# DEPARTMENT OF ELECTRICAL ENGINEERING TECHNICAL REPORT

## WIRELESS TRANSPORT OF MPEG-2 STREAMS

Danny Hong, Gounyoung Kim and Alexandros Eleftheriadis

CU/EE/TR 2002-05-021

Columbia University
Department of Electrical Engineering
500 W. 120th Street, MC 4712
New York, NY 10027

http://www.ee.columbia.edu

Copyright © 2002 – The Trustees of Columbia University in the City of New York
All Rights Reserved

The technical reports in this series are considered to be semi-formal.
The ideas expressed are solely those of the authors, and questions about the content should be directed to them.

# Wireless Transfer of MPEG-2 Transport Streams

**Danny Hong**, **Gounyoung Kim**, **Alexandros Eleftheriadis**
danny@ee.columbia.edu, gk90@columbia.edu, eleft@ee.columbia.edu
Department of Electrical Engineering
Columbia University
New York, NY 10027

## Abstract

Wireless networking has evolved greatly over the past years as the demand for new and better wireless applications grew. From beepers to cellular (analog) phones, from cellular (analog) phones to PCS phones and from PCS phones to "pocket" computers that can access the Internet via wireless connections, wireless devices/applications certainly have taken their place in the telecommunications market and they continue to expand. As wireless devices get better and more powerful, wireless applications that process rich content such as audio, image and video will be in high demand. This paper looks into the most popular audio/video standard, MPEG-2 (specifically the MPEG-2 Systems part), and reports results from a study of its application in wireless. In the application, live audiovisual data are captured and compressed/encoded into MPEG-2 Transport Stream from a camcorder and a MPEG-2 encoder, respectively. Then, the encoded bit stream is packetized and transmitted via a wireless connection to the receiving end. Upon the reception of the data, the receiver forwards the data to a display unit that has the capability to playback the received data in real-time. From the study, the errors (packet losses) that occur due to the erroneous nature of the wireless network and their effects on the quality of the received MPEG-2 Transport Stream are described. The wireless network is comprised of just the sender and the receiver using IEEE 802.11 compatible network cards. Due to the packet losses, distortions are seen when playing back the received data in the display unit. As a remedy to the distortion problem, an error concealment mechanism is also suggested in this paper. There are many proposed techniques/mechanisms for correcting and concealing network errors. However, we propose an error concealment mechanism suitable for high bandwidth, real-time applications that process MPEG-2 Transport Stream (TS). Also, a MPEG-2 TS stream is a multiplexed stream of audio, video and other private data and an algorithm used in an error concealment mechanism for video streams may not work for the TS stream. In this paper, an error concealment mechanism, applicable in real-time, for video streams is discussed. Then, algorithms are proposed to make the same error concealment mechanism work for the TS streams.

## 1. Introduction

Distribution of audio/video via a network is of great interest but also poses a number of challenges. If there is a competing traffic in the network, there is always the risk of congestion, and consequently, packets may be dropped or delayed [1]. In the wireless

network, the problem becomes more interesting as the links are more error-prone than connected links. Also, wireless links are usually of less bandwidth than connected links and, thus, there may be more loss and delay of packets.

There are many ways to alleviate the problems mentioned above. For example, huge bandwidth may be assigned to links (although wireless links are very limited in bandwidth) that have audio/video information going through. Alternatively, before transmitting such information, bandwidth can be reserved on the links that the information will go through. Certainly, these mechanisms can decrease the errors, however, in the wireless network, errors can never be avoided. Even though the wireless network is comprised of links that have more than enough bandwidth, any unexpected interference such as objects passing by between the sending and the receiving ends of the wireless network can cause loss of packets. In this paper, we discuss the packet loss errors that occur in the wireless network and its effects on the quality of the received MPEG-2 Transport Stream (TS).

Due to the packet loss errors, distortions are seen in the receiving end when displaying the received data. Since the errors cannot be avoided, a mechanism had to be developed in order to conceal the errors. In References [2-7], some of the error correcting and concealing techniques/mechanisms for video streams are discussed. However, the discussed mechanisms are not suitable for high bandwidth, real-time applications that process the TS stream. In this paper, we propose an error concealment mechanism that improves the quality of the display by playing back only correctly received pictures. The mechanism requires very little processing; therefore, it is suitable for real-time applications. The mechanism can be easily applied if a MPEG-2 Video stream is being transmitted. In the case of transmitting a TS stream, the mechanism had to be carefully designed for it to work properly. In this paper, we describe the algorithms (Packetizer and Reconstructor in Figure 1) that enable the mechanism to work with the TS stream.

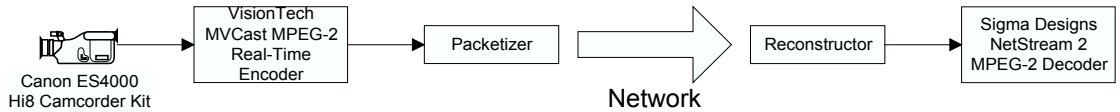


**Figure 1: The algorithms (Packetizer and Reconstructor) suggested for an error concealment mechanism. The Packetizer gets the TS stream from MPEG-2 hardware encoder and packetizes the stream so that the error concealment mechanism can be applied. The Reconstructor reconstructs the TS stream from the received packets, and if any packets are missing, it adds "padding" TS packets so that the MPEG-2 hardware decoder can render the constructed stream without any distortion.**

For this study, we used MPEG-2 standard because it is the most popular standard that is used today for encoding audio/video. The hardware used for this study is described in the next section. In Section 3, the MPEG-2 standard is briefly discussed. Then, in Section 4, the results from the study of the wireless application are listed and explained. In Section 5, the error concealment mechanism is described and in Section 6, the algorithms used for the mechanism are explained in detail. Lastly, this paper ends with conclusion, future research topics and applications in Section 7. The error concealment mechanism for the

TS stream requires the use of a "padding" TS packet (TS pkt) and the structure of the TS pkt is specified in Appendix A. The reasons for using the TS pkt is described in Section 5 and Section 6.

## 2. Hardware Configuration

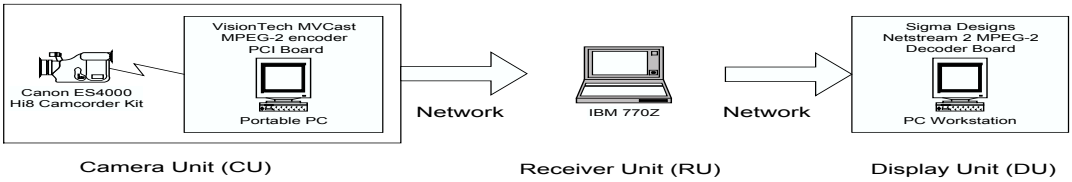


**Figure 2: The System Hardware Configuration**

For the study, we created a system that is comprised of three units (see Figure 2). The first unit is the Camera Unit (CU) and this unit has the capability to capture live audiovisual data, encode the captured data, and transmit the encoded data over a wireless network. In the CU, we used Canon's ES4000 Hi8 Camcorder to capture the live data. The camcorder with S-Video and RCA outputs is directly connected to a VisionTech MVCast MPEG-2 Real-Time Encoder PCI Board that is plugged into a portable PC. The encoder can generate MPEG-2 Video or Systems stream (Program Stream or Transport Stream). For this unit, we developed software that obtains the compressed stream from the encoder, packetizes the stream into UDP packets, and then, transmits them over a wireless network. The portable PC is composed of a Celeron 533 MHz CPU and 128 MB RAM. For the wireless network connectivity, we used one of Lucent's wireless PCMCIA cards (WaveLAN 11 MB Silver Card IEEE). We didn't choose a laptop as the CU because no hardware MPEG-2 encoder w/ PCMCIA interface was available. We eliminated the method of having an ISA adapter directly making the connection between the PC and the wireless card because it proved to be too slow for real-time applications. Hence, we attached a PCI Ethernet NIC to the PC and it is connected via a swapped patch cable to Lucent's WavePOINT Access Point (AP). The wireless card is plugged into the AP and it is connected to Lucent's 7 dBi Omni Directional Antenna to extend the normal operational range of the wireless card.

