# Quality of Service for Video Stream Over IP Networks

## Network Impairment Emulator For Windows Media Streaming Video

Xinying Liu

## Abstract

Multimedia networking is one of the most exciting developments in today's Internet. Streaming technology, which allows the player to start playing audio/video data immediately instead of waiting for the entire file to be downloaded, presents an attractive vehicle for the distribution of multimedia content over the Internet. However, transmitting real-time streaming audio/video across a network is a very demanding task in that it mandates significant bandwidth and quality-of-service (QoS). Due to the underlying protocols, today's Internet technologies support real-time services only in a best-effort manner. The qualities of the audio/video are very sensitive to the network impairments such as packet losses, bit-errors, delays and jitters. To understand the challenges that currently face streaming video delivery, this project investigates the impact of network conditions to video QoS. Specifically, a network traffic model to simulate the network communication problems has been established based on a 2-state Markov model - Gilbert model. Integrated with Windows Media technologies, a network impairment emulator was implemented. These software tools provide a way to better understand the relationship between transmission parameters vs. video quality of service in an emulation environment, which may not be easily analyzed over the real IP networks. These relationships would give us insights for deriving optimal criteria to improve the QoS of video streaming.

*Keywords and Phrases*: Internet, IP networks, streaming video, quality of service, Windows Media, packet capture, network impairment emulator, Internet traffic model

## 1. Introduction

Multimedia networking is one of the most exciting developments in today's Internet that offers enriched communications. Powerful computers and network connectivity present opportunities to deliver real-time audio/video across the Internet to personal computers and digital televisions. Streaming technology, which allows the player to begin playing audio/video immediately instead of waiting for the entire file to be downloaded, is an attractive vehicle for multimedia content distribution. However, transmitting real-time streaming audio/video across a network is a very demanding task in that it mandates significant bandwidth and quality-of-service (QoS). Due to the underlying protocols, today's Internet technologies support real-time services only in a best-effort manner; the perceptual quality of the video is very sensitive to the network impairments such as packet losses, bit-errors, delays and jitters. Poor pictures with jerky and missing frames are common when network condition goes bad.

In addition to network conditions, multiple other factors may affect the QoS of video streams, including (but not limited to) encoder configurations, video compression techniques, streaming format, error concealment techniques and the complexities of the video clip itself.

To better understand the relationship between these parameters to the QoS acquired by the end-user, the idea to develop a simulation environment was raised up by the Video QoS (VQoS) group in the Video Business Unit (VBU) of Tektronix, Inc. The simulation tools, as we call it "Network Impairment Emulator (NIE)", should work for Microsoft Windows Media. It must have the capability to generate packet loss and other bursty random errors that occur in the real Internet. In addition, the defected video under different conditions should be played back repeatedly for further measurements and testing purposes.

The relationships between transmission parameters vs. stream video QoS have been discussed in many papers. Packet-loss effects on MPEG video sent over the public Internet were studied in [13]. The effects of jitter on the perceptual quality of video were studied in [14]. The error resilient video coding, error-concealing schemes had been explored in [2], [3], and [4]. However, none of them have studied on how these factors affect Microsoft Windows Media video.

### 1.1 Traffic Model

To emulate stream video on IP networks, it is important to select an appropriate traffic model. Constructing traffic models that statistically characterize the network traffic is one of the most interesting topics in the network research area. The most important requirement for a model is its accurate agreement with experimental data. Other desirable features are its generality and its simplicity.

A large number of researchers have proposed their models for various types of networks and different application fields. One of the most widely used models describes the traffic fading as a Gaussian process [5]. However, this model is difficult to use in applications, as stated in [6] that "there are no closed form expressions for characteristics associated with the model, such as the probability density function (PDF) of fade durations and the probability distribution of the number of fades inside a fixed time interval".

Another extensively used model in literature is the Hidden Markov Model (HMM). HMM is a powerful tool for modeling stochastic random process. Gilbert [7] initiated the study of HMM for real communication channel error statistics by using a two-state Markov model in 1960. Elliot [13] generalized Gilbert's model slightly in [13]. Many papers use this model to describe burst errors in the network. [2], [9], [12]. The reasons for its popularity are its relative simplicity and its fairly good approximation to the burst packet-loss behavior in the network. It allows us to calculate many important system parameters (such as probabilities of various error sequences and other performance characteristics) in closed form [10]. Reference [18] gave a verification of the first-order Markovian assumption in a Rayleigh fading environment. Besides Gilbert model, there are a lot of other channel models based on Markov chain, which consist of finite or infinite number of states with defined transition probabilities. A review of these models was given in [11].

In this paper, we adopted Gilbert model in our network impairment emulator. The following figure depicts the model.

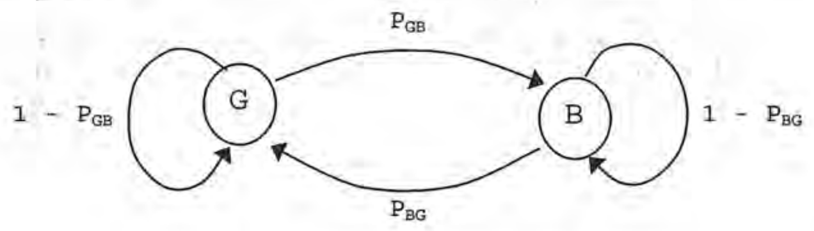


Figure 1. Gilbert Model
A 2-state Markov model to approximate the bursty packet-loss behavior in the noisy channel

The two states in this model are state G (good), where the packets are received error free, and state B (bad), where the packets are lost either due to the network congestion or due to exceeding the maximum allowed transmission delay. The model is fully described by $P_{GB}$, the probability of transition from state G to state B, and $P_{BG}$ the probability of transition from state B to state G. The state transition probability matrix T is:

$$T = \begin{pmatrix} 1 - P_{GB} & P_{GB} \\ P_{BG} & 1 - P_{BG} \end{pmatrix}$$

Since these parameters are not intuitive, we use instead the average packet loss probability, which is given by

$$P_B = \Pr(B) = \frac{P_{GB}}{P_{GB} + P_{BG}} \qquad (1)$$

and the average burst-length,

$$L_B = \frac{1}{P_{BG}}, \tag{2}$$

which is the average number of consecutively lost packets. [4]

## 1.2 Windows Media Basic

Microsoft Windows Media Technology was selected as our first studying goal because its streaming format – Advanced Streaming Format (ASF) is one of the most widely adopted and extensively deployed format on the streaming media market.

Due to the close relationship between our NIE and Windows Media, it is necessary to have a brief introduction to the Microsoft Window Media technology. Microsoft Windows Media Technology is a family of products and services from Microsoft that give you the ability to create, deliver, and play streaming files in the Advanced Streaming Format. The primary components of the Windows Media Technologies are:

- Windows Media Tools: Create streaming media, including encoder, authoring tools, and format converting tools;
- Windows Media Server: Host the deliver streaming media. It supports unicast, multicast and provides monitoring services.
- Windows Media Player: Decoder, the client application that play streaming media

These provide an end-to-end solution for streaming multimedia, from content authoring to delivery to playback.

