Lost Packets Recovery during Video Broadcasting Over Data Network

Hassan H. Soliman, Hossam E. Mostafa and Eman A.E. Ahmed

Department of Electronics and Communicatios Engineering, Faculty of Engineering
El-Mansoura University, Mansoura, Egypt.

hhsoliman@hotmail.com, hossam_moustafa@mans.edu.eg, engineer_em@hotmail.com

Abstract

RTP (Real-time Transport Protocol)\UDP (User datagram protocol) are used in conjunction extensively in communication and entertainment systems that involve streaming media. Streaming media refers to the delivery method of the media rather than the media itself. The disadvantage of using RTP \ UDP is that it does not guarantee any reliability or in order delivery of the packets. Assume applying some steganography methods to transmit data covertly via RTP header and RTP payload in multimedia streaming over networks. Some packets can be lost during video broadcasting over network and others can be delivered out of order. The stego video that will be broadcasted on the network will be MJPEG (Motion Joint Photographic Experts Group) video that is stored on Server has a known IP (Internet Protocol) address and when the client requests this video; the client and the Server will communicate using the Real-Time Streaming Protocol (RTSP) and send data using the RTP. The challenge in this paper is how to retrieve hidden data by applying the three proposed methods used to retrieve these lost packets. The first method is Forward Error Correction (FEC), which adds the FEC RTP parity packet to the original packets. The second method is Interleaving which divides each packet into four parts, each will be added in one consecutive four packets. The third method is Receiver-Based Repair (Interpolation) which bases on the assumption that there is a small difference between two neighbouring packets, which is true especially in case of video. The video stream will be traced using the Ethereal program. The proposed methods were tested and compared according to the retrieval percentage of the hidden data at the receiver.

Keywords: *Real-time Transport protocol, Covert channels, Lost Packets Recovery.*

1. Introduction

Multimedia streaming becomes the major traffic over networks and huge volume of data makes it an ideal candidate for covert communications. There is an increasing demand for streaming video applications on the Internet; various network characteristics make the deployment of these applications more challenging than traditional TCP-based applications like email and the Web. Transmitting high-quality, real-time interactive video over loss networks such as the Internet and wireless networks is very challenging. Because of limited bandwidth on networks and the bandwidth-hungry nature of video, video transmission requires extremely high compression efficiency. In this paper, we describe three proposed methods used to recover lost MJPEG video frames, which will be broadcasted over network assuming end-to-end communication and using RTP/UDP for transport layer. In multimedia, Motion JPEG (M-JPEG or MJPEG) is a video format in which each video frame or interlaced field of a digital video sequence is compressed separately as a JPEG image. Assuming that, the frames of video will include hidden data in the RTP header (using the sequence number field) and in the RTP payload. By using un-reliable protocol as UDP for data transmission, there will be some lost packets so, what if these packets contain hidden secret data. The challenge in this situation is how to recover the dropped packets to retrieve the hidden data. The rest of the paper is organized as the following: **Section 2** explains in brief the Literature survey of Lost Packet Recovery for streaming video, **Section 3**: explains the proposed methods, **Section 4**: gives experimental results and its discussion and **Section 5**: concludes the paper.

2. Related Work

Most of the earlier work on lost packet recovery focuses on repairing packet losses before the scheduled display time of those video frames that are contained in the lost packets (e.g., [1, 2, 3, 4, 5, 6, 7, 8]). However, these approaches are ineffective for interactive video because data losses occur inevitably in packet-switched communication, detecting and repairing losses incur latency. To handle this latency, existing techniques introduce additional delays in frame display time. However, delaying frame play out time impairs greatly the interaction of video communication. Many researchers [9, 10, 11, 12] have proposed using retransmission of lost packets by delaying frame playout time to allow the arrival of retransmitted packets before the display time of their video frames. Any packets that are received after their display time will be discarded. In these schemes, the display time of a frame is delayed by at least three one-way trip times after its initial transmission (two for frame transmission and one for a retransmission request). This latency can significantly impair the interactivity of any video applications on the current Internet. Forward error correction is also commonly proposed for error recovery of continuous media transmission [13, 14, 15, 16, 17]. However, conventional FEC schemes do not work well for interactive video. Therefore, unless the playout time of a frame is delayed, both the original packets and their parity packets must be transmitted within the same frame interval, rendering the schemes very susceptible to burst loss. Moreover, since FEC is applied to a block of packets, before FEC packets are computed and transmitted, large delay must transpire.