The second unit is the Receiver Unit (RU), which is IBM's ThinkPad 770Z laptop with Pentium II 360 MHz CPU and 128 MB RAM. The RU receives the encoded data sent by the CU through the wireless interface and forwards the data to the third unit, the Display Unit (DU). The same wireless card with the same antenna as the ones used in the CU are directly attached to the RU for the wireless network connectivity. For the connectivity between the RU and the DU, a PCMCIA Ethernet NIC is attached to the RU and it is connected to a PCI Ethernet NIC at the DU. For this unit, we developed a software that receives the data sent from the CU and forwards the data to the DU.

We chose the laptop as the RU for two reasons. First, we wanted the CU and the RU to have mobility capability. When the CU is moved to a certain location for recording, the

RU can be moved nearby the CU so that the wireless connectivity between the two units can be made stable. Second, the wireless network card we used is a PCMCIA card, thus, it goes naturally with the laptop. However, the drawback of using the laptop as the RU was that there was no available MPEG-2 hardware decoder for it. So, the third unit (DU) is created as a playback unit.

The third unit, DU, is a Dell Precision 410 Workstation with dual Pentium III 500 MHz CPU and 256 MB RAM. A Sigma Designs NetStream 2 MPEG-2 Decoder Board is plugged into the workstation. The decoder card performs decoding in hardware allowing for a higher rate of decoding compared to software decoders giving the real-time capability. The operating system running on all the machines (CU, RU, and DU) is Windows NT 4.0. The diagram of the system configuration is displayed in Figure 3.

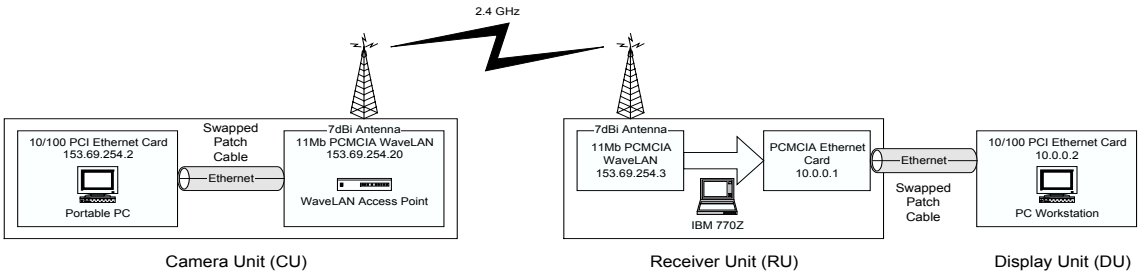


**Figure 3: Network Configurations and Connectivity of the System**

In summary, the camcorder attached in the CU captures live audiovisual data and sends the data to the encoder where the encoding/compression takes place. Subsequently, the encoded audiovisual data are packetized, and then, they are passed through the swapped patch cable from the Ethernet card in the portable PC to the AP. The AP is used as a bridge from Ethernet to wireless. From the wireless card in the AP to the same wireless card in the RU, the data is transmitted in 2.4 GHz range. Then, the data gets forwarded through the Ethernet card from the RU to the Ethernet card in the DU. Finally, the DU plays back, in real-time, the received data using the hardware decoder.

## 3. MPEG-2 Background

### *3.1 MPEG-2 Video*

MPEG-2 [8, 9] Video was designed for coding of interlaced video of standard TV resolution at bit rate range of 4-9 Mbps. This part of MPEG-2 standardizes the video bit stream syntax and decoding syntax. The scope of MPEG-2 Video was considerably revised to include video of higher resolution such as HDTV at higher bit rates as well as hierarchical video coding for a range of applications.

In MPEG, a sequence of transmitted video pictures is typically divided into series of Group of Pictures (GOPs), where each GOP is an arrangement of one or more Intra-coded pictures (I-pictures), Forward Predictive-coded pictures (P-pictures) and Bidirectionally Predicted pictures (B-pictures) [8]. The MPEG-2 encoder specified in Section 2 produces the following sequence of pictures for all the GOPs except for the very first GOP:

| **Picture #** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Picture Type** | I | B | B | P | B | B | P | B | B | P | B | B |

**Figure 4: The GOP Sequence Produced by the Encoder**

The very first GOP for any video stream coded by the encoder looks as follows:

| **Picture #** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Picture Type** | I | P | B | B | P | B | B | P | B | B |

**Figure 5: The Very First GOP Sequence Produced by the Encoder**

The above sequences are both coded sequences. That is, they are the sequences produced by the encoder as an output and they are fed into the decoder as an input to be decoded.

The display sequence for the second sequence is as follows:

| **Picture #** | 1 | 3 | 4 | 2 | 6 | 7 | 5 | 9 | 10 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Picture Type** | I | B | B | P | B | B | P | B | B | P |

**Figure 6: The Corresponding Display Sequence for the Coded Sequence in Figure 5**

### *3.2 MPEG-2 Systems*

The MPEG-2 Systems part (ISO/IEC 13818-1) specifies a coding layer for the combined coded audio and video data and provides the capability for combining with it user-defined private data streams as well as streams that may be specified at a later time. More specifically, the MPEG-2 Systems standard defines a packet structure for multiplexing coded audio and video data into one stream and enables synchronized playback [8].

MPEG-2 Systems defines two types of streams: the Programs Stream (PS) and the Transport Stream (TS). The PS is similar to the MPEG-1 Systems stream and its decoder should be backward compatible with MPEG-1. However, the TS differs significantly from the MPEG-1 Systems stream as well as the PS. It offers robustness necessary for noisy channels as well as the ability to include multiple programs in a single stream. It uses fixed length packets of size 188 bytes, with new header syntax. It is therefore more suited for hardware processing and for error correction schemes [8]. With its small

packets, the TS was developed for transmission systems and can carry all data types, including other (non-MPEG) compressed video, machine control commands for television equipment, bank accounts, or the latest news [10]. Because of this great importance of the TS, we decided to encode the captured audiovisual data in TS stream and the transmission of the stream over the wireless network is studied in this paper.

## 4. Experimental Results

Use of UDP as the transport protocol results in loss of packets when network errors occur. There were some delay and jitter, however, most of the distortions seen on the display, when playing back the received data in real-time, were caused by loss of packets. Thus, in this section, we only discuss packet losses and their effects on the received TS streams.

