VIDEO STREAMING MODELIG OVER OPTICAL NETWORKS

Author: Marcos Esgueva Martínez

Head of department: Algumantas Kajackas

Supervisor: Dr. Sarunas Paulikas

Supervisor UC3M: Carmen Vázquez

Academic Coordinator UC3M: Carmen Vázquez

CONTENTS

Prologue

Goal

The goal of this project is to implement a Matlab model to simulate video degradation in streaming over optical network influenced by networks parameters (delay, jitter, error probability, bandwidth,etc). To achieve it, we needed to make several tasks summarized in the following paragraph.

Motivations

This model is useful to observe how network parameters affects in video quality and how much. For example, estimating error probability, delay and jitter in a specific network, with this model we can forecast if that network will satisfy requirements to perform video streaming.
We can also observe which parameter is more important to take into account in video streaming or what threshold in each network parameter we can set to have an acceptable video quality.

Tasks

First of all, we needed documentation about how video streaming is implemented (protocols, MPEG compression), and how network parameters affect in packets losses. After that, we had to investigate how these losses affect different frames in MPEG compression.
Later, we implemented a video model following these rules.
Last, we executed Matlab code with different video files and values of parameters and we extracted conclusions.

Problems

Some problems we have found doing this project are that there are few similar projects already done and it is difficult to find useful examples in Matlab about this. Another thing is that some values, for example JPEG compression are rough, not exact.

1. Introduction about real-time transmissions

Streaming is a term that refers to watch or listen a file directly on a web page without having to download it to your computer before. It is something like "click and get." It could be said that describes a strategy on demand for distribution of multimedia content through the internet.

Such technology allows to store in a buffer what the user is listening or watching. You can listen to music or watch video without downloading it previously.

1.1 Requirements for Real-Time Multimedia Traffic

According to reference [20] these requirements are:

In order to ensure playback timing and jitter removal timestamps are required. In order to ensure the presence and order of data a sequence number is required.

Most of the real-time multimedia applications are video conferencing where several clients receive data. Therefore the multicast mode is preferred.
In order to deal with congestion a mechanism for sender notification and change of encoding parameters must be provided.

In order to display streams generated by different standards the choice of encoding must be provided.

In order to display audio and video streams (and user data like titles) mixers are required.

In order to use high bit rate streams over a low-bandwidth network translators are required (multimedia payload encoders and decoders)

1.2 Real-time using TCP

As shows figure 1, real-time traffic uses UDP transport protocol for transmission. Some reasons for not using TCP, according reference [9] are:

- TCP forces the receiver application to wait for retransmission in case of packet loss, which causes large delays which are problematic in real-time transmissions.
- TCP cannot support multicast.
- TCP congestion control mechanisms decreases the congestion window.
- TCP headers are larger than a UDP header (40 bytes for TCP compared to 8 bytes for UDP).
- TCP doesn't contain the necessary timestamp and encoding information needed by the receiving application.
- TCP doesn't allow packet loss.

2. Protocols

The main protocols related with video streaming are:

- RTP (Real-Time Protocol)
- RTCP (Real-Time Control Protocol)
- RSVP (Real-Time Reservation Protocol)
- RSTP (Real-Time Streaming Protocol)
- SIP (Session Initiation Protocol)

2.1 RTP (Real-Time Protocol)

According to reference [2] :

It is a IP-based protocol which as its name suggests provides support for the transport of real-time data such as video and audio streams.

RTP is primarily designed for multicast of real-time data, but it can be also used in unicast. It can be used for one-way transport such as video-on-demand as well as interactive services such as Internet telephony.

RTP is designed to work in conjunction with the auxiliary control protocol RTCP to get feedback on quality of data transmission and information about participants in the on-going session.

2.1.1 Advantages for streaming

RTP doesn't ensure real-time delivery itself, but according to reference [14], it provides means for:

- Jitter elimination/reduction (with application's playout buffers).
- Time reconstruction.
- Synchronization of several audio and/or video streams that belong to the same multimedia session.
- Multiplexing of different audio/video streams.
- Content identification.
- Translation of video streams from one encoding type to another

- With the help of RTCP, RTP also provides reliability and flow/congestion control which is implemented within the multimedia application.

2.1.2 How RTP works

According to reference [14]:

Packets sent on the Internet have unpredictable delay and jitter. But multimedia applications require appropriate timing in data transmission and playing back. RTP provides time stamping, sequence numbering, and other mechanisms to take care of the timing issues.

Time stamping is the most important information for real-time applications. The sender sets the timestamp according to the instant the first octet in the packet was generated. After receiving the data packets, the receiver uses the timestamp to reconstruct the original timing in order to play out the data correctly. Timestamp is also used to synchronize different streams with timing properties, such as audio and video data in MPEG. However, RTP itself is not responsible for the synchronization; this has to be done in the application level.

UDP does not deliver packets in timely order, so sequence numbers are used to place the incoming data packets in the correctly. They are also used for packet loss detection. Notice that in some video format, when a video frame is split into several RTP packets, all of them can have the same timestamp, so just timestamp is not enough to put the packets in order.

The payload type identifier specifies the payload format as well as the encoding/compression schemes.

Thanks to this identifier, the receiving application knows how to interpret and play the payload data. Default payload types are defined in RFC 1890. More payload types can be added by providing a profile and payload format specification. At any given time of transmission, an RTP sender can only send one type of payload, although the payload type may change during transmission, for example, to adjust to network congestion.

Another function is source identification. It allows the receiving application to know where the data is coming from. For example, in an audio conference, from the source identifier a user could tell who is talking.

RTP works with RSVP (Resource Reservation Protocol) for the Reservation of bandwidth and for providing the type of QoS guaranteed. RTP also provides transportation for unicast and multicast addresses. For this reason, IGMP is also involved.

Another tool that RTP uses to achieve real-time transmission is RTCP (Real Time Control Protocol), which can diagnose problems that appear in the network and provide feedback about the quality of transmission and congestion.

To set up an RTP session, the application defines a particular pair of destination transport addresses (one network address plus a pair of ports for RTP and RTCP).

In a multimedia session, each medium is carried in a separate RTP session, with its own RTCP packets reporting the reception quality for that session. For example, audio and video would travel on separate RTP sessions, enabling a receiver to select whether or not to receive a particular medium.

2.1.3 RTP Header

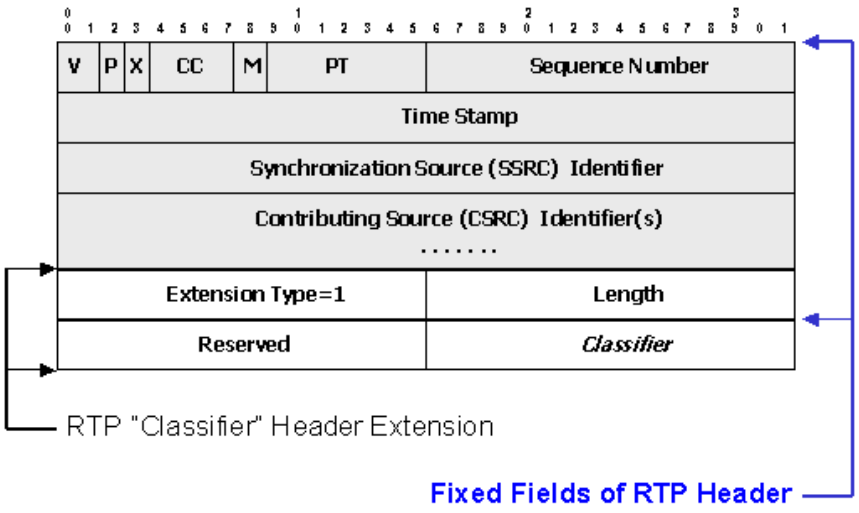

Figure 1. RTP Header

The first twelve octets in a RTP header (see Figure 2) are present in every RTP packet, while the list of CSRC (contributing source) identifiers is present only when inserted by a mixer.