In earlier work [18, 19], an entirely complementary approach proposed to the above-mentioned approaches by focusing on eliminating error propagation when distortion of display images happens. The point of departure from existing approaches was that packets do not have to arrive in time for them to be “useful” for display of that video frame. Of course, if packets can arrive before the display time of their frames, that is optimal. However, due to packet losses and high latency, repair packets inevitably arrive “late”, causing distortion in displaying images, which starts to propagate to following frames. These late repair packets can be used to stop error propagation. In motion-compensated codecs, the correct display of a frame depends on the successful reception of all of its reference frames. If we buffer displayed frames and use late packets to restore errors in the buffered frames, error propagation can be stopped. This is because the buffered frames will be used as reference frames for later frames. This approach named Recovery from Error Spread using Continuous Updates (RESCU). RESCU has been shown effective for interactive video transmission when retransmission is used to recover lost packets and round trip delays are small [18, 19]. Retransmission tends to prolong error propagation because of the delay involved in detecting and retransmitting lost packets. Moreover, in some networks such as wireless cable modems and direct satellites, feedback channels are highly contentious, and bandwidth-limited. Thus, in these networks, frequent transmission of feedback to the sender is too expensive. In publications [18, 19], they presented how retransmission and layered coding can be used along with RESCU. Retransmission is the most commonly used error recovery technique in reliable transport.

3. The proposed methods for lost packet recovery

The RTP payload format for JPEG video stream explained in [20], [21] and [22]. According to using RTP for transmission, sometimes some packets were lost. A packet is lost if [23]:

- A packet never arrived
- A packet arrived, but, corrupted (IP and UDP checksum tests failed)
- A packet arrived after its scheduled play out time.

The receiver uses the 16-bit SEQ (sequence number) to detect the packet loss and to restore the packet sequence. The sequence number is incremented by one for each RTP data packet. RTP provides no guarantee of delivery, but the presence of SEQs makes it possible to detect missing packets. According to RFC 3550 [22], the initial value of SEQ should be random to make known-plain text attacks on encryption more difficult. The sequence number of the first RTP packet can be utilized for covert communication.
One of the following methods can recover lost packets:

- Forward Error Correction (FEC).
- Interleaving.
- Receiver-Based Repair.

3.1 Forward Error Correction (FEC).

Generally FEC is based on adding redundancy to transmit data this can be applied by Using Parity (FEC) packets. Figure 1 illustrates this method.

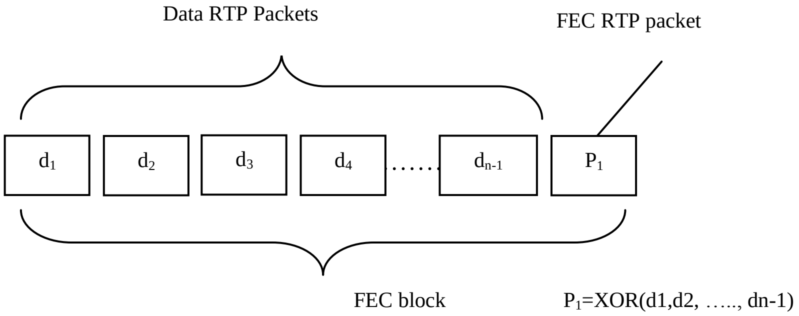


Figure 1: Parity (FEC) packet.

If any of packet gets lost, it can be recovered. For example, if d3 is lost, then
d_3=XOR(d_1, d_2, .. ……, d_{n-1},p_1) , where n-1 is the number of data RTP packets.

This method can be extended to several parity FEC packets, but the additional FEC packets increase the overhead, i.e. increase the required network bandwidth (small FEC block causes large overhead). Large FEC blocks cause delays at the receiver (the receiver needs to wait for all packets in the FEC block in order to be able to do the packet recovery). In this paper, one FEC RTP packet was used.

3.2 Interleaving

The FEC-based methods for data recovery increase the required bandwidth and latency. An alternative method, which does not increase the bandwidth, is interleaving. The video frame (one packet) length is 100 milliseconds. Figure 2 shows The JPEG Packets.

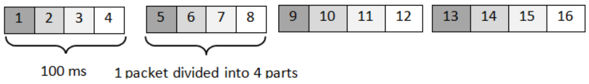


Figure 2: The JPEG packets.