### *4.1 Packet Loss at Fixed Sizes*

We chose two different distances between the CU and the RU – 50 ft. and 150 ft. For each distance, we recorded live audiovisual data through the camcorder, and the encoder, at different encoding rates, generated the TS streams. Then, we ran eight different tests for each of the eight different UDP packet sizes we chose. From the data collected, packet loss rates are calculated for each packet size at each encoding rate per given distance between the CU and the RU. Table 1 lists the packet loss rates for different packet sizes at different encoding rates and 50 ft. distance between the CU and the RU. The packets have fixed size; hence the packet loss rate can also be viewed as the byte (or bit) loss rates. The first column lists the encoding rates and the first row lists the packet sizes at multiples of TS packets. Therefore, the entry corresponding to the second row and second column (1.38%) would be the packet loss rate at 3 Mbps encoding rate with packet size of 25x188 bytes.

| | **25** | **50** | **75** | **100** | **150** | **200** | **250** | **300** |
|---|---|---|---|---|---|---|---|---|
| **3 Mbits/s** | 1.38% | 1.25% | 1.10% | 1.08% | 1.04% | 1.05% | 0.94% | 0.72% |
| **4 Mbits/s** | 1.62% | 1.55% | 1.47% | 1.52% | 1.46% | 1.45% | 1.57% | 1.39% |
| **5 Mbits/s** | 1.45% | 1.33% | 1.22% | 1.32% | 1.14% | 0.82% | 1.18% | 0.81% |
| **6 Mbits/s** | 91.98% | 85.92% | 84.04% | 76.21% | 80.80% | 73.35% | 76.02% | 76.45% |

**Table 1: Packet Loss Rates at 50 Feet Distance between the CU and the RU.**

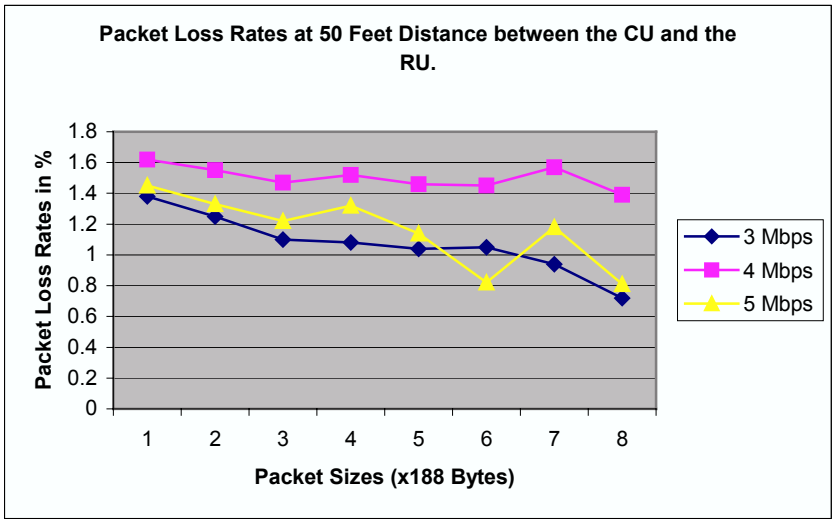


**Figure 7: The Graphical Representation of Table 1. The Packet Loss Values at 6 Mbps Encoding Rate Are Not Displayed in the Graph.**

| | 25 | 50 | 75 | 100 | 150 | 200 | 250 | 300 |
|---|---|---|---|---|---|---|---|---|
| **3 Mbits/s** | 1.71% | 1.40% | 1.36% | 1.30% | 1.52% | 1.22% | 1.18% | 1.12% |
| **4 Mbits/s** | 1.42% | 1.46% | 1.92% | 1.21% | 2.30% | 1.87% | 1.66% | 1.18% |
| **5 Mbits/s** | 6.72% | 4.83% | 1.15% | 0.94% | 0.90% | 1.06% | 1.09% | 0.70% |
| **6 Mbits/s** | 97.5% | 85.9% | 89.26% | 76.4% | 89.62% | 88.72% | 80.76% | 79.94% |

**Table 2: Packet Loss Rates at 150 Feet Distance between the CU and the RU.**

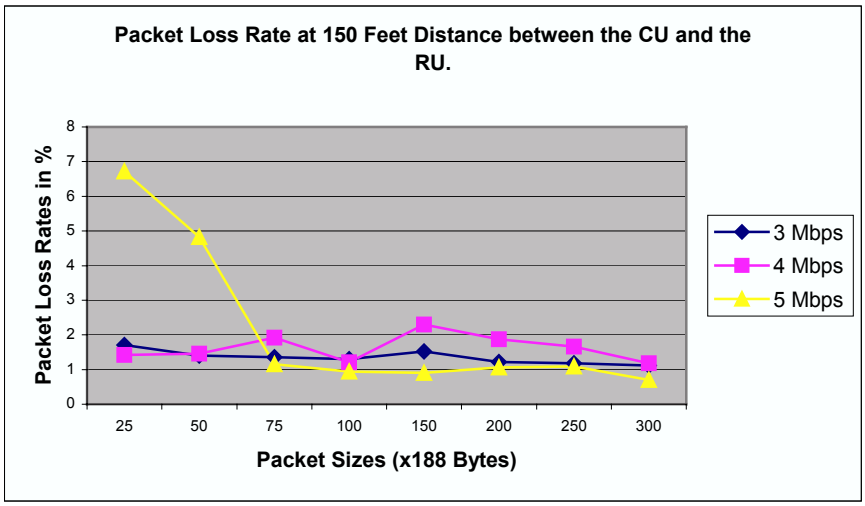


**Figure 8: The Graphical Representation of Table 2. The Packet Loss Values at 6 Mbps Encoding Rate Are Not Displayed in the Graph.**

Each packet loss rate in the above tables is calculated from one hour worth of data. From the data, we speculated that the bigger the packet size, the less error we get. Table 2 is the same as Table 1 except for the distance between the CU and RU. For the measurements in Table 2, we set the distance between the two units to be 150 ft. The packet loss rates obtained in this setting weren't too much different from the ones obtained at 50 ft. distance between the two units. This is due to the fact that we used antennae powerful enough for the two units and the antennae were able to pick up the signals in 150 ft. distance as well as in 50 ft. distance. However, when the distance between the units increased to 250 ft., the error rates were observed to be higher. After the distance between the antennae increased past certain point, the network error started to increase in accordance.

From the above two tables, it is clear that the system does not perform well when encoding at 6 Mbps. We ran exactly the same test but by replacing the laptop, acting as the RU, with a desktop that has the same performance power as the DU. Hence, we have replaced RU with a more powerful machine. In this case, we were able to come up with similar average packet loss rates as the ones we got for the 5 Mbps encoding rate case. This convinced us that the physical wireless interface part of the system is able to support such high throughput, however, the receiving machine wasn't fast enough to process all the data to be received. Nevertheless, even with the more powerful machine acting as RU, we were still observing packet losses. This is due to the fact that a wireless network is error-prone.

According to Reference [11], on UDP, it is possible to send any size datagram less than or equal to the maximum allowed UDP size. However, this is not advisable for a number of reasons. It incurs extra network-system overhead by forcing fragmentation where the entire datagram must be reassembled before delivery. Worst of all, if the local or remote network-systems do not have adequate buffer space available the operation may fail without an error, since datagrams are unreliable. In our case, the network-system has more than enough resources to handle big UDP packets. However, the system lacked resources when obtaining and processing data from the network buffer. Thus, we had to minimize the number of socket calls (by increase the packet sizes) in order to come up with better performance.