The fields have the following meaning according to reference [2] :

Version (V): 2 bits. Version of RTP. Current version is 2 .

Padding (P): 1 bit. Some encoding algorithms require fixed block sizes, therefore padding is needed.
If set the packet contains one or more additional padding octets at the end which are not part of the payload. The last octet of the padding contains a count of how many padding octets should be ignored.

Extension (X): 1 bit. If set, the fixed header is followed by exactly one header extension required by some programmes.

CSRC count (CC): 4 bits. The number of CSRC identifiers that follow the fixed header. This number is more than one if the payload of the RTP packet contains data from several sources.

Marker (M): 1 bit. Defined by a profile, the marker is intended to allow significant events such as frame boundaries to be marked in the packet stream (e.g. starting of a talk spurt in audio, the beginning of frame in a video).

Payload type (PT): 7 bits. Identifies the format of the RTP payload and determines its interpretation by the application.
The size of the payload is a compromise between overhead and packetization delay. In addition, it is desirable that the multimedia packets are as small as possible so that the packet loss doesn't decrease the quality of reception (for example 20 ms of uncompressed voice has 160 samples (bytes). Loss of 20 ms of voice is tolerable. Loss of larger packets can be annoying.

Sequence number: 16 bits. Increments by one for each RTP data packet sent, may be used by the receiver to detect packet loss and to restore packet sequence. The initial value is randomly set.

Timestamp: 32 bits. The sampling instant of the first octet in the RTP data packet. It may be used for synchronization and jitter calculations. The initial value is randomly set.

SSRC: 32 bits. Randomly chosen number to distinguish synchronization sources within the same RTP session. It indicates where the data was combined, or the source of the data if there is only
one source.

CSRC list: 0 to 15 items, 32 bits each. Contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field.

2.2 RTCP (Real-Time Control Protocol)

RTP is a protocol that provides basic transport layer for real time applications but does not provide any mechanism for error and flow control, congestion control, quality feedback and synchronization. For that purpose the RTCP is added as a companion to RTP to provide end-to-end monitoring and data delivery, and QoS.

RTCP is based on the regular transmission of control packets to all participants of a session, using the same mechanism of transport as RTP packets.
RTCP and RTP use different ports; generally consecutives ports where even one is assigned to the RTP flow and the odd one is for RTCP flow.

2.2.1 Advantages for streaming

• It is a tool used by RTP to achieve real-time transmission, which provides feedback about the quality of distribution and congestion.

• RTCP synchronizes the audio and video, knows the number of users in a conference and this way, it estimates the rate which the packages should be sent. It also allows controlled access to more participants.

• Identification: recipients use canonical name to have a list of participants in a session. It is also used to put together different streams of a participant, for example to synchronize audio and video.

• This protocol helps RTP to make easy end-to-end communication and monitor the quality of service and information about the participants in the meeting.

• RTCP messages are stackable – to amortize header overhead multiple RTCP messages can be combined and sent in a compound RTCP message.

• RTCP involves several types of messages, including:

- *Send Report* for transmission and reception of statistics in random time from active users.
- *Receiver Report* for receiving statistics from non-active issuers.
- *Source Description Identifier* for the transport level identifier called CNAME (Canonical Name).
- *Bye* to indicate the end of participation in the connection.
- *Application* for specific applications.

2.2.2 RTCP Header and packets

As we can see in figure 3, and according to reference [19] ,fields in a RTCP header are:

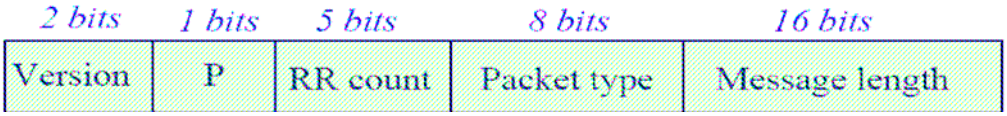

Figure 2. Common RTCP Header

- Version: the same as RTP, version 2.
- P : Padding bit, the same as RTP.
- RR count: number of reception blocks contained in this packet
- Packet type: type of RTCP packet (NACK, Sender Report, Receiver Report, etc)

Sender Report (SR) Packet

RTCP receivers provide reception quality feedback using a SR or a RR packets (if receiver is also a sender)

Receiver Report (RR)Packet

Reception statistics from receivers that are not senders.
Same format as SR packet, but without the sender's information block.

Source Description Packet

Used by source to provide more information about itself. Useful for user interfaces.

Bye Packet

Used to confirm to receivers that a prolonged silence is due to departure of one or more sources rather than network failure.

2.3 RSVP (Resource Reservation Protocol)

According to reference [2]:

RSVP is a network-control protocol that enables Internet applications to obtain differing qualities of service (QoS) for their data flows. Such a capability recognizes that different applications have different network performance requirements.

Thus, RSVP was intended to provide IP networks with the capability to support the divergent performance requirements of differing application types.

It is important to note that RSVP is not a routing protocol. RSVP works in conjunction with routing protocols and installs the equivalent of dynamic access lists along the routes that routing protocols calculate. Thus, implementing RSVP in an existing network does not require migration to a new routing protocol.

2.3.1 Advantages for streaming

According to reference [3], they are:

• *RSVP flows are simplex.*

RSVP distinguishes senders and receivers. Although in many cases, a host can act both as a sender and as a receiver, one RSVP Reservation only reserves resources for data streams in one direction.

• *Supports both multicast and unicast, and adapts to changing memberships and routes.*

RSVP is designed for both multicast and unicast. Since the Reservations are initiated by the receivers and the Reservation states are soft, RSVP can easily handle changing memberships and routes. A host can send IGMP (Internet Group Management Protocol) messages to join a multicast group. Reservation merging enables RSVP to scale to large multicast groups without causing heavy overhead for the sender.

• *RSVP is receiver-oriented and handles heterogeneous receivers.*

In heterogeneous multicast groups, receivers have different capacities and levels of QoS. The receiver oriented RSVP Reservation requests facilitate the handling of heterogeneous multicast groups. Receivers are responsible for choosing its own level of QoS, initiating the Reservation and keeping it active as long as it wants. The senders divide traffic in several flows, each is a separate RSVP flow with different level of QoS. Each RSVP flow is homogeneous and receivers can choose to join one or more flows. This approach makes it possible for heterogeneous receivers to request different QoS tailored to their particular capacities and requirements.

• *RSVP has good compatibility.*

Efforts have been made to run RSVP over both IPv4 and IPv6. It provides opaque transport of traffic control and policy control messages in order to be more adaptive to new technologies. It also provides transparent operation through non-supporting regions.

2.3.2 How RSVP works

According to reference [3]:

RSVP is used to set up Reservations for network resources. When an application in a host (the data stream receiver) requests a specific quality of service (QoS) for its data stream, it uses RSVP to deliver its request to routers along the data stream paths. RSVP is responsible for the negotiation of connection parameters with these routers. If the Reservation is setup, RSVP is also responsible for maintaining router and host states to provide the requested service.

Each node capable of resource Reservation has several local procedures for Reservation setup and enforcement (see Figure 1). Policy control determines whether the user has administrative permission to make the Reservation. In the future, authentication, access control and accounting for Reservation will also be implemented by policy control. Admission control keeps track of the system resources and determines whether the node has sufficient resources to supply the requested QoS.

The RSVP daemon checks with both procedures. If either check fails, the RSVP program returns an error notification to the application that originated the request. If both checks succeed, the RSVP daemon sets parameters in the packet classifier and packet scheduler to obtain the requested QoS. The packet
classifier determines the QoS class for each packet and the packet scheduler orders packet transmission to achieve the promised QoS for each stream.