The transmission stream is generated by rearranging the video units within RTP packets: Figure 3 illustrates a block of interleaved JPEG packets.

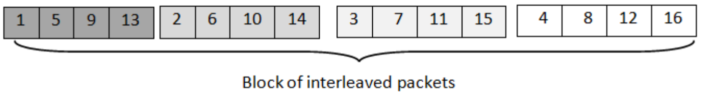


Figure 3: Block of interleaved packets.

The FEC-based methods for data recovery increase the required bandwidth and latency. An alternative method, which does not increase the bandwidth, is interleaving.

Figure 4 shows Packets received by the receiver and Figure 5 shows reconstructed RTP payload after re-sequencing.

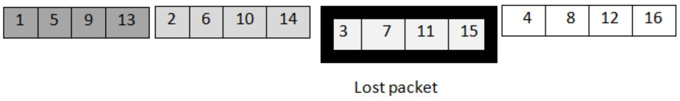


Figure 4: JPEG packets received by the receiver.

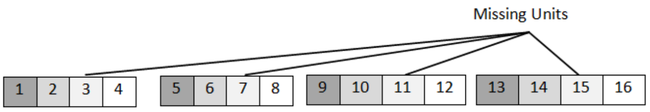


Figure 5: Reconstructed RTP payload after re-sequencing.

Lost packet causes small (25 ms) gaps in the restored video stream. These gaps cannot be noticed. A loss of several packets will cause larger or more frequent gaps, which decreases the reception quality gracefully.

This method does not increase the bandwidth, but increase delays, which are proportional to the length of interleaving blocks. This limits the use in interactive applications such as internet phone. This is good for streaming stored video.

3.3 Receiver-Based Repair

This method does not increase bandwidth requirements or delays. Based under the assumption that there is a small difference between two neighbouring packets that is especially true in the case of video. Figure 6 illustrates this method.

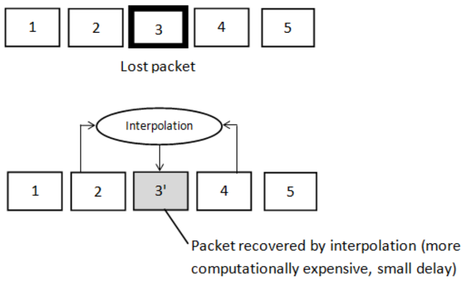

1
2
3
4
5
Lost packet
Interpolation
1
2
3'
4
5
Packet recovered by interpolation (more computationally expensive, small delay)

Figure 6: Receiver-Based Repair method.

Assume the lost frame (Lost Packet) has the order LP. Interpolation, in this work, is applied by calculating the average between the values of the pixels in the previous frame and the values of the pixels in the later frame so; each lost pixel in the lost frame can be calculated as described in (1)

$$\mathbf{f_{LP}\,(x_i,y_j)= (f_{LP-1}(x_i,y_j)+ f_{LP+1}(x_i,y_j)) / 2} \qquad \mathbf{(1)}$$

Where, i, j are the index of the pixel in the frame, f_{LP} (x_i,y_j) is the interpolated value, $f_{LP-1}(x_i,y_j)$ is the value of the same pixel in the previous frame and $f_{LP+1}(x_i,y_j)$ is the value of the same pixel in the later frame. The maximum value for i, j is 256. Interpolation method cannot be used to recover the first or the last packet.

4. Results and Discussion

4.1 Database Description

The Proposed methods were implemented using Java Netbeans IDE 7.4. They were tested on 50 different MJPEG video files, the duration of each video was 50 seconds and the frame rate was 10 frames/sec. The resolution of each video was 256*256. Each stego video stored on Server and was ready to be broadcasted according to the client's request. Ethereal program used to capture the video stream at the receiver side.