In summary, the packet loss rates we obtained are not acceptable for broadcast-quality applications. Even at a packet loss rate as low as 0.703 %, distortions on the display at DU were quite recognizable. To improve the quality of the received TS stream, an error correction or concealment mechanism should be applied.

### *4.2 Average Packet Loss for each Picture Type*

Another important measure to take for such inter-picture coded data, as MPEG-2 encoded bit stream, is the packet loss rates for each picture type. In order to obtain such data, we had the CU packetize the encoded bit stream one picture at a time and have each UDP packet carry one and only one picture information. This way, if a packet is lost, one and

only one picture information is lost. However, in some cases, we had I-pictures composed of number of bytes that exceeded the maximum byte size allowed by UDP. So, for every I-picture, we had two UDP packets carrying the data. Table 3 lists the average packet loss rates for each picture type.

| Picture Type | Ave. Pkt. Loss Rate (Test 1) | Ave. Pkt. Loss Rate (Test 2) | Ave. Pkt. Loss Rate (Test 3) |
|---|---|---|---|
| B | 1.41 % | 1.23 % | 1.29 % |
| P | 0.82 % | 1.35 % | 0.97 % |
| I | 1.18 % | 0.72 % | 1.21 % |

**Table 3: Average Packet Loss Rates for each Picture Type**

With encoding rate set at 3 Mbps at 150 ft. distance between the CU and RU, we ran 3 tests. Each test was performed for one hour, and with the data obtained from the three tests, we calculated the packet loss rates listed above. Number wise, more B-pictures were lost because there were more B-pictures sent over the network. But, taking the ratio of lost packets to the sent packets for each picture type as the average packet loss rate, the overall loss rate for each picture type came to be about the same.

When we were playing-back the live audiovisual data in the DU, we saw distortions in the display for a very short period of time if B-pictures were lost. However, when P or I-pictures were lost, the distortion lasted for longer durations. This is due to the fact that even though only one P-picture is lost, all the pictures that depend on the anchor picture get affected.

When a B-picture is lost, there shouldn't be any distortions in the display because B-pictures cannot be chosen as anchor pictures. However, as stated above, distortions were seen even when only B-pictures were lost. In the next two sections, we introduce an error concealment mechanism that can be used to improve the quality of the received MPEG-2 Transport Stream.

## 5. Error Concealment

Due to the lack of throughput supported by the wireless network devices and the erroneous nature of the wireless link, some packets get lost, and thus, distortions are seen in the receiver's display when audiovisual data are transmitted. In the wireless network, total avoidance of packet loss is impossible unless retransmission is applied. But, for real-time applications, retransmission causes undesirable delays. Therefore, a mechanism should be applied to conceal the unavoidable errors caused by packet losses.

There are many suggested techniques/mechanisms developed for concealing network packet loss errors. However, most of the suggestions are based on the assumption that the packets will contain only a portion of a picture. That is, most of the mechanisms deal

with concealing the errors caused by macroblock (MB) losses. There are two reasons why the mechanisms are not applied in our system. First, as discussed in Section 4, lower packet loss rates are observed for bigger packet sizes, and for better throughput, the UDP packet size should be made as big as possible. Second, since we are dealing with the TS stream, it is very hard to packetize the bit stream in the macroblock level and causes too much delay in the real-time application. Thus, each packet is made big enough to carry one whole picture at a time, and for our system, a mechanism for concealing the loss of whole picture is needed. Reference [12] suggests a technique for concealing loss of whole pictures using the Motion Vectors (MVs) of the received pictures. This again, is not an efficient solution for our application where MPEG-2 TS streams are being sent and received in real-time. There will be too much processing time consumed in the receiver as it needs to look inside the TS stream and go into the video stream in order to obtain the MVs. Before suggesting an error concealment mechanism that is suitable for real-time applications for the MPEG-2 TS stream, the same error concealment mechanism is introduced for the MPEG-2 Video stream in the next subsection.

### *5.1 Error Concealment for MPEG-2 Video Stream*

One efficient way to packetize the MPEG-2 Video stream is to cut it one picture at a time. That is, have each network packet (UDP packet) carry one and only one picture information. This way, if a packet is lost, then the picture carried by the packet can be ignored and other correctly received pictures may be displayed with no distortions (if the lost packet contained a B-picture). Since the encoder always produces a fixed sequence of pictures for each GOP (if this is not the case, the encoder can be modified to do so), the receiver can have the knowledge of the GOP sequence in advance. So, the receiver knows, a priori, which packet, thus which picture, it is going to receive. If an expected packet is not received (due to the network loss), the receiver can do one of the following in order to conceal the error produced by the packet loss:

(From now on, I shall exchange the usage of the terms – packet and picture)

A. If a B-picture is lost, then just ignore the loss and continue displaying other pictures
B. If a P-picture is lost, then discard all the P and B-pictures that depend on the lost picture and continue displaying other correctly received pictures.
C. If an I-picture is lost, then discard all the P and B-pictures that depend on the lost picture and continue displaying other correctly received pictures.

For example, looking at Figure 4, if the 2$^{nd}$ P-picture (Picture #7) is lost from the GOP sequence, then all the pictures corresponding to Picture #8 to Picture #12 should be discarded. Additionally, the first two B-pictures (Picture #2 and Picture #3) in the next GOP sequence should also be discarded.

Packetizing the video stream for the error concealment is a fairly easy task. In the CU, a simple algorithm is added for parsing the encoded bit stream and cutting the stream

whenever a new picture is encountered. The algorithm looks for the picture **start_code** = **start_code_prefix** (0x000001) + **picture_id** (0x00) [9]. The algorithm for the receiver, as described in the previous paragraphs, is added in the DU. Many tests were performed with the system described in Section 2, and as the result of the error concealment mechanism, every picture that was displayed had no distortion even though there were packet losses. However, there were some noticeable discontinuities in the display whenever I or P-pictures (anchor pictures) were lost. The loss of anchor pictures caused many other pictures to be discarded as well. When only few B-pictures were lost, the quality of the display was very hard to distinguish from the display of the same video stream with no packet loss. The next subsection describes a way to apply the same error-concealment mechanism to the MPEG-2 TS stream.

### *5.2 Error Concealment for MPEG-2 Systems Stream*

Packetizing the MPEG-2 Systems stream is trickier than packetizing the MPEG-2 Video stream. Since audio, video and possibly other user data are multiplexed as one stream, extra important information such as synchronization information is also stored in the systems stream. Thus, extra care must be taken when packetizing such stream.

#### 5.2.1 Program Stream

In the PS stream, Packetized Elementary Streams (PESs) from different sources are multiplexed into series of Packs [8]. Each PES starts with an important header (control) information for the data it contains, and as the name indicates, a PES is the basic element of the PS stream. Thus, when packetizing, cutting the stream within a PES should be avoided. Unfortunately, after studying the PS stream generated by the encoder, we found out that one PES can contain information from two different pictures. That is, the encoder just cuts the video/audio stream at any position (as long as the timing criteria are satisfied) and creates the PESs. For this reason, a PES starting near the end of a picture may continue to the next picture. This makes packetizing the PS stream for the error concealment very hard. Without further investigation, we proceed to the next subsection.