Microsoft Media Server (MMS) Protocol is the default protocol from Windows Media (WM) server to Windows Media for unicast service. MMSU is the MMS protocol combined with UDP (User Datagram Protocol) data transport. MMST is the protocol based on TCP (Transmission Control Protocol). Windows Media server also supports HTTP (Hypertext Transfer Protocol) protocol to deliver streaming media through a firewall because HTTP streaming always use port 80 and most firewalls do not block port 80.

## 1.3 Video Compression

Video compression scheme plays an essential role in the stream video process. Without compression, the extremely huge audio/video content is almost impossible to be broadcasted over common Internet bandwidth. There are several major video compression standards, including H.261, H.263, MPEG-1, MPEG-2, MPEG-4, etc. The algorithms to do the compression and decompression are called codecs. In Windows Media Encoder 7, Microsoft Windows Media Video version 7.0 codec is used for most encoding scenarios. It also provides Microsoft MPEG-4 version 3.0 codec for Microsoft MPEG-4 and ISO MPEG-4 video codec for ISO compatible MPEG-4.

MPEG is an acronym for the Moving Picture Experts Group, which was set up by ISO (International Standards Organization) to work on compression. MPEG video compression techniques eliminate both the spatial redundancy in a video frame and the temporal redundancy in successive video frames. MPEG uses bi-directional coding that allows information to be taken from pictures before and after the current picture. Three different types of pictures are needed to support differential and bi-directional coding while minimizing error propagation:

- I-frames: Intra-coded pictures that need no additional information for decoding.
- P-frames: forward Predicted from an earlier I- or P- frame.
- B-frames: Bi-directional predicted from earlier or later I- or P- frames.

Normally, I-frames require more bits than P-frames, whilst B-frames have the lowest bandwidth requirement. A typical video stream frame sequence goes like IBBPBBPBBPBB IBBPBBPBBPBB ...

Apparently, packet loss occurs on a P- or a B- frame might have relatively small effects on the quality of the video, whereas packet loss occurs on an I-frame may severely degrade the perceptual video quality. The errors may propagate to subsequent frames as well.

The rest of the document is laid out as follows: Section 2 gives an overview of the project; Section 3 describes the methodologies used in constructing our Internet traffic model and in developing the network impairment emulators; Section 3 outlines the implementation of the software tools; Section 4 describes the experiments and analyzes the results; Section 5 summarizes our conclusion and future works.

## 2. Overview

The work in this project can be described in 2 major parts. One is constructing the traffic model that characterizes the noise channel of the Internet. The other is to develop the network impairment emulator based on the model and parameters from the first part.

In the first part, we construct the traffic model in 3 steps, as shown in Figure 2.

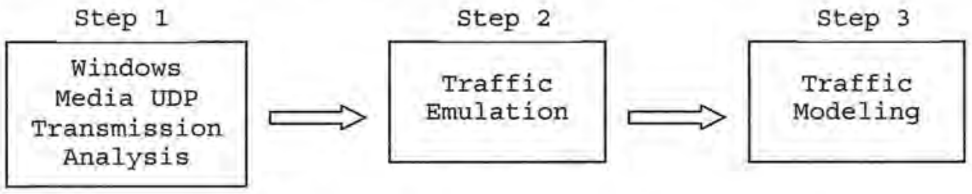


Figure 2. Steps to construct traffic Model for NIE

We started from analyzing the Windows Media UDP transmission properties, and use those properties to emulate the traffic in generic UDP socket to simplify problem. From experiments of the traffic emulation, we measure packet loss ratio and average consecutive loss length on the real Internet channel, and obtained parameters for constructing the Gilbert model for different scenarios. In addition, we studied the agreement of the traffic model to the experimental data to check its validity.

In the second part, the network impairment emulator was implement using packet capture utilities and save the video content into ASF files.

In the UDP-based stream video transmission, a lot of errors occur in the Internet can be treated equivalent as packet losses. For example, packets with bit errors are generally dropped if the Cyclic Redundancy Checks (CRC) fails; application programs may also ignore packets that arrive too late. Thus, our NIE put major effort in emulating the packet-loss behaviors.

Figure 3 shows the architecture of our NIE:

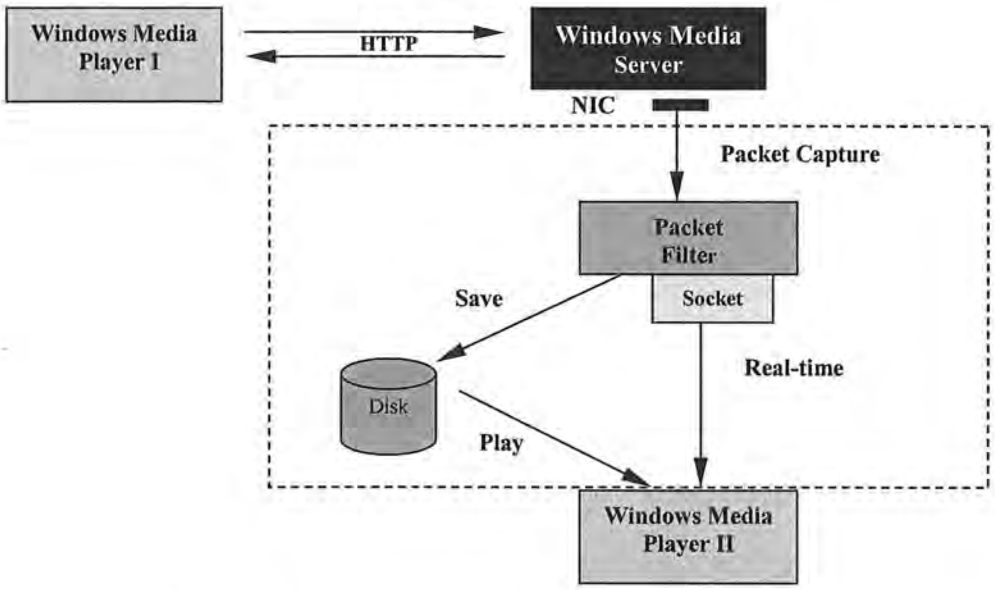


Figure 3. Architecture - Capture HTTP packets to simulate UDP packet loss

In this architecture, while Window Media Server is serving a client (Media player I), we capture the packets at the network adapter, and direct them to our packet filter. In the packet filter, packets are dropped out according to the traffic model. Then the data is saved to disk as an ASF file, which can be played back by Window Media Player II. The part included in the dashed line is our implementation of the Network Impairment Emulator.