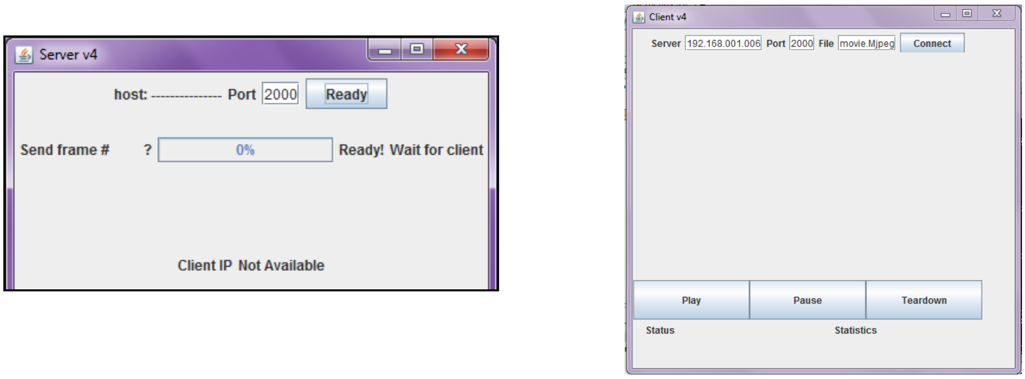
4.2 Hidden Data

The secret messages for embedding were text or image files. The text files used for embedding were Text1, Text2 and Text3. Text1 was (hidden data) the size of it was 12 bytes and Text2 is (data hiding is a very interesting field) the size of this text file was 37 bytes and Text3 was (data hiding is a very interesting field we can hide any data) the size of this text file was 58 bytes. The image files that were used for embedding were (Image1.jpg (1.26Kbyte) 35*37, Image2.jpg (1.9 Kbyte) 45*45 and Image3.jpg (2.44Kbyte) 50*50). The hidden data will be embedded in both JPEG Payload + 2 initial bytes of the sequence number. The relation between packet dropping and the hidden data retrieval percentage (sample video1 used) described in (2).

The hidden data retrieval percentage = (number of hidden retrieved bits / total number of bits of the hidden message) *100 % (2)

4.3 Follow up the stream of video during broadcasting

After preparing the stego video and locating it in the specified server, the port number should be set. Now the stego video is ready for any request from any client as shown in Figure 7 (a).


Server v4

host: -------------- Port 2000

Send frame # ? 0% Ready! Wait for client

Client IP Not Available

Client v4

Server 192.168.001.006 Port 2000 File movie.Mjpeg

Status

Statistics

(a) **(b)**

Figure 7: (a) Server waits for the client request, (b) Client can see the stego video if the play button was clicked.

When the client asks for the stego video, the following entries must be entered:

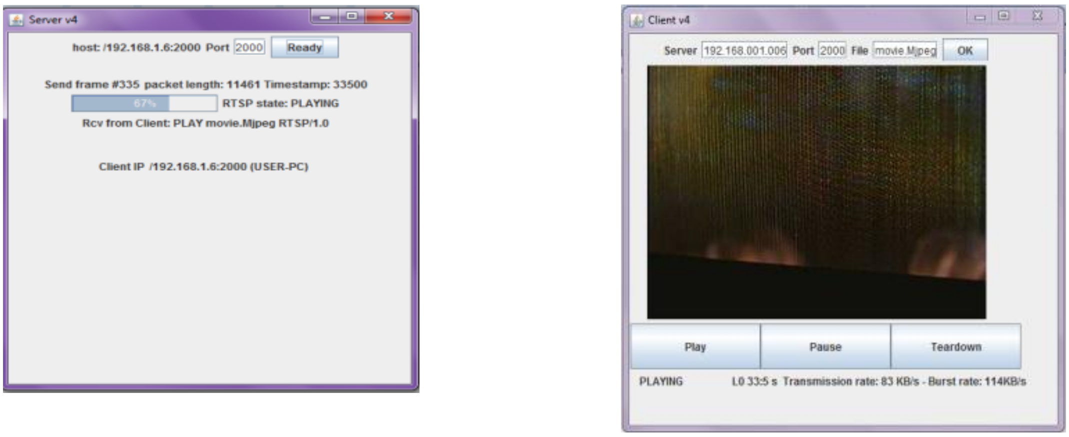
→ The IP for the server or the server name

→ The stored stego video name

→ The same port number set on the server then press connect.

Now the video is ready for playing if the client presses the play button as shown in figure 7 (b).

After the client presses the Play button, he can see the stego video and can pause or teardown the session. Figure 8 (a) shows the server and Figure 8 (b) shows the client during playing the stego video.