#### 5.2.2 Transport Stream

The TS stream is already packetized into the TS pkts that are 188 bytes in size. Each TS pkt can contain only one PES information at a time, and since a PES is usually bigger in size than a TS pkt, one PES is usually broken into many TS pkts. If the TS pkt contains a new PES, then the TS pkt header has the **payload_unit_start_indicator** set [9]. Thus, it is very easy to search for the TS pkts that contain the beginning information of a PES. However, even with the TS stream, there were few cases where one PES contained information from two different pictures. Nevertheless, we came up with a heuristic algorithm for packetizing the stream one picture at a time. The algorithm parses through the stream and keeps cutting it right before a TS pkt with the **payload_unit_start_indicator** set, and the PES contained in the TS pkt carries new picture information. The easiest way to look for a new picture information is to look for

a PES that contains picture **start_code** = **start_code_prefix** (0x000001) + **picture_id** (0x00). However, this method fails at rare occasions because the encoder can also break the **start_code** into two different PESs. Therefore, the algorithm has to look for other **start_code**s as well. More detailed description of the algorithm for packetizing the TS stream can be found in Section 6.

One very important field to keep track of at the receiver is the **continuity_counter** field of the received TS pkts. It is a counter with 4 bits that is incremented with each non-repeated TS pkt per PID (per source) [9]. For example, if a TS stream is composed of a video source and an audio source, then there are two **continuity_counter**s that the receiver must keep track of. With the decoder we used, distorted pictures were displayed when the **continuity_counter**s were not incremented uniformly. Thus, even though we packetized the TS correctly, and only correctly received pictures that are independent of the lost pictures are displayed at the receiver, if there was a jump in the **continuity_counter**, distortions were seen in the display. Losing a packet may result in a discontinuity in the counter and the decoder reacts unfavorably to this. Thus, "padding" TS pkts had to be created in place of the lost packets to make the **continuity_counter** continuous. The "padding" TS pkt is defined in Appendix A.

## 6. The Algorithms Used for the Error Concealment Mechanism

There are two separate algorithms applied for the error concealment mechanism described in Section 5. One is an algorithm for the sender (Packetizer) and the other is for the receiver (Reconstructor). The next subsections describe the algorithms in detail.

### *6.1 Acronyms and Definitions*

Below is the list of the acronyms and definitions used in the descriptions of the sender and receiver algorithms. From herein, unless otherwise specified, the word "packet" means the network packet (UDP packet) sent from the sender to the receiver.

| | |
|---|---|
| AC1 | The Audio Counter 1. This keeps track of the number of audio TS pkts (the TS pkts carrying the PES generated from an audio source) generated from the beginning of the TS bit stream to the end of the previous packet generated. |
| AC2 | The Audio Counter 2. This keeps track of the number of audio TS pkts generated from the beginning of the TS bit stream to the end of the current packet generated. |
| inserted_AC | The total number of audio TS pkts inserted into the receiver's queue. |
| inserted_VC | The total number of video TS pkts inserted into the receiver's queue. |

| | |
|---|---|
| delB | The sequence number of the B-picture that should be discarded because at least one of its anchor pictures is lost. |
| ESN | The expected sequence number of the packet to be received. This is an integer that is kept track of in the receiver. |
| ISN | The sequence number of the received packet that is last inserted into the queue of the receiver. |
| I1 | The sequence number of the first half of the I-picture in the GOP. I-pictures are sometimes bigger than the maximum allowed UDP packet size, and thus, the pictures are cut in half before being sent. |
| I2 | The sequence number of the second half of the I-picture. |
| MSN | The number of missed/lost packets in the GOP. $MSN = SN - ESN$. |
| NP | The (network) packet sent from the sender to the receiver. |
| **ext_start_id** | The extension start id. 0xB5 [9]. |
| **group_start_id** | The group start id. 0xB8 [9]. |
| last_slice_pos | The byte position of the last slice of the current picture. |
| **picture_id** | The picture start id. 0x00 [9]. |
| P1 | The sequence number of the very first P-picture in the GOP. |
| P2 | The sequence number of the second P-picture in the GOP. |
| P3 | The sequence number of the third (last) P-picture in the GOP. |
| sc_prefix | The **start_code_prefix**. 0x000001 [9]. |
| slice_id | The id of a slice. |
| SN | The sequence number of the packet. It's an integer starting from 0 and then it is incremented by one for the next packet and so on. |
| **start_code** | The 32-bit codes embedded in the coded bit stream that are unique. They are used for several purposes including identifying some of the layers in the coding syntax. It consists of a 24-bit prefix (0x000001) and an 8-bit **stream_id** [9]. |
| **stream_id** | The 8-bit id [9]. |
| **sync_byte** | The TS sync byte. 0x47 [9]. |
| VC1 | The Video Counter 1. This keeps track of the number of video TS pkts (the TS pkts carrying the PES generated from a video source) generated from the beginning of the TS bit stream to the end of the previous packet generated. |
| VC2 | The Video Counter 2. This keeps track of the number of video TS pkts generated from the beginning of the TS bit stream to the end of the current packet generated. |

### *6.2 Sender Algorithm (The Packetizer)*

The Packetizer is placed in the CU and it is used to packetize the TS bit stream a picture at a time. It is a heuristic algorithm solely based on the encoder specified in Section 2. There may be a need to modify the algorithm when using a different encoder. The steps for the algorithm are as follows:

A) Create the very first NP (with SN = 0) by cutting the bit stream right before the first TS pkt that contains a PES of an audio or a video source with the **payload_unit_start_indicator** set. Unique **PID**s are assigned for each source (audio and video) and this makes associating a TS pkt with a source very easy. The beginning of the generated bit stream always contains null TS pkts (**PID** = 0xFF), Program Association Table (**PID** = 0x00) and Program Map Table (**PID** = 0x10) before the actual audiovisual data [9]. All these information will be sent out to the network as the very first packet.

B) Set slice_id = 0. Then, fast scan 25 TS pkts. It is speculated from many experimental results that every picture is comprised of more than 25 TS pkts. The **sync_byte** and the fixed size of TS pkts make it easy to fast scan the bit stream. While scanning through the bit stream, for each audio TS pkt, AC2 should be incremented. Likewise, for each video TS pkt, VC2 should be incremented. The audio video sources have unique **PID**s, thus, it is very easy to distinguish between the two types of TS pkts.

C) Scan through the bit stream one byte at a time until the sc_prefix is encountered. While scanning through the bit stream, for each audio TS pkt, AC2 should be incremented. Likewise, for each video TS pkt, VC2 should be incremented.

D) **If** the next byte (**stream_id**) == slice_id
{
    Update the last_slice_pos and go back to Step C.
}

**Else if** the slice_id != 0 and **stream_id** == **picture_id** or **group_start_id** or **ext_start_id**
{
    Go to Step E. It is speculated that the new and only new picture always starts with at least one of the three **stream_id**s specified above.
}

**Else**
{
    Go back to Step C.
}

E) The location for the beginning of a new picture is found. Go back a TS pkt at a time until the one with **payload_unit_start_indicator** set is encountered (a new PES is encountered).

F) Packetize and send out to the network. Set AC1 = AC2 and VC1 = VC2. Repeat from Step B until end of the bit stream.