## 3. Methods

### 3.1 Traffic Model

Of all the transport-layer protocols, our primary concern is the video stream on UDP transport, because UDP is the most natural and efficient way to transmit real-time or on-demand multimedia data. Also because of its connectionless and unreliable transport nature, no retransmission mechanism exists; network errors may directly influence the QoS at the end-users.

The characteristics of the Windows Media UDP video streaming have been studied. Three typical UDP transmissions load that associated with three connecting bandwidths - 56K bps, 100K bps and 300K bps – have been simulated in UDP packet transmitting experiments. We call this part as "network traffic analyzer".

The network traffic analyzer works as follows. UDP packets, with sending timestamps and sequence number wrapped inside, were transmitted between a 'sender' and a 'receiver' in unicast mode. The sender has control over the packet size, sending rate and sending durations. The receiver calculates the transmission delay for each packet, and collects useful statistics such as packet loss probability, average burst length, number of out-of-orders, etc. Since the traffic conditions may vary during the day, the measurements were performed at 6 different hours of the day.

The experimental results are compared with the Gilbert model to verify its accuracy, and to determine the model parameters under different network conditions.

### 3.2 Network Impairment Emulator

Because Window Media use its proprietary ASF format to store and deliver encoded media content, we have little knowledge about how the video data are encoded, packetized and streamed out of the Windows Media Server in the low level. Neither did we know how the receiving packet are buffered, ordered and rendered at the client application, Windows Media Player. Thus, two major technical challenges that faced with this project were: (1) How do we capture the stream video packets? And at what Internet hierarchy layer should the packet-capture be conducted? (2) How can we force the Windows Media Player play the defect content generated by our NIE?

#### 3.2.1 Packet Capture

In our project, stream video packet-captures are performed below the IP layer, since we want to capture and drop the packets as if they were lost on the physical wires of the network, where packet-loss really occurs.

The tool we used to capture packet is Winpcap. WinPcap is an architecture for packet captures and network analysis for the Win32 platforms, based on the model of BPF (Berkley Packet Filter) and libpcap, which are famous on UNIX. WinPcap includes a kernel-level packet filter driver, a low-level dynamic link library (packet.dll), and a high-level and system-independent library (libpcap). (WinPcap are available for free from the University Politecnico di Torino, Italy) [6]

With the aid of WinPcap, we have the capability to capture packets from the network adapter or Network Interface Card (NIC) in a promiscuous mode. In addition, WinPcap provides packet filters that help us filter out the packets that we are not interested in. For example, a filter as "`udp and src host xxx.xxx.xxx.xxx and dst host yyy.yyy.yyy.yyy and dst port zzzz` " will only capture the UDP packets from source IP address `xxx.xxx.xxx.xxx` to destination IP address `yyy.yyy.yyy.yyy` and destination port `zzzz`.

Large video data packets are very likely to exceed the MTU (Maximum Transfer Unit) of the underlying link (MTU of Ethernet is 1500 bytes). Therefore, an important consideration in packet capture process is the IP fragmentation.

In the packet-switching Internet, when a source host or a router transmit an IP (Internet Protocol) datagram on a link where the MTU of the link is less than the IP datagram's size, the IP datagram must be fragmented. This allows IP nodes to connect regardless of different MTUs in intermediate network segments and without user intervention. When IP fragmentation occurs, the IP payload is segmented and each segment is sent with its IP header. The IP header contains information required to reassemble the original IP payload at the destination host. Apparently, IP fragmentation and reassembly presents an expensive overhead - both at the routers (or sending hosts) and at the destination host. On the modern Internet, fragmentation is highly discouraged; Internet routers are busy enough with the forwarding of IP traffic. Fragmentation can be avoided in TCP transport by including the MSS (maximum segmentation size) value in the option field of the TCP header during the TCP connection setup process, SYN; whereas in UDP transport, fragmentation is generally hard to avoid because its connectionless nature.

If packets are found to be fragmented, care must to be taken in re-assembling the packets.

### 3.2.2 ASF file playback

From the NIC card, the packets are tapped out and forwarded into the packet filter, where packets are dropped according to the traffic model to simulate packet-loss. Basically, there are two approaches to playback the ASF data stream coming out of our packet filter. One is to save the data into a file and play it back thereafter; the other is to force the Windows Media Player play it in real time. The method we implemented in NIE is the first approach. The advantage of it is that you have the copy of the defect video file; you can test and measure the quality deterioration repeatedly at any time. The second approach is desirable in its real-time feature, but it is hard to achieve because the close system of Windows Media Player does not provide such flexibility to validate and open a stream coming out of a source other than Windows Media Server.

Saving the stream video data into a valid ASF file requires the knowledge about the format of the ASF file. As we mentioned before, ASF format is the proprietary data format for Window Media to store and stream audio/video content. Its specifications are unknown to public. However, we can get some important information about the file and the stream format from both ASFRecorder [7] and our own experiments using our packet-dump utilities.

ASF stream is composed of three parts, the header, the data and the end parts. The header contains essential information about this particular ASF file, including the header-length, total file size, total number of packets, *ASF chunk size* and some time information. The header may also contain the script of the conversation in the video clip, which makes the header very large. Large headers may be packetized into multiple packets with rising sequence numbers.

Following the header packets are the data packets. Each data packet has a small header that includes the packet length, sequence number and part flag information. Video data start after the header. Each data packet with a unique sequence number is mapped to a chunk of data in the ASF file. The concept of a data chunk in the ASF file is a fixed length of space. The *ASF chunk size* is given in the ASF header, which is normally larger than the maximum size of any data packets. The lengths of the data packet vary. If the length of the data packet is less than the *ASF chunk size*, 0's stuff up the rest spaces.

The end packet indicates the end of the streaming.

The process to save ASF stream into ASF file can be show in the Figure 4.