Server v4

host: /192.168.1.6:2000 Port 2000

Send frame #335 packet length: 11461 Timestamp: 33500

67% RTSP state: PLAYING

Rcv from Client: PLAY movie.Mjpeg RTSP/1.0

Client IP /192.168.1.6:2000 (USER-PC)

Client v4

Server 192.168.001.006 Port 2000 File movie.Mjpeg

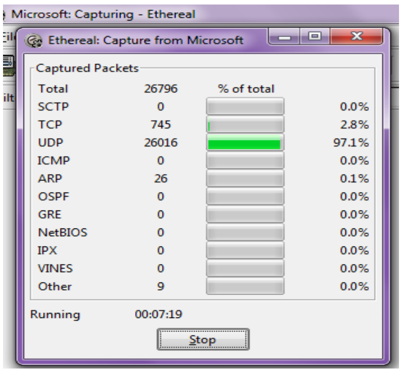
PLAYING L0 33:5 s Transmission rate: 83 KB/s - Burst rate: 114KB/s

(a) **(b)**

Figure (8): (a) The server, (b) The client during the stego video playing

Where, Transmission rate= (cumulative size of the video stream / packet timestamp) KB/s, Burst rate = (payload length / time interval between consecutive packets) KB/s and the Number of lost packets (L) increases if the timestamp of the packet received is greater than expected.

Ethereal program used to follow up the video stream. Figure 9 (a) shows The UDP/RTP captured packets during MJPEG video transmission. Figure 9 (b) shows the payload of one frame captured using the Ethereal program. By analysis this payload, the secret data can be retrieved.



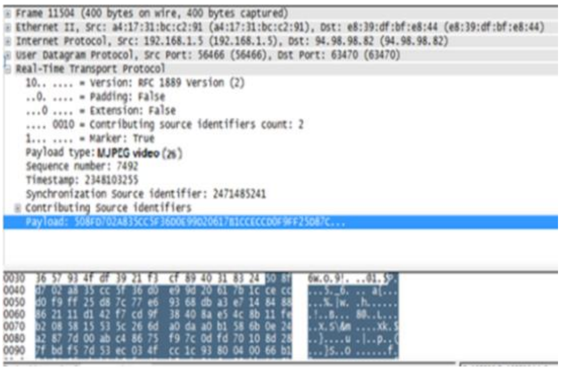
Microsoft: Capturing - Ethereal

Ethereal: Capture from Microsoft

Captured Packets

		% of total	
Total	26796		
SCTP	0		0.0%
TCP	745		2.8%
UDP	26016		97.1%
ICMP	0		0.0%
ARP	26		0.1%
OSPF	0		0.0%
GRE	0		0.0%
NetBIOS	0		0.0%
IPX	0		0.0%
VINES	0		0.0%
Other	9		0.0%

Running 00:07:19



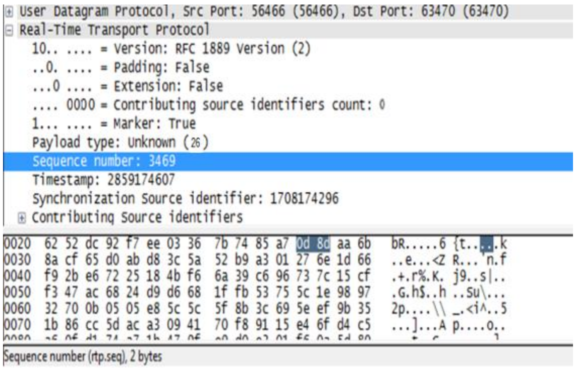
```
⊞ Frame 11504 (400 bytes on wire, 400 bytes captured)
⊞ Ethernet II, Src: a4:17:31:bc:c2:91 (a4:17:31:bc:c2:91), Dst: e8:39:df:bf:e8:44 (e8:39:df:bf:e8:44)
⊞ Internet Protocol, Src: 192.168.1.5 (192.168.1.5), Dst: 94.98.98.82 (94.98.98.82)
⊞ User Datagram Protocol, Src Port: 56466 (56466), Dst Port: 63470 (63470)
⊟ Real-Time Transport Protocol
    10.. .... = Version: RFC 1889 Version (2)
    ..0. .... = Padding: False
    ...0 .... = Extension: False
    .... 0010 = Contributing source identifiers count: 2
    1... .... = Marker: True
    Payload type: MJPEG video (26)
    Sequence number: 7492
    Timestamp: 2348103255
    Synchronization Source identifier: 2471485241
  ⊞ Contributing Source identifiers
    Payload: 508FD702A835CC5F3600E99D20617B1CCECCD0F9FF25D87C...

0030  36 57 93 4f df 39 21 f3  cf 89 40 31 83 24 50 8f   6W.O.9!. ..@1.$P.
0040  d7 02 a8 35 cc 5f 36 d0  e9 9d 20 61 7b 1c ce cc   ...5._6. .. a{...
0050  d0 f9 ff 25 d8 7c 77 e6  93 68 db a3 e7 14 84 88   ...%.|w. .h......
0060  86 21 11 d1 42 f7 cd 9f  38 40 8a e5 4c 8b 11 fe   .!..B... 8@..L...
0070  b2 08 58 15 53 5c 26 6d  a0 da a0 b1 58 6b 0e 24   ..X.S\&m ....Xk.$
0080  a2 87 7d 00 ab c4 86 75  f9 7c 0d fd 70 10 8d 28   ..}....u .|..p..(
0090  7f bd f5 7d 53 ec 03 4f  cc 1c 93 80 04 00 66 b1   ...}S..O ......f.
```

