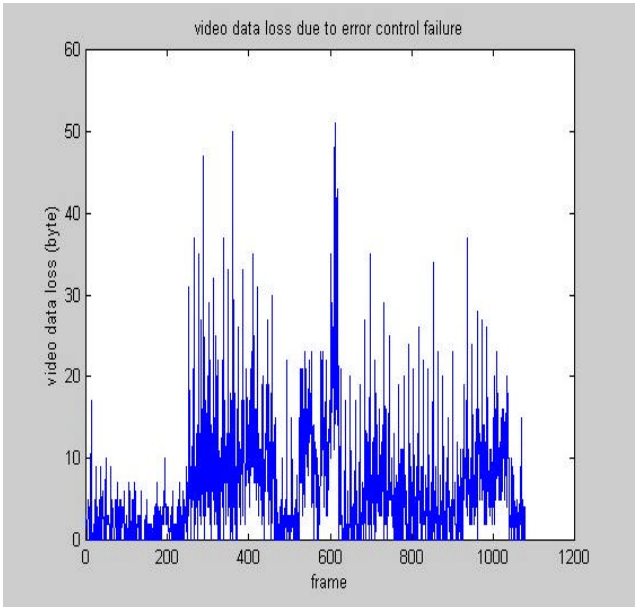
(c) Average data loss = 6.7 bytes/frame

Figure 3. Video data loss without error control (BER=0.01)

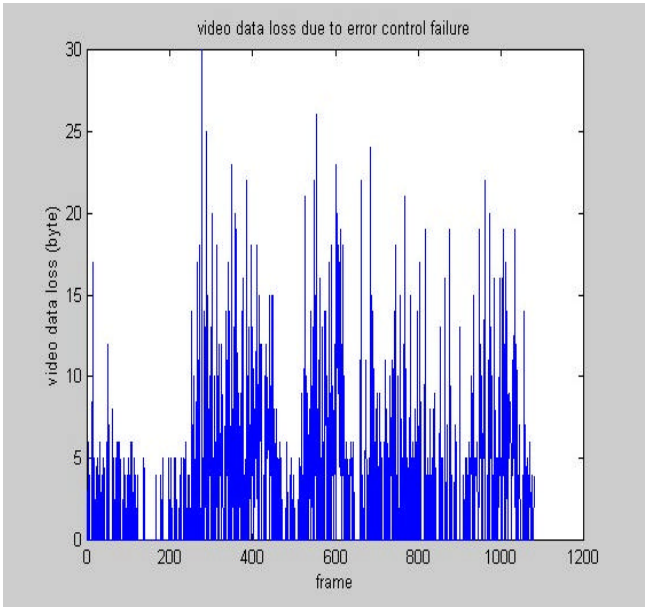
5.2 Error control

We also simulated the effect of error control. First, when there is no error control, i.e. M3 = 0, Figure 3 shows video data loss for each frame. Figure 3(a) is the total data loss. Figure 3(b) is part of the data loss due to checksum error of IP header or UDP header. Another part of data loss due to payload data error is show in Figure 3(c). What we are most interested in is the payload data error which can be recovered by error control. So the following figures only show this kind of data error. Notice that we choose BER p_b = 0.01, which might be higher than reality in order to show the significant results. Also we didn't simulate retransmission in

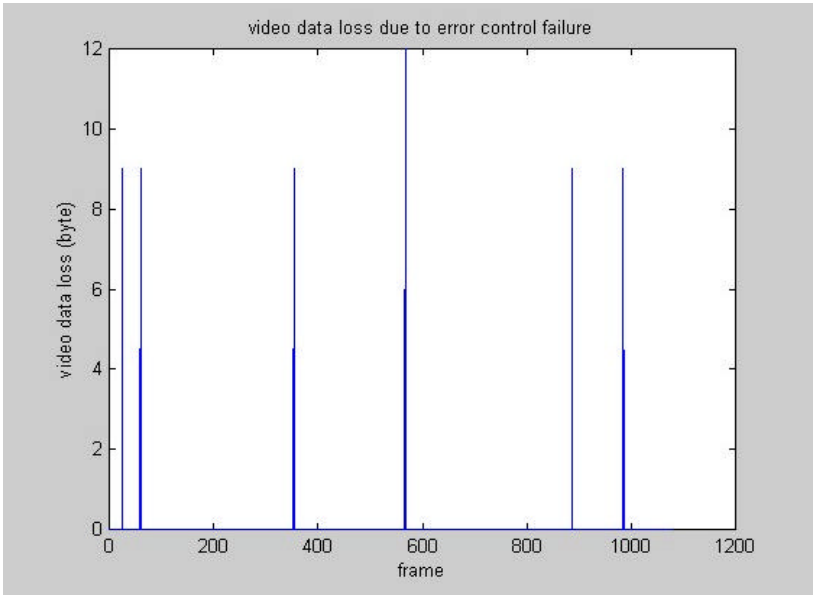
RLP layer, the actual packet loss due to checksum error will be largely reduced when retransmission is taken into consideration.