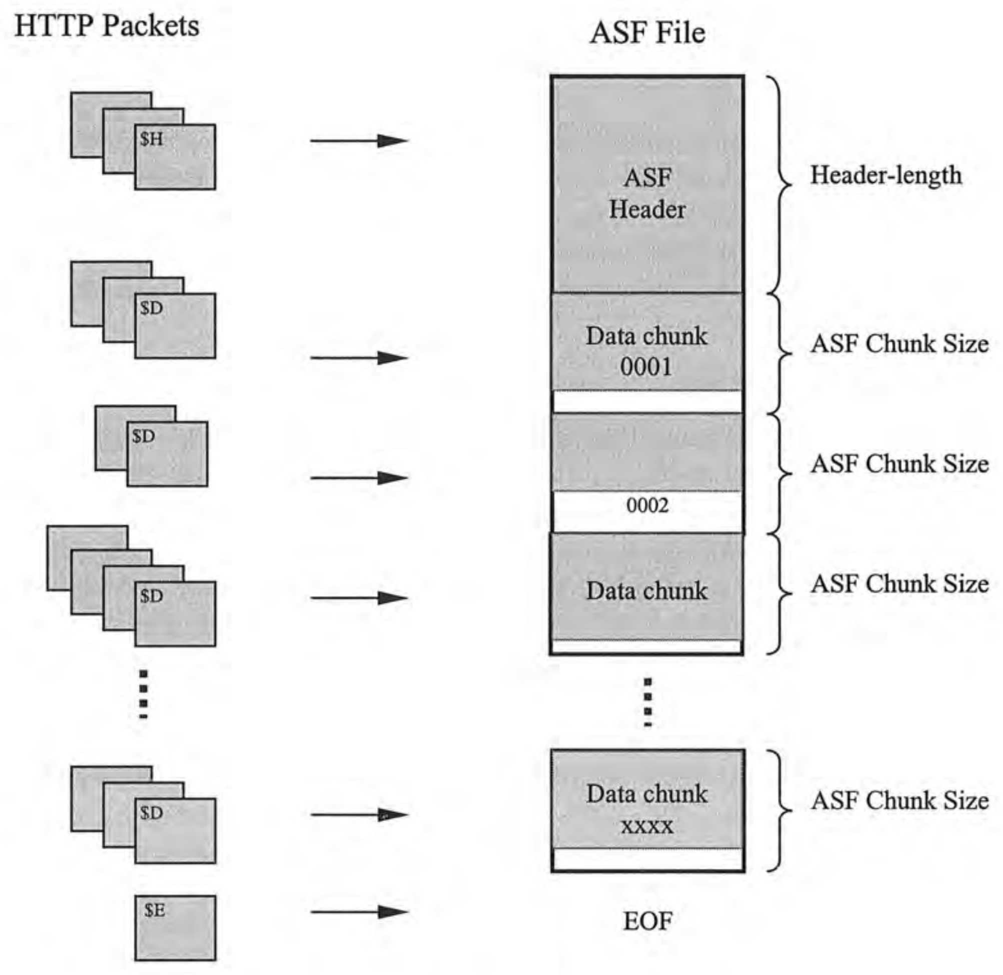


Figure 4. How to save ASF stream into ASF file on disk

## 3.2.3 HTTP vs. MMSU

If you are careful, you may have noticed that in our architecture (Figure 2) the protocol used between Windows Media Server and Window Media Player I is HTTP, instead of MMSU. You may argue that now that you are interested in UDP transmission, why your network impairment emulator drops the HTTP packets, instead of the UDP packets? Well, we call this approach as "capture and drop HTTP packets to simulate UDP packet-loss". What justify us to do this is based on our analyses of the Window Media HTTP transmission and MMSU transmission using our packet-dump utilities.

Table –1 lists the comparisons of Windows Media HTTP transmissions vs. MMSU transmission.

| **Protocol** | | **HTTP** | **MMSU** |
|---|---|---|---|
| **Connection Setup** | | Standard HTTP request/response | Microsoft's proprietary MMS protocol to setup the request/ response using TCP. Details unknown. |
| **Transport Protocol** | | TCP | UDP |
| **Packeti-zation** | **Starting flag** | Head Chunk: '$H'<br>Data Chunk: '$D'<br>End Chunk: '$E' | None |
| | **Data chunk size** | Same<br>For a particular ASF file | |
| | **Data chunk header length** | 12 bytes | 8 bytes |
| | **Data chunk header format** | Different | |
| | **Data chunk header content** | Similar. Both include packet length, sequence number, etc | |
| **Fragmentation (if applicable)** | | Conducted at the sender | Conducted at bypassing routers |
| **Able to go through firewall?** | | Yes | No |

Table 1 – Comparison of Windows Media HTTP transmission vs. MMSU transmission

In Windows Media Sever, HTTP transmission and MMSU transmission both stream from a same ASF file. (ASF files are encoded specially for streaming purposes.) Though there are quite a lot of differences between these two types of transmissions, the commonalities in the packetization of the video data enable us to treat HTTP packets equivalent as UDP packets, if the protocol-specific headers are removed. In other words, we are only interested in the video data part of the packets when write them into a file. Only how the data is packetized and how much data is wrapped in the packet matter when we consider packet losses.

The approach to 'capture and drop HTTP packets to simulate UDP packet loss' takes the advantage that HTTP is a simple and open protocol. And the characteristic starting letters - "$H", "$D" and "$E" - give us good indicators of the packet type and boundary information. Additionally, the capability of HTTP transmitted stream to go through the corporate network firewalls enable us to connect to the outside world.

# 4. Implementations

## 4.1 Implementation of network traffic analyzer

The network traffic analyzer is implemented in C using Unix BSD socket and Winsock. It is portable on SunOS, Linux and Win32 platform.

The network traffic analyzer consists of two programs: sender and receiver. The sender generates and sends out UDP packets at a constant rate with a rising sequence number. The first 8 bytes of the UDP packet store the sending timestamp, which is of type timeval.

```
struct timeval
{
      long sec;          // in seconds
      long usec;         // in microseconds
};
```

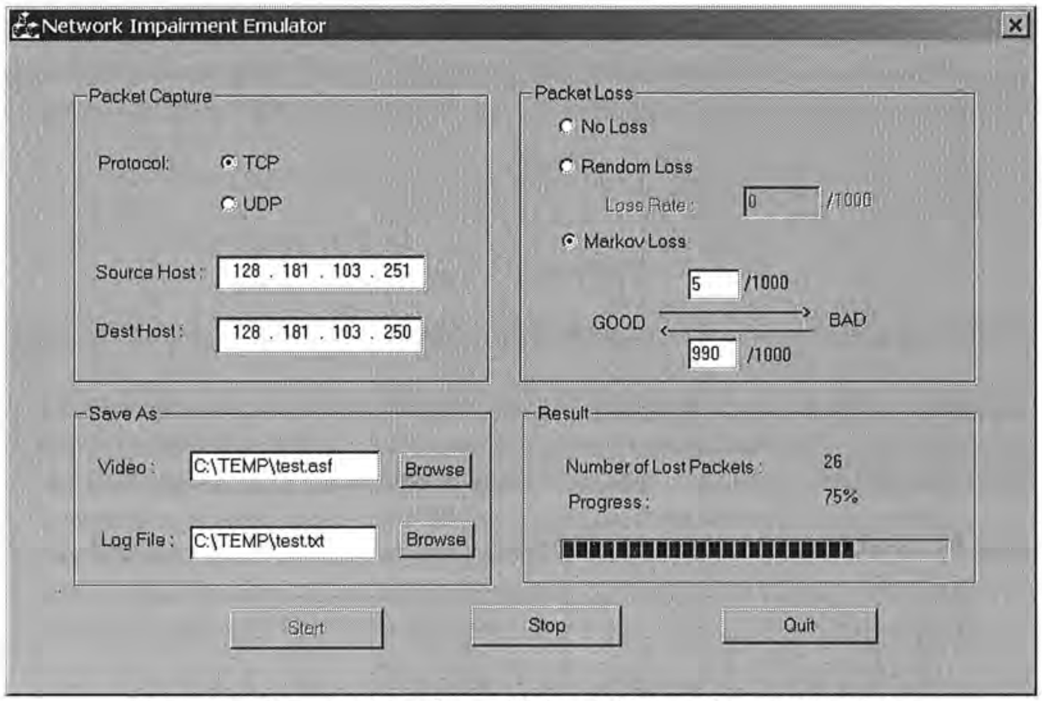
The rest of the packet is stuffed with the sequence number.