It is important to update and attach SN, AC1, AC2, VC1, and VC2 whenever packetizing. AC1, AC2, VC1, and VC2 are initially set to 0. They are used by the receiver in order to determine the number of "padding" TS pkts needed in case of packet losses. This is very important in order to maintain the continuity of the **continuity_counter** of each source. For each subsequent packet, SN should be incremented by one. I-pictures are sometimes bigger in size than the maximum UDP packet size. Hence, when packetizing an I-picture, the data should be first cut in half and two packets should be generated with different but consecutive SNs. The receiver looks at the SN of the received packet in order to determine whether it has received a correct packet or not.

In Step D, we look for any one of the three **stream_id**s – **picture_id**, **group_start_id** or **ext_start_id** – in order to distinguish between the pictures and find the right positions to cut in the TS bit stream. Normally, it should be enough to look for the **picture_id** for the beginning of a new picture. However, there were few cases where the picture **start_code** is broken into two different **PES**s. That is, there was one PES that ended right at the middle of the **start_code** and the next **PES** continued from there. The encoder always generates **group_start_id** whenever an I-picture is to be generated. So, for packetization, the sender can safely cut the TS bit stream right before the PES that contains the **stream_id**. Also, it is speculated that the encoder produces **ext_start_id** whenever a new picture is being generated. Therefore, the sender can also use the **stream_id** in order to cut the bit stream a picture at a time. It is possible that the encoder can break a **start_code**, with any one of the three **stream_id**s, when generating **PES**s, however, breaking more that one **start_code** at the same time will be almost impossible because the **stream_id**s are located very close to each other and a **PES** usually contains more bits than the bits allocated between the **stream_id**s.

Below is the diagram that shows the steps describing the algorithm for the sender. Basically, the sender needs to packetize the TS bit stream, generated by the encoder, a picture at a time. Then, attach some control information (SN, AC1, AC2, VC1, and VC) in front of each packet before sending it out to the network. The receiver, when applying the error concealment mechanism, uses the control information.

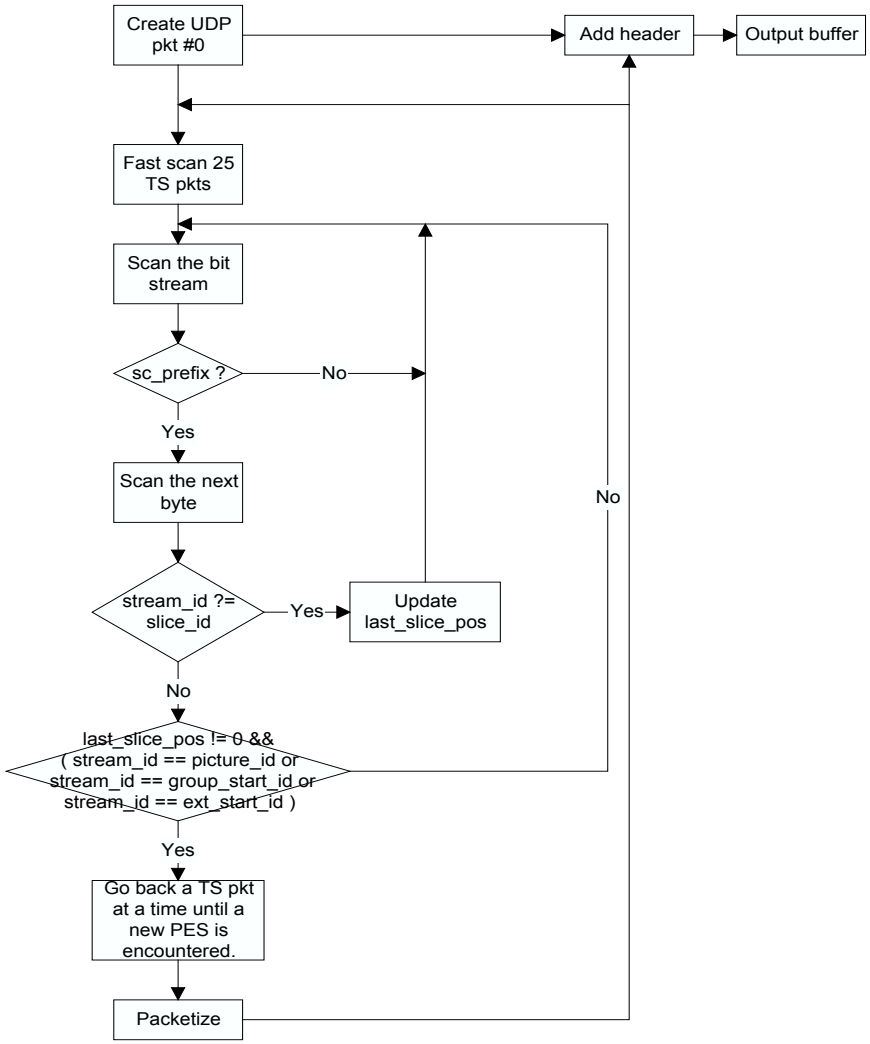


**Figure 9: The Packetizer – The algorithm packetizes the TS bit stream a picture at a time. For each packet generated, a header composed of SN, AC1, AC2, VC1 and VC2 is added. The packets inserted into the output buffer are sent out to the network.**

### *6.2 Receiver Algorithm (The Reconstructor)*

The Reconstructor is placed in the DU and it gets called whenever a packet is received. In the case of receiving the MPEG-2 Video bit stream, a lost picture can just be ignored if it is a B-picture. Otherwise, if the lost picture is an anchor picture (P or I-picture), then all the pictures that are dependent on the lost picture should be discarded. However, in the case where the packets carry the TS bit stream, the lost picture should be replaced by "padding" TS pkts. This is needed in order to maintain the continuity of the **continuity_counter**. The algorithm basically shows a way to replace lost data with the "padding" TS pkts. It is a heuristic algorithm solely based on the decoder specified in Section 2. There may be a need to modify the algorithm when using a different decoder. The steps for the algorithm is described as follows:

A) Set MSN = SN – ESN.

B) **If** MSN < 0, then the received packet is missing at least one anchor picture. Replace the received packet with "padding" TS pkts and insert them into the queue.
**Else if** MSN == 0, go to Step C.
**Else** (MSN > 0), go to Step D.