RSVP daemon also communicates with the routing process to determine the path to send its Reservation requests and to handle changing memberships and routes.

This Reservation procedure is repeated at routers along the reverse data stream path until the Reservation merges with another reservation for the same source stream.

Reservations are implemented through two types of RSVP messages: PATH and RESV. The PATH messages are sent periodically from the sender to the multicast address. A PATH message contains flow spec to describe sender template (data format, source address, source port) and traffic characteristics.

This information is used by receivers to find the reverse path to the sender and to determine what resources should be reserved. Receivers must join the multicast group in order to receive PATH messages.

RESV messages are generated by the receivers and contains reservation parameters including flow spec and filter spec. The filter spec defines what packets in the flow should be used by the packet classifier.

The flow spec is used in packet scheduler and its content depends on the service. RESV messages follow the exact reverse path of PATH messages, setting up Reservations for one or more senders at every node.

The Reservation states RSVP builds at the routers are soft states. The RSVP daemon needs to send refresh messages periodically to maintain the Reservation states. The absence of refresh message within a certain time will destroy the Reservation state. By using soft states, RSVP can easily handle changing memberships and routes.

The Reservation requests are initiated by the receivers. They do not need to travel all the way to the source of the sender. In stead, it travels upstream until it meets another Reservation request for the same source stream, then merges with that Reservation.
This Reservation merging leads to the primary advantage of RSVP: scalability---a large number of users can be added to a multicast group without increasing the data traffic significantly. So RSVP can scale to large multicast groups and the average protocol overhead decreases as the number of participants increases.

The Reservation process does not actually transmit the data and provide the requested quality of service. But through Reservation, RSVP guarantees the network resources are available when the transmission actually takes place.

Although RSVP sits on top of IP in the protocol stack, it is not a routing protocol, but rather an internet control protocol. Actually, RSVP relies on the underlying routing protocols to find where it should deliver the Reservation requests. RSVP is also intended to cooperate with unicast and multicast routing protocols. When the RSVP-managed flow changes its path, the routing module will notify the RSVP module of the route changes. Therefore, RSVP can quickly adjust the resource Reservation to new routes.

The delivery of Reservation parameters is different from the determination of these parameters. How to set the connection parameters to achieve the requested QoS is the task of QoS control devices, the role of RSVP is just a general facility to distribute these parameters. Since different applications may have different QoS control devices, RSVP is designed to treat these QoS parameters as opaque data to be delivered to and interpreted by the control modules at the routers. This logical separation of QoS control devices and distribution facility simplifies the design of RSVP and makes it more adaptive to new network technologies and applications.

2.3.3 RSVP Header

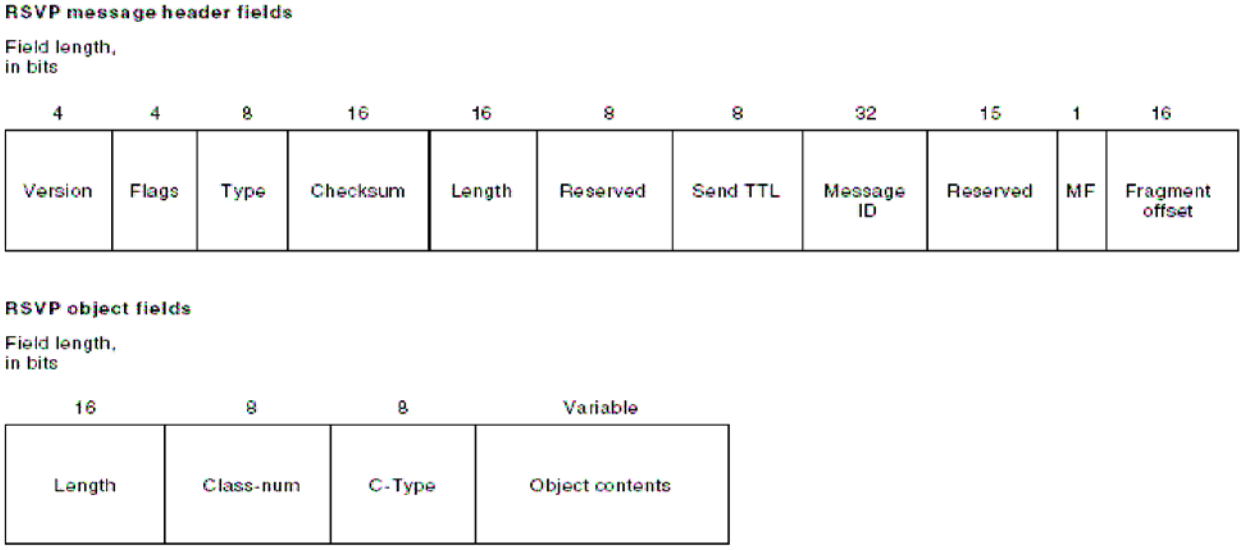

Figure 3. RSVP Header

RSVP message header, as Figure 9 shows, fields are the following ones (reference [21]):

• **Version**— A 4-bit field indicating the protocol version number (currently version 1).

• **Flags**— A 4-bit field with no flags currently defined.

• **Type**— An 8-bit field with six possible (integer) values, indicating the type of message.

• **Checksum**—A 16-bit field representing a standard TCP/UDP checksum over the contents of the RSVP message, with the checksum field replaced by 0.

• **Length**—A 16-bit field representing the length of this RSVP packet in bytes, including the common header and the variable-length objects that follow. If the More Fragment (MF) flag is set or the Fragment Offset field is nonzero, this is the length of the current fragment of a larger message.

• **Send TTL**—An 8-bit field indicating the IP time-to-live (TTL) value with which the message was sent.

• **Message ID**—A 32-bit field providing a label shared by all fragments of one message from a given next/previous RSVP hop.

• **More fragments (MF) flag**—Low-order bit of a 1-byte word with the other 7 high-order bits specified as reserved. MF is set on for all but the last fragment of a message.

• **Fragment offset**—A 24-bit field representing the byte offset of the fragment in the message

RSVP object fields the following ones:

• **Length**—Is a 16-bit field containing the total object length in bytes (must always be a multiple of 4 and must be at least 4).

• **Class-num**—Identifies the object class. Each object class has a name.
The high-order bit of the Class-Num field determines what action a node should take if it does not recognize the Class-Num of an object.

• **C-type**—Object type, unique within Class-Num. The maximum object content length is 65528 bytes. The Class-Num and C-Type fields (together with the flag bit) can be used together as a 16-bit number to define a unique type for each object.

• **Object contents**—The Length, Class-Num, and C-Type fields specify the form of the object content.

2.4 RTSP (Real-Time Streaming Protocol)

Realtime Streaming Protocol is an application level protocol that aims to provide a robust protocol for streaming multimedia in applications over unicast and multicast, and to support interoperability between clients and servers from different vendors.

RTSP is designed to work with lower-level protocols like RTP, RSVP to provide a complete streaming service over internet. It provides means for choosing delivery channels (such as UDP, multicast UDP and TCP), and delivery mechanisms based upon RTP.

It provides "VCR-style" remote control functionality for audio and video streams, like pause, fast forward, reverse, and absolute positioning.Sources of data include both live data feeds and stored clips.
RTSP is considered more of a framework than a protocol.

2.4.1 Advantages for streaming

According to reference [12 and 13] these are RSTP main advantages:

• Extensible: You can add new methods and parameters
• Independent Transportation: Information flows will be transported using another protocol.
• Multiserver: Each stream within a presentation may reside in a different server.
• Load balancing using redirection at connect, during stream.
• Control about quality of service, efficiency of delivery, rights management, and measurement.
• Provides for control and delivery of realtime media and associated events between a media server and large numbers of media clients.
• Device control , for example camera pan, zoom, etc.
• RTSP is implemented on multiple operating system platforms, it allows interoperability between clients and servers from different manufacturers.