The receiver receives the UDP packets from the sender. It retrieves the sending time-stamp from the packet and gets the current system time. The latency between sender and receiver can be calculated. Since there is no synchronized clock between two remote hosts, the resulting latency is just a relative measure on the delay of the packets. The arrive/loss status and the orders of the arriving packets can be easily obtained from the packet sequence number. After the sender stops sending, the receiver starts doing the statistics and gives a report like the following:

```
**********************************************************************
                  REVEIVER REPORT

Start @ : <02:37:34.200000>     End @ : <02:40:54.498000>

        Total Packets Sent       = 2000
        Total Packets Received   = 1990
        Total Time Elapsed       = 200.298 sec
        Average Delay            = 99.340316 sec (not absolute value)
        Number of Lost Packets   = 10
        Out-of-Orders            = 0
        Number of burst          = 8
        Avg consecutive pkt loss = 1.25
        Packet Loss Rate         = 0.5%

Lost packets are:
              190              1
              253              1
              414              1
              640              1
              651              1
              1449             1
              1632 - 1633      2
              1937 - 1938      2

Burst Length Distribution:
        Burst length = 1          count = 6
        Burst length = 2          count = 2
**********************************************************************
```

## 4.2 Implementation of the Network Impairment Emulator

The network impairment emulator is implemented using Visual C/C++, and runs on Windows platform. The packet capture part uses the functions and libraries provided by WinPcap.


Network Impairment Emulator

Packet Capture

Protocol: [✓ TCP] UDP

Source Host: 128 . 181 . 103 . 251

Dest Host: 128 . 181 . 103 . 250

Packet Loss

No Loss

Random Loss

Loss Rate: 0 /1000

[✓ Markov Loss]

GOOD → BAD: 5 /1000

BAD → GOOD: 990 /1000

Save As

Video: C:\TEMP\test.asf

Log File: C:\TEMP\test.txt

Result

Number of Lost Packets: 26

Progress: 75%

Figure 5. The user interface of Network Impairment Emulator

Figure 5 shows the graphical user interface (UI) of NIE, which is implemented using Microsoft Foundation Class (MFC). It is designed to be a user-friendly dialog-based UI that allows user to input the packet-capture filter information, select packet-loss model (random loss or Markov model) and parameters, and specify destination location of the ASF and log files. After clicking the 'Start' button, a message will prompt you to start the Windows Media player I. The stream data will be captured, dispatched and saved into a file. A progress bar shows the progress status of the transmission. When the transmission completes, a statistic report pops up showing the packet-loss statistics. The packet lost information can be written into another file, and be further analyzed using Microsoft Excel or other statistical tools.

In WinPcap architecture, libpcap is the packet capture library that provides a high level interface to packet-capture system. In the program, we initially call pcap_lookupdev() to look up the network device on the running computer. Then pcap_open_live() and pcap_lookupnet () open the adapter and obtain the subnet and netmask. The user provided filter is compiled and set with pcap_compile () and pcap_setfilter(). pcap_dispatch () is the function used to collect and process packets. In pcap_dispatch (), a callback routine is required as parameter to specify what to do for each captured packet. That is where the packets are re-assembled into ASF files.

In UDP transmissions, large video packets are fragmented at the passing routers. In IP header, the bits indicating "Don't fragment" (DF) (the 21st bit) or "More fragment" (MF) (the 22nd bit) need to be checked for each packet in the dispatch handler. If MF bit is set, it means the packet is fragmented and more fragments would come following this one. Only the first fragment has the UDP header. The consequent fragments would only have IP headers. The whole length of the packet and the offset information in the IP header are useful to re-construct the packet.

In HTTP transmissions, large video packets are fragmented before sending out. Therefore, every packet has both IP and TCP header. However, only the first packet in the ASF data chunk has the 12 bytes header, which includes sequence number and length information. The starting flags at the beginning of each ASF chunk - "$H", "$D" or "$E" - provide good type and boundary information for the re-assembly. If the length of the chunk is less than the given *ASF chunk size* (refer to section 2.3.2), the rest of the space is stuffed with 0's. For example, if the *ASF chunk size* is 5300 bytes. We have a data chunk that is fragmented into 4 segments - A, B, C, and D. The size of these segments are 1448, 1480, 1480, 800 respectively, 5208 bytes in total. In perfect conditions, the data chunk is written in file as ABCD. 92 bytes of 0's would be stuffed at the end of the ASF chunk.

If the first segment of the data chunk is lost, the 12-byte chunk header is lost consequently. Therefore, the whole data chunk is ignored. In other words, the consequent segments won't be written into file though they may arrive successfully. On the other hand, if the first segment of the data chunk arrives successfully, but one or more of the consequent fragments are lost, then the lost part is ignored; the arrived data is sequentially written into file. The missed part is padded with 0's at the end of the chunk. In the previous example, if A, C and D arrive, but B is lost, then it is written as ACD in the file. There would be 1480 + 92 = 1572 bytes of 0's at the tail.

The arrive/lost status of a packet is determined by the traffic model. Two traffic models, random-loss model and Markov-loss model are implemented using random number generators. Specifically, the if-else branches are taken by whether the random number is greater than the specified probability or not.

The process to capture and save the packets can be terminated by user in the middle. To accomplish this, this process is kept in a separate thread from the thread that user click the 'Stop' button. Mutual exclusion is achieved by using critical sections, which prevent multiple threads from accessing sharing variables simultaneously.

In MFC, the thread function is in the global scope. It calls libpcap routine pcap_dispatch(). Since libpcap is a C library, it requires the callback function, the dispatch-handler, to follow the C call convention. However, to prevent using too many global variables, we prefer to implement the dispatch-handler as a member function in dialog class in order to access member variables more easily. An extra function, the pseudo dispatch-handler, is called by the thread function to comply with the C call convention. The pseudo dispatch-handler calls the dispatch-handler in the dialog class.

Two helper routines, `parse_UDP()` and `parse_TCP()` were written to interpret the IP header and the UDP, TCP header respectively. These information plus the descriptions of some major steps can be written into a log file for debugging purposes.

## 5. Experiments and Results

In this section, we discuss the experiments and results. Section 4.1 describes how we study the Windows Media traffic load. Section 4.2 talks about the experiment to measure packet loss and consecutive loss length in real UDP transmission. From these results, parameters for Gilbert model were derived. Applying those parameters, we compared Gilbert model to experimental data to verify the model accuracy. Section 4.3 briefly discusses the measurement of the picture quality deterioration.