C) A correct packet is received.
   a. **If** SN != I2 and ISN < SN – 1, then one or more packets were lost prior to the correctly received packet. Replace the lost packets with "padding" TS pkts and insert them into the queue.
   b. **If** SN == DelB or DelB + 1
      {
      The received packet contains a B-picture with at least one anchor picture missing. Replace the received packet with "padding" TS pkts and insert them into the queue.
      }
      **Else if** SN == I1
      {
      The first half of an I-picture is received. This should be held in a temporary buffer until the second half of the picture is correctly received.
      }
      **Else if** SN == I2
      {
      The second half of an I-picture is received. Both payloads of the packets corresponding to I1 and I2 should be inserted into the queue.
      }
      **Else**
      {
      P or B-picture is received. Insert the picture into the queue.
      }
   c. ESN = SN + 1

D) One or more packets are lost.
   a. **If** SN == I1
      {
      Replace the lost packet(s) with "padding" TS pkts and insert them into the queue. Also, hold the received first half of the I-picture in a temporary buffer until the second half of the picture is correctly received.
      **If** missed more than 2 packets (MSN > 2), then update DelB.
      }

```
Else if (SN == P or I1) and (MSN < 3)
{
        Only B-pictures are lost.  Just replace the lost packets with
        "padding" TS pkts and insert them into the queue.
}

Else if SN == B and MSN ==1
{
        Only one B-picture is lost.  Just replace the lost packet with
        "padding" TS pkts and then insert them into the queue.
}

Else
{
        While MSN >= 0
        {
          If ESN == I1, then ESN += 13, MSN -= 13 and update DelB.
          Else if ESN == I2, then ESN +=12, MSN -= 12 and update DelB.
          Else if ESN == P1, then ESN += 9, MSN -= 9 and update DelB.
          Else if ESN == P2, then ESN += 6, MSN -= 6 and update DelB.
          Else if ESN == P3, then ESN += 3, MSN -= 3 and update DelB.
          Else, ESN += 1 and MSN -= 1.
        }
        Replace the lost packets with "padding" TS pkts.
}
```

The receiver inserts all the correctly received packets into its queue. The lost packets are replaced by "padding" TS pkts and they are inserted into the queue. The receiver keeps track of the number of the audio and video TS pkts inserted into the queue (inserted_AC and inserted_VC). From the received packet, the SN, AC1, AC2, VC1 and VC2 values are obtained. One or more packets are lost when SN > ESN. Total of num_audio_ts_pkts (= AC1 – inserted_AC) audio TS packets are lost and num_audio_ts_pkts amount of "padding" TS pkts, with the audio source **PID**, should be inserted into the queue. Similarly, for num_video_ts_pkts = VC1 – inserted_VC, num_video_ts_pkts amount of "padding" TS pkts, with the video source **PID**, should be inserted into the queue. If SN < ESN or SN == delB, then the received packet should be replaced with "padding" TS pkts. In this case, num_audio_ts_pkts = AC2 – AC1 and num_video_ts_pkts = VC2 – VC1. Another important field in the "padding" TS pkt is the continuity_counter field. In the case of replacing an audio TS pkt, inserted_AC % 16 should be placed in the continuity_counter field of the "padding" TS pkt, and inserted_VC % 16 when replacing a video TS pkt. Note that the inserted_AC value increases by one whenever an audio TS pkt is inserted into the queue. The same holds for the inserted_VC value.

Below is the diagram that shows the steps describing the algorithm for the receiver. Basically, the receiver only displays correctly received pictures that do not depend on lost pictures. All the other pictures (lost pictures and the pictures that depend on the lost pictures) are replaced by the "padding" TS pkts. The receiver makes sure that the continuity_counters TS pkts are increased uniformly.

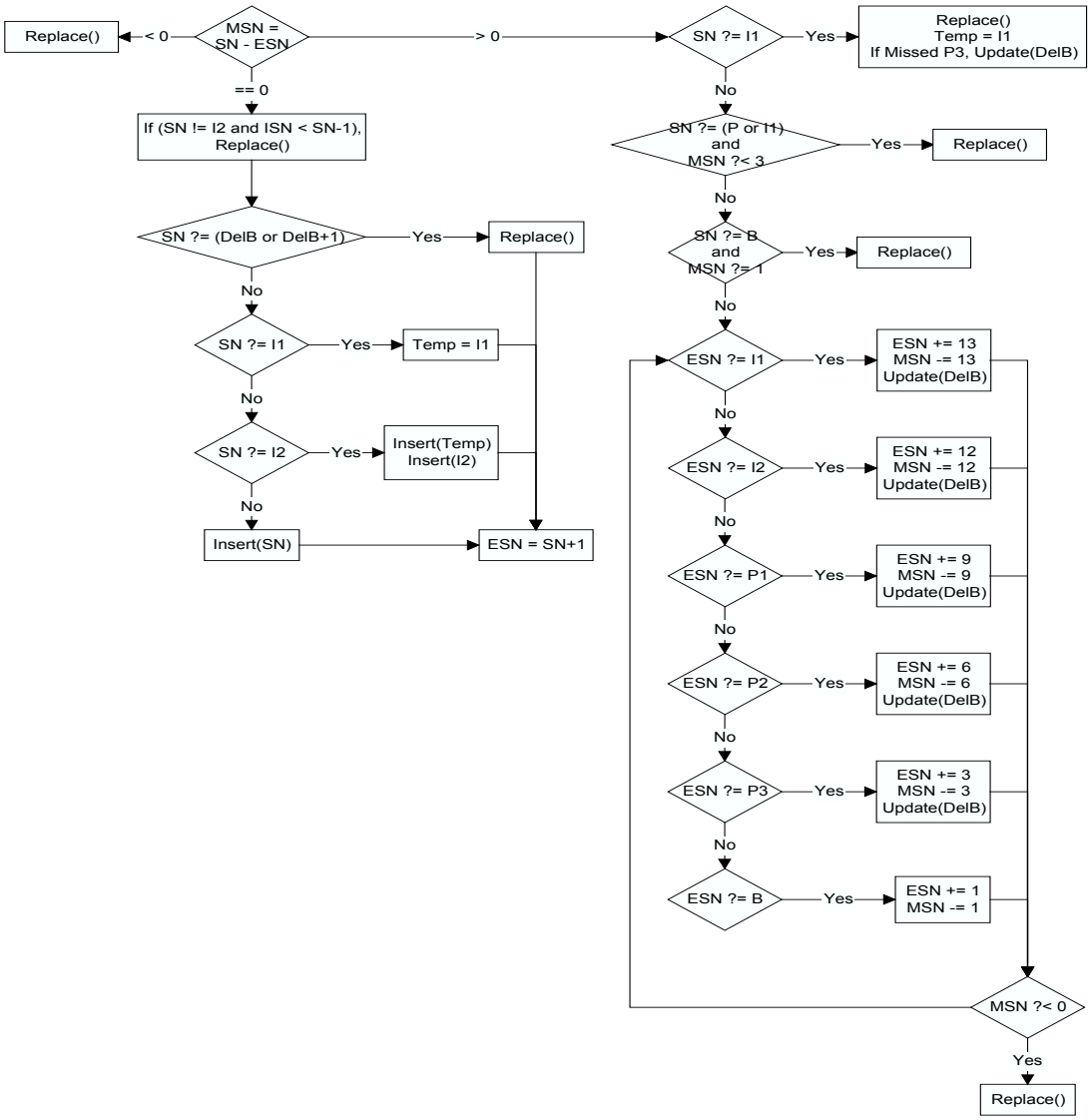


**Figure 10: The Reconstructor – The algorithm replaces the lost pictures and all the pictures that depend on the lost pictures with "padding" TS packets for error concealment. Replace(): Replaces the lost packet with "padding" TS pkts. Update(delB): Updates the value of delB. Insert(SN): Inserts the packet with specified SN into the queue.**

## 7. Conclusion, Future Research Topics and Applications

Use of wireless devices and applications is increasing. There is significant on-going research to make all the services that run on wired networks also work in wireless networks with the same or close to the same QoS. Also, many audio/video applications are using one of the MPEG standards for compression/decompression. MPEG-2 Transport Streams are being utilized in digital TV applications, ubiquitously. Thus, it can be foreseen that in the near future there will be many wireless devices that can send/receive audiovisual data encoded in MPEG-2 Transport Stream.
One main disadvantage of using a wireless network compared to a wired network is that the wireless network can never be error-free. Even with very powerful and high bandwidth network devices, errors can never be avoided. Thus, it is very important to come up with error protection, correction, and concealment methods for wireless applications. In this paper we focused on a wireless application that transfers MPEG-2 Transport Stream. We described the errors and their effects in the application.