2.4.2 How RTSP Works

According to reference [12]:

RTSP takes advantage of streaming which breaks data into packets sized according to the bandwidth available between client and server. When the client has received enough packets, the user's software can be playing one packet, decompressing another, and downloading the third. This enables the user to listen or view the realtime file almost immediately, and without downloading the entire media file. This applies to live datafeeds as well as stored clips.

To send a control request, the client constructs a line consisting of the method, the request URL, and the protocol version number. Then, the client includes a general header, a request header and possibly an entity header, as for the http protocol. This is sent to the server, which executes the request if possible. It then returns a response containing a status line and general response and entity headers. The status line contains the protocol version, the numeric status code, and a textual description. The media streams are left unspecified by RTSP. These could be RTP streams, or any other form of media transmission. RTSP only specifies the control and its up to the client and server software to maintain the mapping between the control channel and the media streams.

A key concept in RTSP is the notion of a session. RTSP works by first requesting a presentation to be started by a server, receiving in return a session identifier which it then uses in all subsequent controls. Eventually, the client can request the teardown of session, which releases the associated resources. The session identifier represents the shared state between the client and server. If the state is lost, for example through one of the machines being rebooted, then the protocol relies on the transport of the media stopping automatically, eg. through not receiving RTCP messages if using RTP, or the implementation using the GET_PARAMETER method below as a keepalive.

The control requests and responses may be sent over either TCP or UDP. Since the order of the requests matters, the requests are sequenced, so if any requests are lost, they must be retransmitted. Using UDP thus requires the construction of retransmission mechanisms, so there are very few occasions when the applicationcan get away with using UDP.The most obvious additions to the request header fields are a Cseq field to contain the sequence numbers of requests generated by the client, and a Session field to both the request and response headers to identify the session. Session identifiers are generated in response to a SETUP request, and must be used in all stateful methods. The Transport field allows the client and server to negotiate and set

parameters for the sending of the media stream. In particular, it allows the client and server to set ports and multicast addresses for the RTP streams.

RTSP Request :

A request message from a client to a server or vice versa includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

RTSP Response :

After receiving and interpreting a request message, the recipient responds with an RTSP response message.
The first line of a Response message is the StatusLine, consisting of the protocol version followed by a numeric status code; and the textual phrase associated with the status code, with each element separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

2.5 SIP (Session Initiation Protocol)

According to reference [23]:

SIP is an ASCII protocol that facilitates the formation, modification and execution of communication sessions between individual or multiple participants. The participants can either be a person (videoconferencing clients) or an automation component (voicemail server) or a device that can interact in a similar manner. Various interaction types can be incorporated in these communications, including peer-to-peer or multipoint communication. Users have an address that simulates an email address for identification and location purposes.

SIP is used to set up, modify and terminate an RTP-based multimedia session (like IP telephone conversation or videoconferencing).

An alternative protocol to SIP is H.323, which is more complex and more difficult to implement/configure. SIP is a simple protocol well suited for Internet.

2.5.1 Advantages for streaming

User Location

SIP determines user locations by a registration process. When a soft-phone is activated on a laptop, it sends out a registration to the SIP server announcing availability to the communications network. Voice over-IP (VoIP) phones, cellular phones, or even complete teleconferencing systems can be registered as well. Depending on the registration point chosen, there may be several different locations registered simultaneously.

User Availability

User availability is simply a method of determining whether or not a user would be willing to answer a request to communicate. If you “call” and no one answers, SIP determines that a user is not available. A user can have several locations registered, but might only accept incoming communications on one device. If that is not answered, it transfers to another device, or transfers the call to another application, such as voicemail.

User Capabilities

With all the various different methods and standards of multimedia communications, something is needed to check for compatibility between the communications and the users’ capabilities. For example, if a user has an IP phone on their desk, a white-board conference via that device would not work. This function also determines which encryption/decryption methods a user can support.

Session Setup

SIP establishes the session parameter for both ends of the communications - more specifically, where one person calls and the other answers. SIP provides the means to setup and/or establish communications.

Session Management

This function provides the greatest amount of user awe. Provided a device is capable, a user could transfer from one device to another - such as from an IP-based phone to a laptop -without causing a noticeable impact. A user's overall capabilities would change - such as being able to start new applications such as white-board sharing - perhaps affecting the voice quality temporarily as SIP re-evaluates and modifies the communications streams to return the voice quality. With SIP session management, a user can also change a session by making it a conference call, changing a telephone call to a video conference, or opening an in-house developed application. And finally, SIP terminates the communications.
Although SIP has five functions, it is currently easier to think of SIP as the setup, management,and tear down of IP-based communications. The user location and capabilities functions could easily become absorbed into the session setup function. SIP maintains the distinct five functions to remain open and provide for an unknown future.

2.5.2 How SIP works

According to reference [23]:

We can see how this works by taking a look at a simple example of user 'A' trying to call user 'B'.
UAC **(**User agent client**)** of user 'A' sends a SIP invite request to its local UAS **(**User agent server**).** If this is a proxy server, then it forwards the request to the next UAS until it hits its destination. If it is a re-direct server then it returns the direct address of B. In SIP, addresses are represented as URL's and follow a similar layout to email addresses.

The invite includes a full description of the session including all of the media streams that A wants to use. B replies to the invitation, but includes a description of any modifications that he wants to make. This is for compatibility reasons, as B might support all of the features that A asked for.

After this negotiation is completed, the session is created and A and B can communicate. At the end of the call, either side can send a disconnect, terminating the session.

All of this process is automatic. For example, when A calls B, if B picks up his VoIP videophone, the phone automatically handles the media negotiation process. When B puts the phone down, the disconnect is automatically sent.

It is also the job of a SIP device to register its current location with a UAS. This ensures that a user can be found even when mobile.

3. MPEG Compression

MPEG-2 is a video coding standard created by the Moving Picture Experts Group (MPEG) and finalized in 1994.

The MPEG-2 bit stream is basically just a series of coded frames one after the other. There are headers and time stamps to help decoders align audio and scrub through the bit stream.

Color model

Coding a frame in MPEG-2 format always begins by representing the original color frame in YCbCr format. The Y component represents luminance, while Cb and Cr represent chrominance differences. The three components of this color model are mostly uncorrelated so this transformation is a useful first step in reducing redundant information in the frame.

Chroma subsampling

According to reference [11]:

Another way to reduce the amount of information to be encoded is by taking advantage of human vision characteristics. Human eyes are much more sensitive to luminance than chrominance, so it is common to subsample both chrominance channels. The most commonly used subsampling format is denoted by 4:2:0, which means that chrominance sampling is decimated by 2 in the horizontal and vertical direction. Both chrominance channels are reduced to one quarter the original data rate in this way and the net effect is for total frame data rate to be cut in half with hardly any perceptual effect on image quality.

I, P, and B frames

The next encoding step can vary from frame to frame. There are actually three possible types of frames, called I, P, and B frames. Each will be discussed separately below.