(a) **(b)**

Figure 9: (a) The UDP/RTP captured packets, (b) The RTP payload

In the proposed methods the initial value of the sequence number which has the length of 16 bits used to embed 16 bits of embedding data (suppose these 16 bits are 00001101 10001101 which represented using hexadecimal as 0d 8d) this means setting the initial sequence number to be 3469 as shown in Figure 10. This value is incremented by one for each RTP data packet sent and used by the receiver to detect packet loss and to restore packet sequence.



```
⊞ User Datagram Protocol, Src Port: 56466 (56466), Dst Port: 63470 (63470)
⊟ Real-Time Transport Protocol
    10.. .... = Version: RFC 1889 Version (2)
    ..0. .... = Padding: False
    ...0 .... = Extension: False
    .... 0000 = Contributing source identifiers count: 0
    1... .... = Marker: True
    Payload type: Unknown (26)
    Sequence number: 3469
    Timestamp: 2859174607
    Synchronization Source identifier: 1708174296
  ⊞ Contributing Source identifiers

0020  62 52 dc 92 f7 ee 03 36  7b 74 85 a7 0d 8d aa 6b   bR.....6 {t.....k
0030  8a cf 65 d0 ab d8 3c 5a  52 b9 a3 01 27 6e 1d 66   ..e...<Z R...'n.f
0040  f9 2b e6 72 25 18 4b f6  6a 39 c6 96 73 7c 15 cf   .+.r%.K. j9..s|..
0050  f3 47 ac 68 24 d9 d6 68  1f fb 53 75 5c 1e 98 97   .G.h$..h ..Su\...
0060  32 70 0b 05 05 e8 5c 5c  5f 8b 3c 69 5e ef 9b 35   2p....\\ _.<i^..5
0070  1b 86 cc 5d ac a3 09 41  70 f8 91 15 e4 6f d4 c5   ...]...A p....o..

Sequence number (rtp.seq), 2 bytes
```

Figure 10: Hidden data embedded in the initial value of the sequence number.

Hiding using the initial value of the sequence number did not cause an increasing in bandwidth or delaying in the receiving time, but the length of embedding is a limit (2 bytes).

4.4 Experimental Results

The relation between the dropping packets and the hidden data (image3 used) retrieval percentage (sample video1 used) shown in Table 1.

Table 1: The relation between the dropping packets and the hidden data (image3) retrieval percentage

Packet dropped%	Hidden data (Image3) Retrieval Percentage%
5%	97 %
10%	94.5%
15%	86.8%
20%	83.7%

Table 2: The enhancement in table (2) after using Receiver-Based Repair (Interpolation).

Packet Dropped%	Hidden Data Retrieval Percentage %
5%	98.6 %
10%	96.4 %
15%	88.6 %
20%	85.3 %

After applying the three proposed methods in this paper which used to recover the dropped packets the hidden data retrieval percentage% enhanced as shown in the Tables (2, 3, 4).

Table 3: The enhancement in table (2) after using (FEC).

Packet Dropped%	Hidden Data Retrieval Percentage %
5%	98 %
10%	95.6 %
15%	87.8 %
20%	84.5 %

Table 4: The enhancement in table (2) after using Interleaving.