(a) M3=2, Average data loss = 6.3 bytes/frame



(b) M3=6, Average data loss = 3.3 bytes/frame

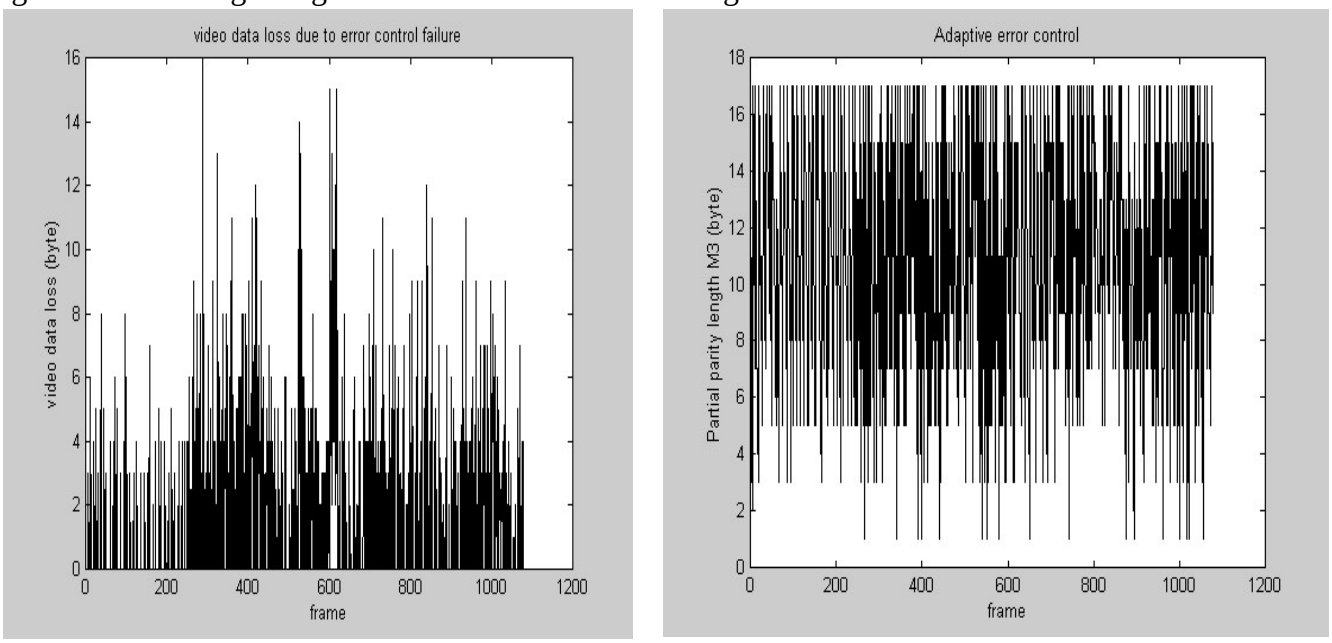


(c) M3 = 16, Average data loss = 0.053 bytes/frame

Figure 4. Video data loss with error control (BER=0.01)

To test error control, we choose different parity data size as M3 = 2, 6, 16 bytes, respectively. The data errors are shown in Figure 4. It is evident that when we increase parity data size, more error can be recovered by error control coding. However, the disadvantage of adding parity data is that more redundant data will be transmitted though the network. As discussed in Section 3, we further add parity data size adaptation algorithm. In this case, we just choose an arbitrary initialize size as 16 bytes, with M3step = 2 bytes, M3min = 0 bytes, M3max = 32 bytes. What we finally get is an average M3 = 10.8 bytes, with average data loss 1.67

bytes/frame. The simulation results are displayed in Figure 5, where the dynamic of M3 is also shown in Figure 5(b). We should notice that M3 keeps changing rapidly, which may not do good to the system performance. So as suggested in Section 3, we only need adaptive M3 algorithm at the beginning and turns it off when it converges.



(a) Average data loss = 1.67 byte/frame (b) M3_initial = 16 bytes, M3_average=10.8 bytes

Figure 5 video data loss with adaptive parity data length

6 Conclusion and Future works

In this report, we proposed a video transmission approach over wireless networks. The RLP and error control layer combine to process error data. Data error in UDP payload is ignored by using UDP Lite. Theoretical performance analysis is given and simulation results show the effectiveness of the proposed algorithm. Some works are suggested for future study.

- Other error models and their effect on system performance.
- Further simulations for the retransmission mechanism in RLP layer.
- Try the proposed algorithm to some real video data.
- Further investigate video transmission over both internet and wireless networks.

References

[1] A. Singh. et al. “Performance evaluation of UDP Lite for cellular video”. NOSSDAV’01, 2001.

[2] H. Zheng and J. Boyce. “An improved UDP protocol for video transmission over internet-to-wireless networks”. IEEE Trans. on Multimedia. 3(3): 356-365. 2001.

[3] H. Liu and M. Zarki. “Adaptive source rate control for real-time wireless video transmission”. Mobile Networks and Applications. 3: 49-60. 1998.

[4] S. Worrall. “Optimal packetization of MPEG-4 using RTP over mobile networks”. IEE Proceedings-Commun. 148(4):197-201. 2001.

[5] A. Majumdar, et al. “Multicast and unicast real-time video streaming over wireless LANs”. IEEE Trans. on Circuits and Systems for Video Tech. 12(6):524-534. 2002.
[6] http://www.vixs.com
[7] H. Liu, et al., “Error control schemes for networks: an overview,” Mobile Networks and Applications, 2, pp. 167-182, 1997.
[8] M. Zarki, “Video over IP”, Infocom 2001.\
[9] G. Ding, “Review of video transmission over mobile wireless networks,” CS641 course midterm, 2002.
[10] MPEG-4 and H.263 Video Traces for Network Performance Evaluation. http://www-tkn.ee.tu-berlin.de/research/trace/trace.html
[11] F. Fitzek and M. Reisslein, "MPEG--4 and H.263 Video Traces for Network Performance Evaluation," IEEE Network, 15(6): 40-54. 2001.
[12] L Larzon, et al, “The UDP Lite Protocol,” internet draft, Jan. 2002.