I frame coding

According to reference [15]:

An I frame is *intra* or *spatially* coded so that all the information necessary to reconstruct it is encoded within that frame's segment of the MPEG-2 bit stream. It is a self contained image compressed in a manner similar to a JPEG image.
Each block of the frame is processed independently with an 8x8 discrete cosine transform (DCT). This transform generates a representation of each 8x8 block in the frequency domain instead of the spatial domain. Pixels in each block of a natural video are likely to be correlated, the resulting DCT coefficients typically consist of a few large values and many small values. The relative sizes of these coefficients represent how important each one is for reconstructing the block accurately.
The coefficients are then quantized using:

QDCT = round((8 x DCT) / (scale x Qi)

where *DCT* and *QDCT* represent the coefficient matrices before and after quantization, *scale* is used to determine quality, and *Qi* is a perceptually weighted quantization table. The default value of *Qi*,weights low frequency content higher that high frequency content, mimicking the response characteristics of a human eye. The quality scalar, *scale*, can be any value from 1 to 31. If desired, MPEG-2 allows the value of *scale* to be mapped non-linearly to higher compression levels, with a maximum of 112.

This quantization process is lossy because the true floating point values of the DCT coefficients are not preserved. Instead, the quantized coefficients are just an approximation of the true coefficients and the quality of the approximation determines the quality of the frame reconstructed from this bit stream.
Next the 8x8 block of quantized coefficients is arranged into a vector by indexing the matrix in zigzag order.

This ordering is approximately low frequency to high frequency for a very important reason: small blocks (8x8) of natural video are likely to contain mostly low frequency content. This ordering tends to group non-zero terms at the front of the vector and zero terms at the end. This type of distribution is beneficial to the coding that follows.

The final steps for coding an I frame are lossless with respect to the quantized DCT coefficients. For each block, the DC coefficient is differentially coded with the last block's DC term. The AC coefficients are run-length encoded (RLE) and then Huffman coded. The result is a compact bit stream from which the quantized DCT coefficients can be reconstructed perfectly.

P frame coding

According to reference [15]:

A P frame is *inter* or *temporally* coded. That means it uses correlation between the current frame and a past frame to achieve compression. The MPEG-2 standard dictates that the past frame must be an I or P frame, but not a B frame.

Temporal coding is achieved using motion vectors. The basic idea is to match each macroblock in the current frame with a 16x16 pixel area in the past reference frame as closely as possible. *Closeness* here can be computed in many ways, but a simple and common measure is sum of absolute differences (SAD). The offset from the current macroblock position to the closest matching 16x16 pixel area in the reference frame is recorded as two values: horizontal and vertical motion vectors.

The search to find the best motion vectors is performed in the luminance channel only. Whatever motion vector is found then applies to all three channels of that macroblock.There are a few common algorithms for finding motion vectors.

These motion vectors are a simple way to convey a lot of information, but are not always a perfect match. To give better quality reconstruction, error between the actual macroblock and the predicted macroblock is then encoded. This coding of the residual is almost exactly the same as I frame coding. In fact the only difference is that the quantization equation becomes

QDCT = floor((8 x DCT) / (scale x Qp))

where *floor* is used in place of *round*, and *Qp* is a different weighted quantization table. The default for *Qp* is

During reconstruction, the residual is decoded and added to the motion vector predicted macroblock.

B frame coding

According to reference [15]:

A "B" frame is simply a more general version of a P frame. Motion vectors can refer not only to a past frame, but to a future frame, or both a past and future frame. Using future frames is exactly like a P frame except for referencing the future. Using past and future frames together works by averaging the predicted past macroblock with the predicted future macroblock. The residual is coded like a P frame in either case.

Encoded frame size

Encoded frame size greatly depends on the quantization scale value, but the relative frame sizes demonstrate how well temporal coding can work. In general, a P frame will be 30% the size of an I frame, and a B frame will be only 15% of the size of an I frame.

Frame sequencing

According to reference [11]:

The ordering of I, P, and B frames is fairly flexible in MPEG-2. The first frame must always be an I frame because there is no other data for a P or B frame to reference. After that, I, P, and B frames can be mixed in any order.
Remember that the P frames reference the last I or P frame and the B frames reference the closest past *and* future I or P frames. Due to those requirements, the frames must be coded and transmitted out of display order.
The order would be: *I P B B P B B P B B I B B*

In network transmission is quite used I frames repetitions among 18 frames, it is ***I B B P B B P B B P B B P B B P B B***

4. Influence of network parameters on streaming video.

Packet delay variation and packet loss have been shown to be the key characteristics in determining whether a network can transport good quality video. These characteristics can be used to assess how well a network can transport video and to troubleshoot problems in networks whose performance has deteriorated due to reconfiguration or changing traffic loads and, finally, as an indication of margin, or headroom, in the network.

Bandwidth

Insufficient bandwidth will result in packet loss, which leads to lousy video.
Thanks to RSVP, we are able to do bandwidth Reservations for any application that requires predictable data delivery over the Internet.

RSVP guarantees good quality and acceptability in a heavily loaded network.

Packet loss

The packet loss is the main cause of degradation of quality. Due to the way in which the video is encoded, the loss of a package in one frame can affect the following frames. For example a frame that is Type I, its degradation resulted in degradation of the following ones, until the reception of a new frame I.

In the work of Liang, Apostolopoulos and Griod it is introduced an analysis of the impact the packet loss in the quality of compressed video. To do it, they consider that each individual package represents a frame and if a frame gets lost is replaced by the previous received. Coding is such that I will be sent a frame I every N frames, the rest ones are type P.

To measure the distortion total the use the sum of MSE (Mean squared error) of each frame. The conclusion reached is that not only it is important the mean of lost packets, also the distribution of such losses. For example, total distortion due to two consecutive losses is higher than if the losses were independent (separated by more N frames).

Delay and Jitter

If packets are delayed by the network such as in Figure 15, some packets arrive in bursts with interpacket delays shorter than when they were transmitted such as those in the “1 Second” group while others are delayed, as in the “3 Seconds” group, such that they arrive with greater delay between packets than when they were transmitted from the source. This is defined as packet jitter and is the time difference between when a packet actually arrives and the expected arrival time. A receiver (decoder) displaying the video at its nominal rate must accommodate the varying input stream arrival times by buffering the data arriving early and assuring that there is enough already stored data when packets arrive late by prefetching data before beginning to display the video. Thus there is a balance made in all systems between having sufficient buffer memory to smooth all network delays and encountering excessive delay due to a large buffer memory.

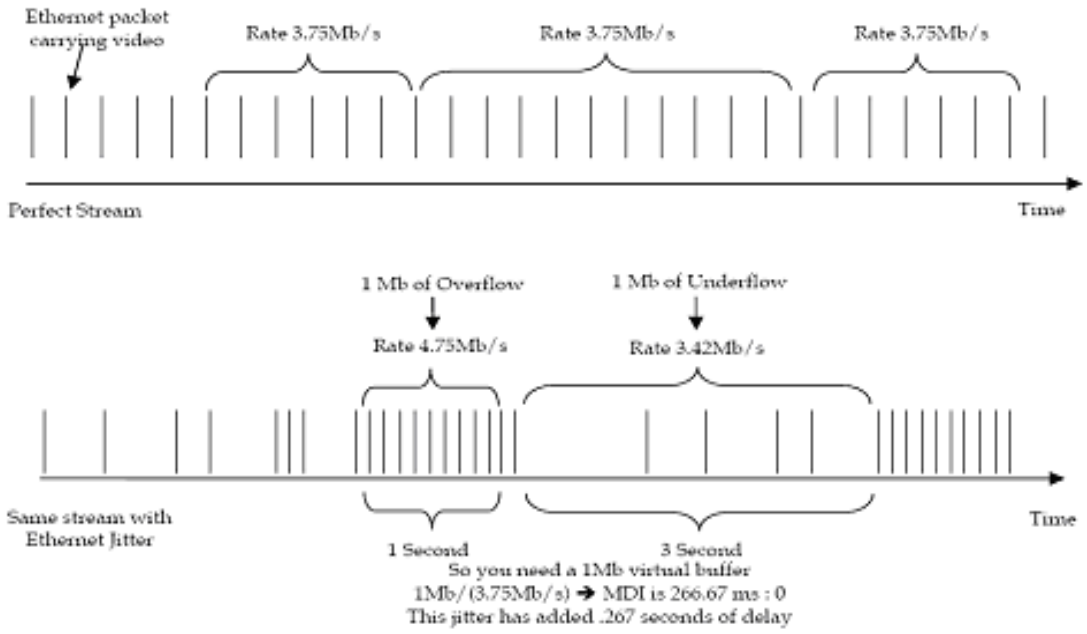

Figure 4. Delaying packets in a network

For a given size de-jitter buffer then, there is a maximum amount of network jitter that can be accommodated without buffer overflow or underflow - either of which would result in a poor video display.