Packet Dropped%	Hidden Data Retrieval Percentage %
5%	99 %
10%	97 %
15%	89.2 %
20%	86.2 %

Tables (2, 3, 4) show that the Interleaving gives the best results for Hidden Data Retrieval Percentage versus the dropped packets increasing. Figure 11 illustrates the comparison between using Receiver-Based Repair, FEC and Interleaving methods used for hidden image3 recovery. Table 5 (a) shows hidden recovered data (image3 and Text2) for 5% lost packets, Table 5 (b) shows the same hidden recovered data after using FEC method, Table 5 (c) shows the same hidden recovered data after using Interpolation method, Table 5 (d) shows the same hidden recovered data after using Interleaving method.

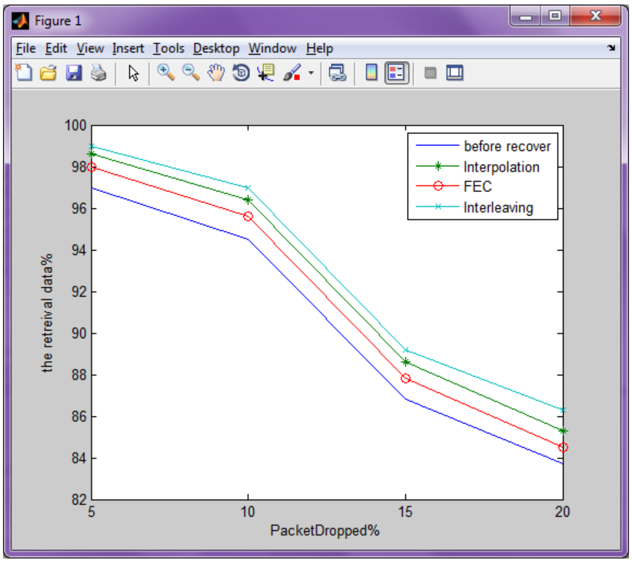
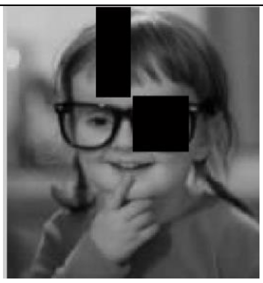
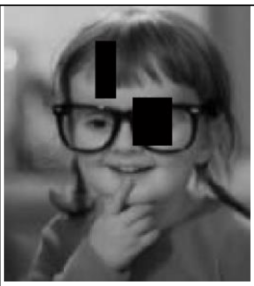
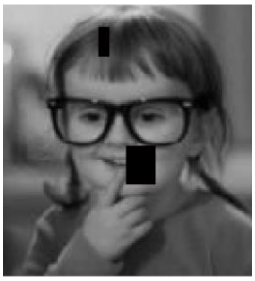
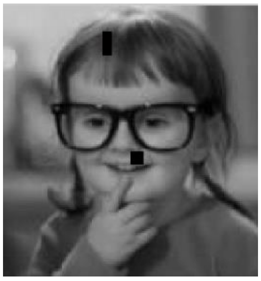


Figure 11: comparison between using Receiver-Based Repair (Interpolation), (FEC) and Interleaving approaches used for hidden data recovery.

Table 5: (a) Retrieved hidden data before using any supposed methods, (b) after using FEC, (c) after using Interpolation, (d) after using Interleaving.

Hidden data	Recovered hidden data for 5% lost packets			
Image3				
Text2	dota hidang is@a very interEsting fielp	fata hidinF is akvery interestinG field	data hiding us a very intyresting fiebd	data heding is a very interesting#field
	(a)	(b)	(c)	(d)

5. Conclusion

The paper proposed three methods for lost packet recovery for video broadcasting over the network. The protocols that are used for the transport layer were RTP/UDP. The used videos were M-JPEG stego videos (contain hidden data in RTP payload and in the initial value of the sequence number field). The hidden data may be grey level image or text. When the client requests the stored stego video, the video stream that was captured at the receiver using the Ethereal program. The proposed methods evaluated by calculating the hidden data retrieval percentage. The results show that the hidden data retrieval percentage increases as the number of dropped packets decreases and the Interleaving method achieved the best results for retrieving the hidden data from the video stream. Retrieving the images was better than retrieving the texts because if any bit changed in the text he ASCII of it will change completely, the challenge of future work will focus on this point.

References

[1]. M. Ellisz, D. P. Pezarosz, T. Kypraios and C. Perkinsz, "Modelling Packet Loss in RTP-based Streaming Video for Residential Users", Local Computer Networks (LCN), 2012 IEEE 37th Conference, 22-25 Oct. 2012, PP. 220-223.