### 5.1 Windows Media Transmission Properties

There are four widely used encoding profiles in Windows Media. They are associated with four common data transmission speed – 28Kbps, 56Kbps, 100Kbps and 300Kbps.

Table –2 shows the typical UDP traffic load in these 4 transmissions.

| Encoding Profiles / Parameters | 28K | 56K | 100K | 300K |
|---|---|---|---|---|
| Avg. packet length (bytes) | 900 | 930 | 1280 | 3779 |
| Sending rate (packets/sec) | 3.0 | 5.0 | 10.0 | 10.0 |
| Avg. bit rate | ~ 21.6K | ~ 37.2K | ~ 100K | ~ 300K |

Table 2. Traffic load in 4 major transmission profiles in Windows Media

Figure 6-8 shows the packet length histograms for a particular video clip in 3 different transmission speeds. From the charts, we find out that the packet lengths are almost constant in each transmission profile.

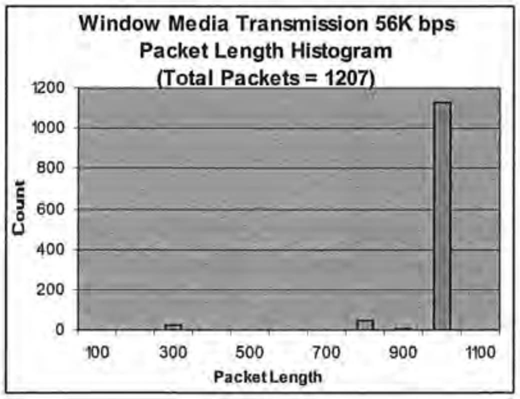


Figure 6

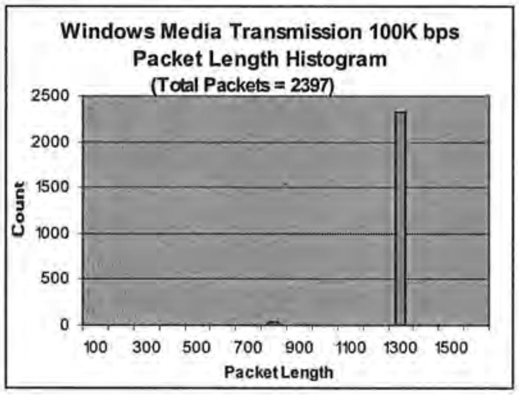


Figure 7

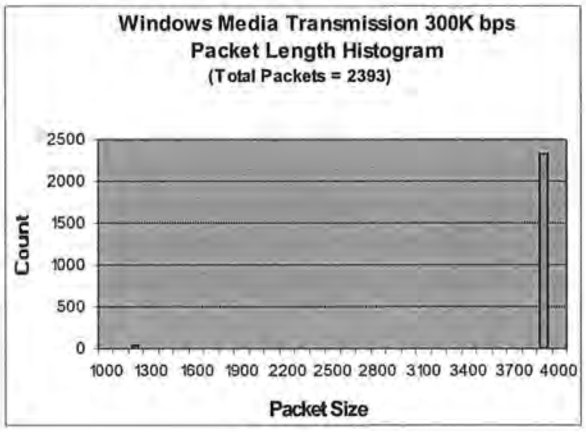


Figure 8

Therefore, we can consider the Windows Media video stream as a constant bit-rate channel. The 100K and 300K transmission take full advantage of the bandwidth with a sending rate of 10 packets / second. Modem users (28K or 56K) over telephone lines generally cannot achieve the upper limit of the bandwidth, thus the video is encoded at some lower bit rate. The packets are sent at lower speeds as well.

In the 300K transmission, the quality of the video is no doubt the best. Both the audio and the pictures are clear. The frame rate can reach about 23-25 fps. In the 28K transmission, the video compression factor is high, thus video quality is heavily degraded. The frame size is smaller than those in higher encoding profiles. The frame rate can only be about 5-8 fps. But the audio quality is still acceptable.

## 5.2 Measure Real UDP Traffic

With the knowledge of the traffic load characteristics of Windows Media, we simulated them with normal UDP traffic to study the packet delivery performance on different connections. Using the 'Sender' and 'Receiver' utilities, we performed UDP transmission between 2 types of host pairs – T3 to a dual T1 transmission line, and T3 to ADSL line. (The bandwidth of T1 and T3 connections are 1.544M bps and around 44Mbps respectively. The downstream and upstream bandwidth of the DSL line is 768K bps and 128K bps).

Because there are more and more home users install ADSL lines or cable modems to access Internet. It is interesting to understand the network conditions over these broadband home connections. For the limitation of space, in this paper we only give out the results in the T3 to ADSL line pair, in which the T3 host served as a sender, the ADSL line as a receiver.

The experiments were designed as follows:

| Experiment | I | II | III |
|---|---|---|---|
| Traffic Load Description | Light | Normal | Heavy |
| Packet size (bytes) | 1000 | 1250 | 3750 |
| Sending Rate (pkt/sec) | 5 | 10 | 10 |
| Bit rate (bps) | 40K | 100K | 300K |

Table 3 – UDP traffic experiments

UDP packets were sent continuously for a period of 60 minutes according to the given packet size and sending rate in Table 3. Experiment I, II and III represent three levels of network traffic load - light, normal and heavy – in the network connection, which we referred as 40K, 100K and 300K. Packet loss rate and average consecutive loss length, and number of out-of-orders were measured in different hours of the day.

Figure 9 and 10 show the comparison of packet loss rate and average consecutive loss length in experiment I, II and III, and their fluctuations during the day. In our results, out-of-order packets occurred very rarely. We can almost ignore their occurrences.