There are three different scenarios in video streaming quality related to delay and jitter: no loss and no jitter, channel with jitter and channel with losses.

In the case of no losses and packet fixed delay, decoding is performed in time and form, resulting in a smooth and continuous playback of video.

In the case of jitter, the variation in the time between arrivals makes the decoding not in the right time and, depending on how big is this variation, one packet could be considered lost. The user will see the image of the last frame decodified frozen until the arrival of the next table. This frame will be reproduced less time than required in order to maintain the time sequence with the next frame.

In the case of losses, apart from the effect of freezing the image is added the problem of the necessity of the previous frames to decode the next frames .Intuitively you can see then the effects of the loss and jitter on the perception of the user are similar. This similarity led to do a research to compare the combined effects of these factors (experiment by Claypool and Tanner in 1999).
The experiment consisted in generating distorted video sequences with different levels of loss and jitter and then sort then by quality. They analysed video sequences with different temporal variation in order to determine the combined influence of all parameters on perceived quality.

For three categories of temporal variation (low, medium and high) they obtained sequences of the type low jitter high jitter, low and high rate loss. A subjective MOS test was carried out by ACR for 25 different sequences, including an indication from each person about the number of distortion events received. Finally they compared the obtained results in the case of jitter versus the case of losses.

There is a high correlation between MOS values obtained and the average number of distortion events detected, which suggests the use of a counter of those events to estimate perceived quality (eg, identifying how many times certain thresholds are exceeded in parameters).

5. Report about Matlab model

Purpose of the work

The aim of this project is implementing a Matlab model to simulate video degradation in streaming over optical network influenced by networks parameters (delay, jitter, error probability, bandwidth,etc).

In our Matlab model we consider only the three first ones because how we mentioned bandwidth for example does not affect video quality thanks to RSVP protocol. Therefore, you can only modify that parameters through variables “error_prob“ , “average_jitter“ and “average_delay”.

Software requirements

In order to run this Matlab code, you need to have Matlab 2008 Release or higher version in order to execute “mmreader“ Matlab function.

Input video must be MPEG2 type, so first of all it is necessary convert AVI files in MPEG with an external code or software. In our case we have used a freeware program called “Easy AVI VCD DVD MPEG Converter“.

It is also needed a suitable MPEG2 codec suitable for Matlab. We haved used Elecard MPEG-2 for Microsoft Windows XP.

Code structure

This Matlab model is divided in these sections that will be explained later:

- Calculus of the size of each frame and the number of packets to send each frame.
- Extracting each frame from the video file.
- RTP header generation
- UDP/IP packing
- Calculus of error/lost packets because of error probability, delay and jitter.

- Degradation of frames.

How video streaming model works

First of all ,to start using this model, you have to call *"video_streaming_model"* function with four parameters: video name ending in .mpeg, a string that represents error probability using *"eval"* Matlab function to evaluate that expression and extract a number, average delay in milliseconds and average jitter expressed in percentage.

One example to call this function it could be: video_streaming_model('foreman.mpeg' , eval('caller', ' 10^(-9)') , 31.4 , 5)

Next, there is an explanation about how this code works, and the reasons taken in this implementation:

First of all, total size of a RGB frame is calculated getting the video resolution and multiplicating it by the number of colour components per pixel in a RGB image. With the resolution you obtain the total number of pixels, and each one is codified with three bytes in RGB.

After that, we can calculate the size of each type of frame (I, P, B) because as we know from the documentation, a I frame is obtained compressing the original RGB frame by using an algorithm quite similar to JPEG (whose compression ratio in 20 times less than the RGB picture approximately). From the data in the documentation we know P frames size are three times less than I frames approximately, and B frames are six times less, so we can obtain the size of P and B frames.

Now, knowing the maximum payload in Ethernet (1500bytes) and adding the header size of the protocols we will need (MPEG2 payload header:12bytes RTP:12bytes UDP:8bytes IP:20), we can know the maximum payload in each packet: 1448 bytes. With this, we can know how many packets we need to transmit each type of frame, and the total packets.

According to documentation, a good sequence to transmit MPEG2 through a network is: IBBPBBPBBPBBPBBPBB, it is one I frame each 18 frames, one P frame after two B frames. Following this sequence, we save in a cell the type of frame for each frame.

After that, we read the original video and save each frame in a cell structure using *"read"* function from *"mmreader".*

To generate RTP header we have a look to the documentation to know what is inside each field.
We consider that we are not using padding or extension header, so P and X fields are "zero".
There is only one sender so CC is "one", and CSRC list is empty.
M bit is set when a new frame starts in a packet.
PT is 33 because this is the payload type for MPEG2.
SSRC identifier is a 32 bits number, it is a random number between 0 and 2^{32}-1 that identifies the sender.
Sequence is a 16 bits random number that is increased with each packet.
Timestamp in the first frame is a 32 bits random number. In the following frames this value is incremented according to the frame rate ((30*3000) / frameRate).

To pack in UDP and IP level we use through Mex files an external library written in C code (pnet.c).

To generate error packets we use a random function that selects what numbers of packet are going to be with errors according to given error probability. In the case of the delay, we generate random delays for each packet according to given average jitter variation and average delay. Then we check if those values are higher than maximum delay allowed according to frame rate; if they are, those packets will be considered lost.

Now for each lost packet, we check what kind of frame it contained in order to know from which previous frame we extract the missing information , the following frames affected by this lost and the number of bytes we will have to change in all these frames. In case a B frame is lost, only this frame is affected, but in case of P frame,

the following frames will have the same error until we receive a new P frame or an I frame, it is, along the two following ones. Something similar happens with a I frame, the error will be spread until we receive a new I frame (during the following 17 frames).
Then, I check what part of the frame was lost in order to know the starting byte I have to begin to change the erroneous frames from; and depending on the type of frame and the number of packets needed to transmit a complete frame I will calculate the total number of bytes to be changed (if it is the last packet of a frame there could be less data because the packet may be not completely full).

Once I have all this information, I can start changing bytes in the images. In the case I lose information of the first frame, it can not be obtained from a previous frame, so lost pixels will be black.

Finally, we save all frames in a video file using *"avifile"* and *"addframe"* Matlab functions.

Tests

To check that this code works properly, first we have use Matlab debugger in order to see if all calculated values, loops, conditions and so on are right.
After that, we have checked with some video samples if bytes in affected frames have been changed properly by the corresponding bytes of the right previous frame. I mean, if we have to modify in a frame P bytes from number 18000 until byte number 36000, we must check these bytes have been changed by the corresponding bytes of the previous P or I frame.

6. Conclusions

Conclusions of this project about the effect of delay and jitter on the perceived quality were:

- The effects of jitter on the degradation of the perceived quality are very similar to the effects of the losses. Moreover, the degradation is severe in the presence of low levels of loss and/or jitter, while higher values of these parameters do not degrade the perceived quality proportionately.

  Example: For 'foreman.mpeg' video, with 31.6 miliseconds if we vary jitter from 5% to 6%, triple frames are affected but if we change from 9% to 10%, not even the double frames are affected by degradation.