[2]. 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, 2010, PP. 5–12.

[3]. X. Li, S. Paul, P. Pancha, and M. Ammar, "Layered video multicast with retransmission (lvmr):evaluation of error recovery schemes.", In Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video, St Louis, May 2011, PP. 8–15.

[4]. P. Lioyd, "An exploration of Covert Channels witin Voice Over IP". Dept. of Computing and Information Sciences, Thesis, Rochester Institute of Technology (2010), http://www.tele.pw.edu.pl/~krzysiek/pdf/steg-seminar-2003.pdf (retrieved June 17, 2010)

[5]. B. Lampson, "A note on the confinement problem," communication of the ACM" , vol. 16, October 1973. , PP. 613-615.

[6]. T. Maheshwari, N. Gupta, " Proposal for Robust and Adaptive Forward Error Correction for real-time Audio-Video streaming solutions" , IOSR Journal of Electronics and Communication Engineering (IOSR-JECE), ISSN: 2278-2834, ISBN: 2278-8735. Volume 3, Issue 2 (Sep-Oct. 2012), PP. 32-36

[7]. J. Evans, A. Begen, A., J. Greengrass, and C. Filsfils, "Toward Lossless Video Transport (to appear in IEEE Internet Computing)", November 2011.

[8]. Q. et al., "An Adaptive Motion-Based Unequal Error Protection Approach for Real-Time Video Transport Over Wireless IP Networks", IEEE TRANSACTIONS ON MULTIMEDIA vol. 8, no. 5, October 2006, PP. 1033 - 1044, XP055011882

[9]. G. Ramamurthy and D. Raychaudhuri. "Performance of packet video with combined error recovery and concealment". In *Proceedings of the IEEE INFOCOM*, pages 753–761, April 1995.

[10]. 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*, pages 5–12, 1996.

[11]. X. Li, S. Paul, P. Pancha, and M. Ammar. "Layered video multicast with retransmission (lvmr):evaluation of error recovery schemes", In *Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video, St Louis*, May 1997.

[12]. R. Xu, C. Myers, H. Zhang, and R. Yavatkar. "Resilient multicast support for continuous-media applications", In *Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video*, St. Louis, May 1997.

[13]. C. Leicher. "Hierarchical encoding of MPEG sequences using priority encoding transmission (pet)". Technical Report 94-058, ICSI, November 1994.

[14]. A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan. "Priority encoding transmission". *IEEE Transactions on Information Theory,* 42(6), November 1996.

[15]. T. Kinoshita, Nakahashi, and M. Takizawa. "Variable bit-rate hdtv coding algorithm for ATM environments in B-ISDN", In *Proceedings of SPIE Conference on Visual Communications and Image Processing: Visual Communication*, pages 604–612, November 1991.

[16]. J-C. Bolot and T. Turletti. ," Adaptive error control for packet video in the internet.", In *Proceedings of International Conference on Internet Protocols*, Lausanne, September 1996.

[17]. E. W. Biersack. "Performance evaluation of FEC in ATM networks", In *Proceedings of the ACM SIGCOMM,* Baltimore, MD, August 1992.

[18]. I. Rhee. , "Error control techniques for interactive low-bit rate video transmission over the internet", In *Proceeding of the ACM SIGCOMM'98 (to appear),* Vancouver, Canada, Sept. 1998.

[19]. I. Rhee, "Retransmission-based error control for interactive video applications over the internet", In *Proceedings of International Conference on Multimedia Computing and Systems*, pages 118–127, Texas, Austin, June 1998.

[20]. B. et al.," RTP Payload Format for JPEG-compressed Video", 1998: http://tools.ietf.org/html/rfc2435

[21]. H. Schulzrine., S. Casner., R. Frederick. and V. Jacobson..: "RTP: A Transport Protocol for Real-time Applications"., RFC3550, 2003: http://tools.ietf.org/html/rfc3550

[22]. H. Zhao, Y. Q. Shi. and N. Ansari., "Steganography in Streaming Multimedia over Networks", Transactions on Data Hiding and Multimedia Security VII . Lecture Notes in Computer Science Volume 7110, 2012, PP. 96-114

[23]. D. Barry," Understanding Packet Loss in Network Monitoring and Analysis Appliances", http://esj.com/articles/2012/12/13/understanding-packet-loss.aspx.