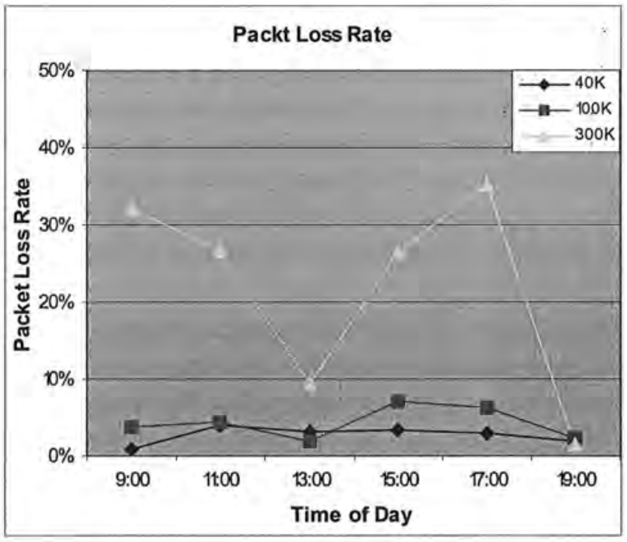


Figure 9 – Comparison of packet loss rate

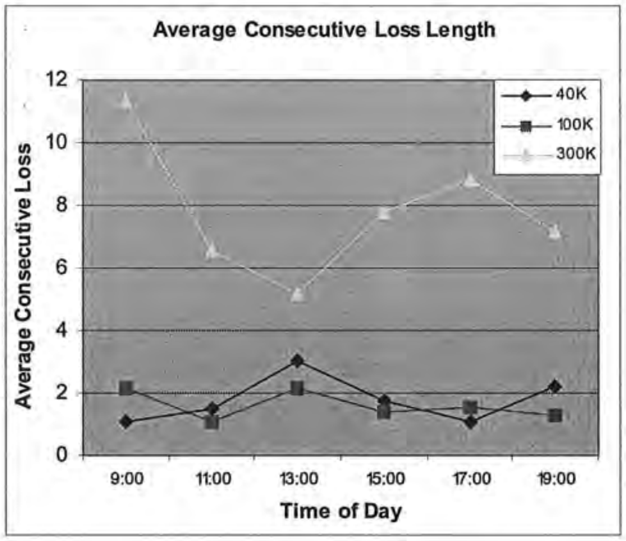


Figure 10 – Comparison of average consecutive loss length

From the charts, we can see the trend is that packet loss rate and the average consecutive loss length increase as the traffic loads increase. The reason is obvious – the heavy traffic adds more burden to the network connection, thus exacerbate the network conditions.

In the cases of 40K and 100K, the traffic loads were low and moderate. The packet loss behaviors were similar. About 1-8% of the packets were lost. The losses were not very bursty, as shown in Figure 6, where the average consecutive loss lengths were around 1 to 3.

In the case of 300K, packets were larger and were sent faster. They had more chance to overflow the router's incoming buffer. Additionally, due to fragmentation, the loss of a single fragment may result in the loss of the whole packets. Therefore, the packet loss rate and the burstness in the 300K-transmission were significantly higher than those in the other two transmissions.

Obtaining these packet-loss statistics under different network conditions, we can derive the optimizing Gilbert model parameters. Specifically, we can calculate $P_{BG}$ and $P_{GB}$ according to formula (1) and (2) using the results we get from experiments.

We list the results of four typical scenarios in Table 4. Each scenario represents a network condition associated with a packet loss rate and an average consecutive loss length.

| | Scenario I | Scenario II | Scenario III | Scenario IV |
|---|---|---|---|---|
| **Description** | A pretty good connection | Slightly heavy packet loss | A noisy channel | A very noisy channel |
| **Packet Loss Rate** | 1% | 4% | 10% | 25% |
| **Avg. Cons. Loss Length** | 1.05 | 1.16 | 1.60 | 3.56 |
| $P_{GB}$ | 0.96% | 3.59% | 6.94% | 9.37% |
| $P_{BG}$ | 95.2% | 86.2% | 62.5 | 28.1% |

Table 4

Applying these parameters to Gilbert model, we can check how this model agrees with experimental data. The metric we chose to do the comparison was the average consecutive loss length histogram. This metric describes the distribution of the packet loss run length generated by the model and by the experiments. Figure 11 ~ 14 illustrate the comparisons for the four scenarios. The results were obtained by sending UDP packets at a rate of 10 packets/second over a period of 200 seconds.