Figure 5. Frame with 5% jitter

Figure 6. Frame with 6% jitter

- Sequences with less temporal variation were more robust against the presence of jitter than those ones with higher temporal variation. That means that jitter affects more over video quality than delay.

  Example: executing the Matlab model for the same values of delay but different jitter, you can observe there is a big difference in degradation. For 'foreman.mpeg' video, with 31.5 miliseconds delay, and 5% and 6% jitter, we obtained some videos with 3 frames, and 9 frames affected respectively.
  In the opposite case, with the same jitter but different delay, the result was 9 and 13 frames affected by degradation respectively.

Figure 7. Frame with 5% jitter

Figure 8. Frame with 6% jitter

- We can observe that error probability affects just a little bit to video degradation.

  Example: For 'foreman.mpeg' , with 31.5 miliseconds of delay, 5% jitter and 10^(-9) error probability, only one frame is affected. In the other hand, if we keep the same values for the parameters, and multiply error probability by 10, we obtain only two frames were affected by degradation.

Figure 9. Frame with 10^(-9) error prob.

Figure 10. Frame with 10^(-9) error prob.

7. Summarize in Spanish ("Resumen en español")

Prólogo

Objetivo

El objetivo de este proyecto es implementar un modelo en Matlab para simular la degradación de video en *"streaming"* sobre redes de fibra óptica, en función de los parámetros de red (retardo, *"jitter"*, error de probabilidad, ancho de banda, etc). Para realizar dicho proyecto, fueron realizadas varias tareas resumidas en el párrafo que sigue a continuación.

Tareas

En primer lugar, fue necesaria una amplia documentación sobre cómo se realiza el *"streaming"* de video (protocolos, compresión MPEG) y sobre cómo los parámetros de red afectan a la pérdida de paquetes y a su vez dichas pérdidas a los fotogramas de un archivo de video.
Una vez conocido todo lo anterior, se implementó un modelo siguiendo las reglas extraídas y se realizaron varias pruebas con diferentes archivos de video y valores de los parámetros para sacar conclusiones.

1. Requisitos para el tráfico multimedia en tiempo real

De acuerdo a la referencia bibliográfica [20], estos son:

- Marcas de tiempo para asegurar una correcta reproducción y la eliminación del *"jitter*
- Numeración de secuencias para un correcto orden de los datos.
- Mecanismos de notificación y de compresión para aliviar los efectos de la congestión.

2. Protocolos

A continuación se exponen los protocolos relacionados con el *"streaming"* de video :

2.1 RTP (Real-Time Protocol)

RTP no asegura por sí mismo la entrega en tiempo real pero, de acuerdo a la referencia [14], proporciona medios para:

- Eliminación/reducción de *"jitter"* usando *"buffers"*.
- Reconstrucción del tiempo usando *"time-stamping"* (marcas de tiempo).
- Multiplexación y sincronización de varios flujos que pertenecen a la misma sesión multimedia.
- Identificación de contenido.
- Con la ayuda de RTCP, proporciona también fiabilidad y control de flujo/congestión.

2.1.2 Funcionamiento de RTP

De acuerdo a la referencia [14]:

El *"time-stamping"* es vital para las aplicaciones de tiempo real. El emisor fija el valor este campo de acuerdo al instante de tiempo en el que el primer octeto del paquete fue generado. Con esto, después de recibir los paquetes, el receptor usa dicho campo para reconstruir el tiempo original y reproducir el archivo de video o audio correctamente. También es usado para sincronizar diferentes flujos.
RTP no es responsable de la sincronización, sino que es la capa de aplicación la que tiene que encargarse de ello.

Dado que los datos de un fotograma de un video pueden ser repartidos en varios paquetes, es necesario fijar un número de secuencia que permita ordenar dichos paquetes. Este campo puede usarse también para detectar la pérdida de paquetes.

Existe también un campo que contiene el identificador de tipo de carga útil así como el tipo de compresión y codificación para que la aplicación sepa interpretar dichos datos.

Otra función que proporciona RTP es la identificación de la fuente, que permite por ejemplo en una conferencia conocer el usuario que está hablando.

Para establecer una sesión RTP, la aplicación define un par concreto de direcciones de destino (una dirección de red más un par de puertos para RTP y RTCP). Cada medio será transportado en una sesión separada, por ejemplo el audio y el video.

2.1.3 Cabecera RTP

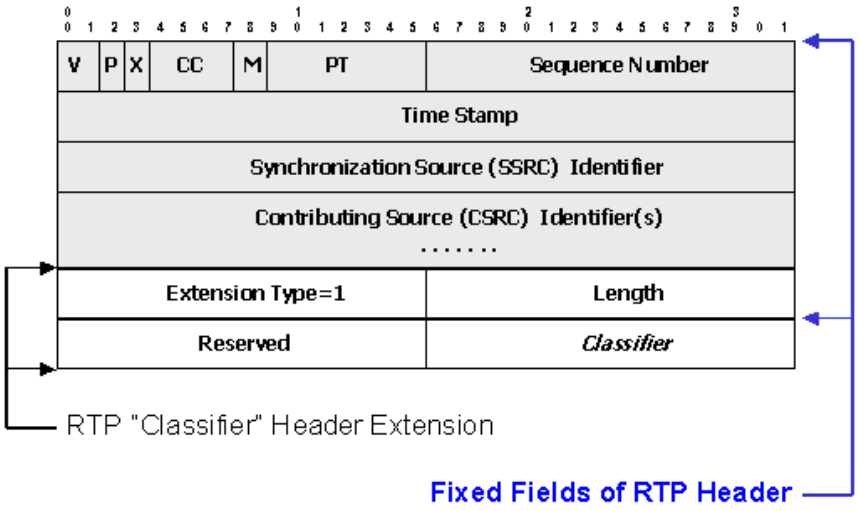

Figura 1. Cabecera RTP

Los primeros doce octetos de la cabecera están presentes en cada paquete RTP, mientras que la lista de CSRC (fuentes), está solo presente cuando se usan mezcladores.

A continuación se explica el significado de cada campo de acuerdo a la referencia [2] :

Versión (V): 2 bits. Versión de RTP. La versión actual es la 2 .

Relleno (P): 1 bit. Algunos algoritmos de codificación requieren bloques de tamaño fijo, por tanto, a veces es necesario el relleno. Si está activado, el ultimo octeto del relleno contiene el número de cuántos bytes de relleno deberían ser ignorados.

Extensión (X): 1 bit. Si está activado, a la cabecera fija le sigue otra de extensión, necesaria para algunos programas.

Contador de CSRC (CC): 4 bits. Indica el número de identificadores CSRC (fuentes) que siguen a la cabecera fija.

Marca (M): 1 bit. Indica un evento significante como el fin de un fotograma, cuando alguien empieza a hablar en una conversación, etc.

Tipo de Payload (PT): 7 bits. Identifica el formato de la carga útil en un paquete RTP.

Número de secuencia: 16 bits. Sirve para detectar la pérdida de paquetes en el receptor. Dicho número es incrementado en uno cada vez que un paquete RTP es enviado. El valor inicial es fijado aleatoriamente.

Marca de tiempo (*"Timestamp"*): 32 bits. Determina el instante de muestreo del primer octeto del paquete RTP. Se usa para la sincronización y el cálculo del *"jitter".*

SSRC: 32 bits. Número aleatorio usado para distinguir las diferentes fuentes en una sesión RTP. Indica la fuente de los datos.

Lista de CSRC (fuentes): permite hasta 16 fuentes de 32 bits cada una.