One way to conceal the errors caused by packet loss in a real-time application is to show only the correctly received pictures that are independent of the lost pictures. Packetizing the MPEG-2 Video stream and applying the error concealment mechanism is fairly easy task. In the case of the MPEG-2 TS stream, the same mechanism is trickier to apply. In this paper, heuristically derived algorithms for the error concealment mechanism are proposed. The test results show that the proposed algorithm works well, however, in some instances, noticeable discontinuities in the display were seen. This undesirable event occurs whenever I or P-pictures are lost. Thus, applying the error concealment mechanism is the same as trading off distortions with discontinuities. In the case of just B-picture losses, discontinuities were very hard to notice, thus, applying the error concealment mechanism was beneficial. Nevertheless, the experiment results in Section 4 show that I or P-pictures are equally-likely to be lost as the B-pictures.

The error concealment mechanism mentioned in this paper is useful in getting rid of the distortions in the display. Discarding one or two pictures in every second (30 pictures) do not have much visual effect. However, if an I or P-picture is lost, then the whole sequence of pictures that depend on the lost picture must be discarded. This may cause noticeable discontinuities in the display. To make the error concealment mechanism work more effectively, the following can be applied:

- Transmit each anchor picture multiple times so that the probability of its loss can be decreased.
- Allow retransmission for the anchor pictures
- Have more anchor pictures in the GOP sequence
- Interleave the anchor pictures so that loss of any one packet will not result in the loss of the whole anchor picture.

## Appendix A: The "padding" TS pkt used for error concealment

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 47 | 00 | 11 | 04 | FF | FF | FF | FF |
| 8 | FF | FF | FF | FF | FF | FF | FF | FF |
| 16 | FF | FF | FF | FF | FF | FF | FF | FF |
| 24 | FF | FF | FF | FF | FF | FF | FF | FF |
| 32 | FF | FF | FF | FF | FF | FF | FF | FF |
| 40 | FF | FF | FF | FF | FF | FF | FF | FF |
| 48 | FF | FF | FF | FF | FF | FF | FF | FF |
| 56 | FF | FF | FF | FF | FF | FF | FF | FF |
| 64 | FF | FF | FF | FF | FF | FF | FF | FF |
| 72 | FF | FF | FF | FF | FF | FF | FF | FF |
| 80 | FF | FF | FF | FF | FF | FF | FF | FF |
| 88 | FF | FF | FF | FF | FF | FF | FF | FF |
| 96 | FF | FF | FF | FF | FF | FF | FF | FF |
| 104 | FF | FF | FF | FF | FF | FF | FF | FF |
| 112 | FF | FF | FF | FF | FF | FF | FF | FF |
| 120 | FF | FF | FF | FF | FF | FF | FF | FF |
| 128 | FF | FF | FF | FF | FF | FF | FF | FF |
| 136 | FF | FF | FF | FF | FF | FF | FF | FF |
| 144 | FF | FF | FF | FF | FF | FF | FF | FF |
| 152 | FF | FF | FF | FF | FF | FF | FF | FF |
| 160 | FF | FF | FF | FF | FF | FF | FF | FF |
| 168 | FF | FF | FF | FF | FF | FF | FF | FF |
| 176 | FF | FF | FF | FF | FF | FF | FF | FF |
| 184 | FF | FF | FF | FF | | | | |

**Figure 11: A "padding" TS pkt.**

The above figure is the hexadecimal structure of a "padding" TS pkt. The numbers in the first column represent the byte positions of the first bytes in their corresponding rows. The very first byte of the "padding" TS pkt, 0x47, is the **sync_byte**. Then the next three bits are the **transport_error_indicator**, **payload_unit_start_indicator**, and **transport_priority**. For the "padding" TS pkt, the three bits are all set to 0b0 ("0" in binary). Then the next 13 bits represent the **PID**. The value of the **PID** for each source is set in the encoder, and from the **PID** value, the receiver distinguishes between an audio TS pkt and a video TS pkt. In the case where there are other sources such as user private data and secondary video source, then unique **PID**s will be added to the sources. The fourth byte of the "padding" TS pkt is comprised of the **transport_scrambling_control** (2 bits), the **adaptation_field_control** (2 bits), and the **continuity_counter** (4 bits). The **transport_scrambling_control** and the **adaptation_field_control** are both set to 0b00. The **continuity_counter** is modified for each "padding" TS pkt so that it increases uniformly for each source. The "padding" TS pkt in Figure 11 is for a source with **PID** = 0x0011 and **continuity_counter** = 4 [8, 9].

***References***

1 M. Henry, U. Hengartner, P. Steenkiste, and Thomas Gross. *MPEG System Streams in Best-Effort Networks*. In Proceedings of Packet Video 1999.

2 S. Cen, P. Cosman, and F. Azadegan. *Decision Trees for Error Concealment in Video Decoding*. In Proceedings of the Data Compression Conference, IEEE, 1998.

3 B. W. Wah and X. Su. *Streaming Video with Transformation-Based Error Concealment and Reconstruction*. In Proceedings of the IEEE International Conference on Multimedia Computing and Systems Volume I, IEEE, 1998.

4 R. Hasimoto-Beltran and A. A. Khokhar. *Dynamic Adaptive Forward Error Control Framework for Image Transmission Over Lossy Networks*. In Proceedings of the International Conference on Information Technology: Coding and Computing, IEEE, 2000.

5 P. Salama, N. Shroff, E. J. Delp. *A Fast Suboptimal Approach to Error Concealment in Encoded Video Streams*. In Proceedings of the 1997 International Conference on Image Processing, IEEE, 1997.

6 B. Solaiman, R. Pyndiah, O. Aitsab, and Ch. Roux. *Transmission Errors Recovery using Fuzzy Block Similarity Measures*. In Proceedings of the 1997 International Conference on Image Processing, IEEE, 1997.

7 K. Shuaib, T. Saadawi, M. Lee, S. Gringeri, and B. Bash. *A Cell Loss Padding Technique for the Transport of MPEG-2 Transport Stream over ATM/AAL*. In Proceedings of the 1997 International Conference on Intelligent Information. IEEE, 1997.

8 B. G. Haskell, A. Puri, and A. N. Netravali. *Digital Video: An Introduction to MPEG-2*. Chapman & Hall, 1997.

9 ISO/IEC IS 13818, ITU-T Recommendation H.262, "*Information Technology – Generic Coding of Moving Pictures and Associated Audio Information*."

10 David K. Fibush. *Testing Multimedia Transmission Systems*. IEEE Design & Test of Computers 1995.

11 Bob Quinn and Dave Shute. *Windows Sockets Network Programming*. Adisson – Wesley Publishing Company, Inc., 1996.

12 T. ShanaBleh and M. Ghanbari. *Interframe Loss Concealment Techniques for Bursty Packet Losses in IP Environments*. In Proceedings of Packet Video 2000.