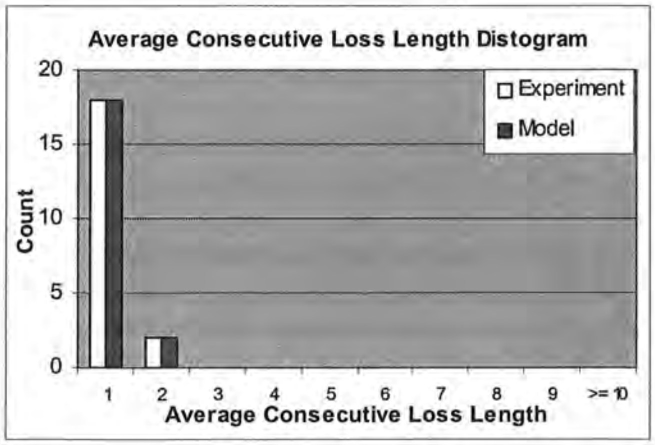


Figure 11 - Scenario I
Packet loss rate = 1%

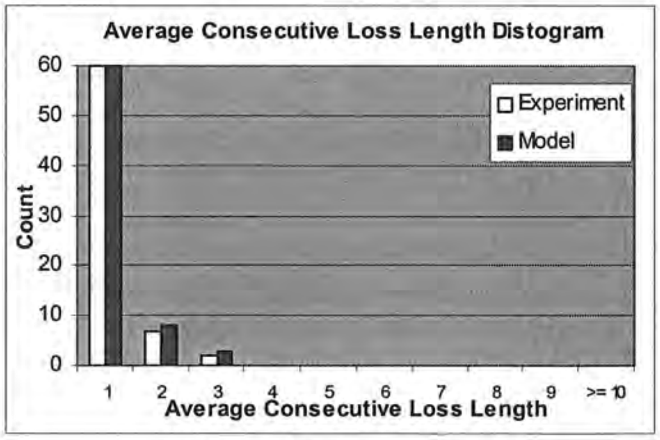


Figure 12 - Scenario II
Packet loss rate = 4%

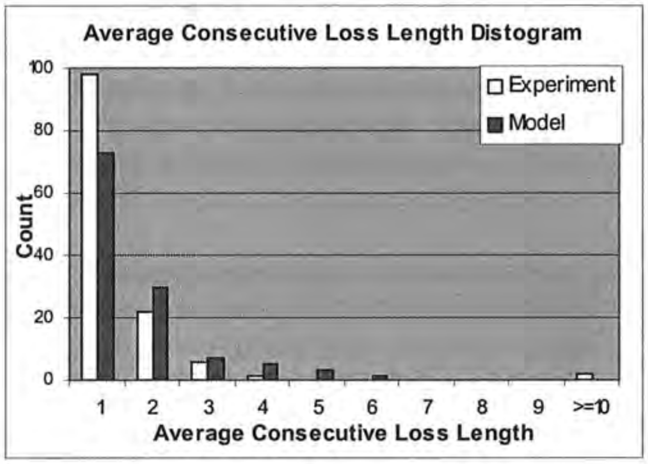


Figure 13 - Scenario III
Packet loss rate = 10%

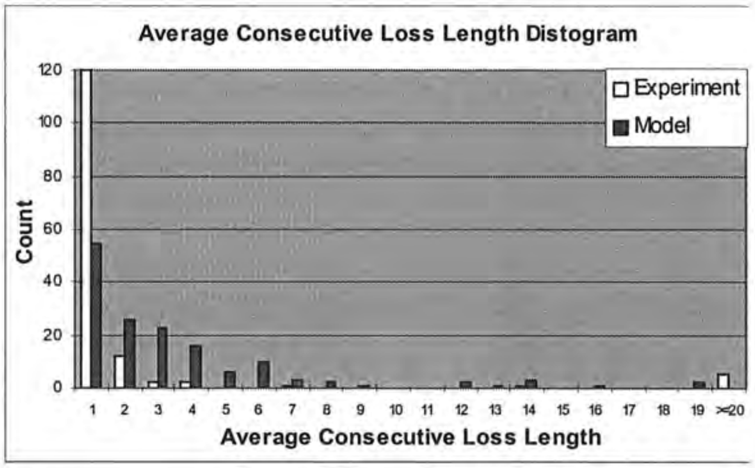


Figure 14 - Scenario IV
Packet loss rate = 25%

We can see from these charts that the Gilbert model agrees with the experimental data well when the packet-loss rates were relatively small. However, at heavy packet-loss conditions, the model gives relatively flat curve in the histogram; whereas in the experimental data, consecutive loss run length of 1 to 4 were in great majority, other lengths were relatively infrequent. Additionally, in experimental data, it was normal to find a long sequence, say, over 50, of packets get lost in a run, which was hardly ever happened in the Gilbert model.

## 5.3 Picture Quality

The measurement of the picture quality deterioration is not part of this project. However, from some simple experiments, we can see that the packet loss can bring freeze frames, blurring, and blocking. Some degradation only occurs in a blink of eye, some other errors may propagate to several seconds. These phenomenons conformed well to the video compression scheme. The picture quality is normally not acceptable when packet loss rate exceeds 5%. From this point of view, using Gilbert model in this project is a good approximation.

## 6. Discussion and Future Work

In this project, we studied the traffic load characteristics of both HTTP and UDP transmission for Microsoft Window Media. Based on this knowledge, we constructed and examined the accuracy of the Gilbert Model, and obtained appropriate parameters for different network conditions.

While Gilbert Model satisfies its ability to generate burst errors when the network conditions are relatively good, it has some limitations so far as its suitability to represent very noisy channels is concerned. The limitations of Gilbert Model arise from its renewal nature and from the assumption of the geometric distribution for run length of G and B [11]. By "renewal", we mean that the error processes generate independent errors. The transition probability renew to the same probability after each transition, which is not the case in most channels. For example, a burst of loss occur in a real channel might be a run length of 20 packets, then 1 packet is received, then another 30 packets of consecutive loss. Normally, we would view this sequence as a burst of length 51. However, in Gilbert model, when the only 1 packet is received, the transition probability is renewed to the same $P_{GB}$ as defined. The odd to have an adjacent 30 packet-losses is very low. From another point of view, in real channels, after a burst has progressed for a considerable length, it is reasonable to expect that the probability of continuation of the burst should decrease, unlike in Gilbert model in which it remains the same.

To overcome the limitation of Gilbert model, more complicated channel models might be considered to more and more accurately describe the real channel conditions. These models might include Fritchman's partitioned finite-state (N states) model, Markov models based on conditional gap distributions, etc.

Based on the traffic Model, we implemented our Network Impairment Emulator for Microsoft Windows Media. The emulator can capture stream video packets and drop them according to the traffic model. The defected video can be saved into file for further analysis and measurement. However, due to the close system of Windows Media, we have little knowledge about how it buffers and renders packets. The video viewed from the saved file might be different from viewing the stream with noise injected in real time. In the next step, we want to setup the real-time connection between our NIE and the Window Media player. However, this step may not be easy without the technical support from Microsoft.

To measure picture quality systematically, there are various approaches. We can use the MSE (mean squared error) and PSNR (peek signal to noise ratio) metrics. There are also algorithms that intelligently compare the target video with a reference, such as HVS (human vision system) model. Subjective scoring is also widely used to evaluate the quality deterioration. The work to determine which approach best fits Windows Media to measure picture quality is left for future work.

## References

[1] James F. Kurose, Keith W. Ross, "Multimedia Networking" in "Computer Networking: A Top Down Approach Featuring the Internet", chapter 6.

[2] B. Girod, K. Stuhlmuller, M. Link and U. Horn, "Packet Loss Resilient Internet Video Streaming" in IS&T/SPIE Conference on Visual Communications and Image Processing 99, pp833-844

[3] Kay Sripanidkulchai and Tsuhan Chen, "Network-Adaptive Video Coding and Transmission" in IS&T/SPIE Conference on Visual Communications and Image Processing 99, pp. 854-861

[4] Fabrice Le Leannec and Christine Guillemot, "Error resilient Video Transmission over the Internet" in IS&T/SPIE Conference on Visual Communications and Image Processing 99, pp271-280

[5] Clarke, R. H., "A statistical theory of mobile radio reception", Bell System Technical Journal, Vol. 47, pp. 957-1000, 1968

[6] William Turin, Robert van Nobelen, "Hidden Markov Modeling of Flat Fading Channels", IEEE Journal on Selected Areas in Communications, Vol. 16, NO. 9, December 1998.

[7] Gilbert, E. N. "Capacity of a Burst-Noise Channel", Bell Systems Technical Journal. 39: 1253-1266, 1960

[8] Elliot, E.O. "Estimation of error rates for codes on burst-error channels", Bell System Technical Journal, pp. 1977, September 1963

[9] Klaus Stuhlmuller, Niko Farber, Michael Link, and Bernd Girod, "Analysis of Video Transmission over Lossy Channels", IEEE Journal on Selected Areas in Communications, Vol. 18, NO. 6, pp. 1012, June 2000

[10] Turin, W., "Digital Transmission Systems: Performance Analysis and Modeling", New York: McGraw-Hill, 1998

[11] Kanal, L. N. and Sastry A. R. K., "Modeling for Channels with Memory and their applications to error control", Proc. of the IEEE, vol. 66, no. 7, pp. 724-744, July 1978

[12] Ger Koole, Zhen Liu, Rhonda Righter, "Optimal Transmission Policies for Noisy Channel", Technical Report WS-515, Faculteit der EXacte Wetenschappen, Vrije Universiteit Amsterdam, 1999.

[13] Boyce, Jill M. and Gaglianello, Robert D., "Packet Loss Effects on MPEG Video Sent Over the Public Internet", ACM Multimedia 98

[14] Claypool, M. and Tanner J., "The Effect of Jitter on the Perceptual Quality of Video", Proceedings of ACM Multimedia, Oct. 30 – Nov. 5, 1999

[15] Winpcap, http://netgroup-serv.polito.it/windump/, documentations: http://netgroup-serv.polito.it/windump/docs/Default.htm

[16] ASFRecorder, http://www.geocities.com/asfrecorder/

[17] Tektronix publication, "A Guide to MPEG Fundamentals and Protocol Analysis", Tektronix 1998

[18] Wang, Hong-shen, Chang, Pao-chi, "On Verifying the First-Order Markovian Assumption for a Rayleigh Fading Channel Model", IEEE Trans. Veh. Tech., Vol 45, No. 2, May 1996