2.2 RTCP (Real-Time Control Protocol)

2.2.1 Ventajas para el *"streaming"*

• Proporciona información sobre congestión y la calidad de servicio de la transmisión.
• Tiene conocimiento sobre el número de participantes en una conferencia y de esta forma, estima la tasa a la que los paquetes deberían ser enviados. También permite un acceso controlado a más participantes.
• Proporciona la identificación de participantes usando nombres canónicos.

2.3 RSVP (Resource Reservation Protocol)

De acuerdo a la referencia [2]:

RSVP es un protocolo de control de red que permite a las aplicaciones obtener diferentes calidades de servicio (QoS); sin embargo, RSVP no es un protocolo de encaminamiento sino que trabaja en conjunto con este tipo de protocolos.

2.3.1 Características

De acuerdo a la referencia [3], son:

• Reserva recursos para flujos de datos en una dirección.

• Soporta multicast y unicast. En multicast cada receptor puede tener su propio nivel de QoS.
• Es transparente para la capa IP, funciona tanto para la v4 como para la v6.

2.4 RTSP (Real-Time Streaming Protocol)

RTSP es usado para proporcionar un servicio completo de *"streaming"* sobre internet.

2.4.1 Ventajas que presenta

De acuerdo a las referencias [12 y 13] las principales ventajas son:

• Extensible: puedes añadir nuevos métodos y parámetros.
• Independencia del transporte.
• Multiservidor.
• Balanceo de carga.
• Control sobre la calidad de servicio, eficiencia y medidas
• Control de dispositivos, por ejemplo hacer *"zoom"* con una cámara.
• Interoperabilidad entre clients y servidores de diferentes fabricantes.

2.5 SIP (Session Initiation Protocol)

De acuerdo a la referencia [23]:

SIP es usado para establecer modificar y terminar una session multimedia basada en RTP como por ejemplo una videoconferencia. SIP es más simple que H.323 pero adecuado para Internet.

SIP proporciona las siguientes funciones:

- Localización de usuarios.
- Disponibilidad del usuario.
- Capacidades del usuario para ver incompatibilidades.
- Gestión de la sesión.

3. Compresión MPEG

El flujo de bits de MPEG-2 es básicamente una serie de *"frames"* codificados uno detrás de otro.

La codificación puede variar de un *"frame"* a otro. Existen tres tipos diferentes de *"frames"*, llamados I, P y B.

***"Frame"* I:** contiene una imagen completa comprimida de manera similar a JPEG.
***"Frame"* P:** usa la correlación entre el *"frame"* actual y el anterior *"frame"* I o P para comprimir la información.
***"Frame"* B:** contiene la diferencia de información respecto a un *"frame"* anterior y a uno futuro.

Aproximadamente un *"frame"* P será el 30% de un *"frame"* I y uno del tipo B será el 15% de un I.

De acuerdo a la referencia [11]:

El orden entre los *"frames"* I, P y B es bastante flexible pero una opción recomendable para *"streaming"* es esta: ***I B B P B B P B B P B B P B B P B*** , y así sucesivamente.

4. Influencia de los parámetros de red en *"streaming"*

La variación del retraso y la pérdida de paquetes son los parámetros clave para determinar si una red puede soportar *"streaming"* de forma correcta. Por tanto, estas características pueden usarse para determinar cómo de bien una red puede transportar video en diferentes condiciones de carga, configuraciones, etc.

Ancho de banda

Un ancho de banda insuficiente provocará la pérdida de paquetes, lo que conduce a una reproducción del video con saltos.
Gracias a RSVP, se puede establecer una reserva de ancho de banda para aplicaciones que requieren una entrega de datos predecible en Internet. RSVP garantiza una calidad aceptable de reproducción incluso en situaciones de carga.

Pérdida de paquetes

La pérdida de paquetes es la principal causa de degradación de la calidad. Debido a la forma en la que se codifica el video, la pérdida de un paquete puede afectar a los siguientes *"frames"* . Por ejemplo, con la compresión MPEG, si se pierde un paquete que contiene un *"frame"* I, esto resultará en la degradación de los siguientes *"frames"* hasta que recibamos un nuevo *"frame"* I.

En el estudio de Liang Apostolopoulos y Griod, se llega a la conclusión de que la distorsión total debida a dos pérdidas consecutivas es más alta que si las pérdidas son independientes, es decir, separadas por más *"frames"* .

Retardo y *"jitter"*

Si los paquetes sufren un retraso debido a la red, como muestra la figura 2, algunos paquetes llegan en ráfagas con retrasos entre paquetes más pequeños que cuando fueron transmitidos como el grupo "1 Second" , mientras que otros sufren un retraso como muestra el grupo "3 Seconds" con un retraso mayor entre paquetes que cuando fueron transmitidos desde la fuente. Esto es viene derivado del "*jitter"*, que es la diferencia de tiempo entre cuando un paquete llega a su destino y cuando estaba previsto llegar.

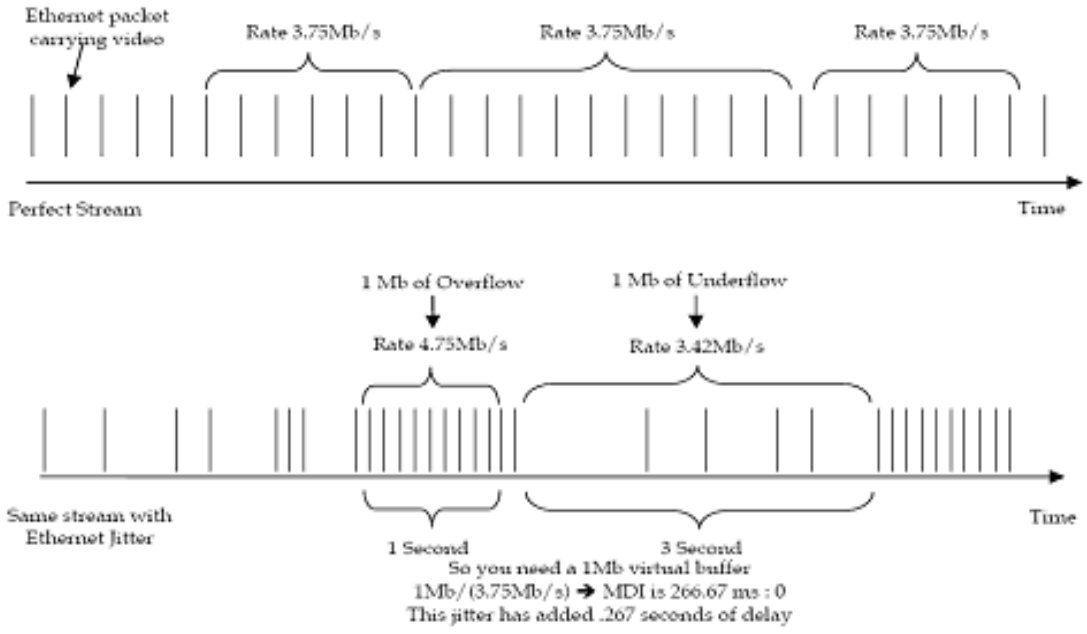

Figura 2. Retraso de paquetes en una red

Un receptor (decodificador) reproduciendo un video, debe adecuar los tiempos de recepción del flujo de entrada, haciendo *"buffering"* de los datos recibidos antes de tiempo y asegurar que hay suficientes datos almacenados en ese *"buffer"* para cuando los paquetes lleguen con retraso.

Hay tres diferentes situaciones cuando hacemos *"streaming"* de video relacionados con el retardo y el *"jitter":* sin pérdidas y sin *"jitter",* canal con *"jitter"* y canal con pérdidas.

En el caso de pérdidas, aparte del efecto que se produce de congelar la imagen, se añade el problema de la necesidad de los *"frames"* anteriores para decodificar los siguientes *"frames"*. Intuitivamente, se puede ver entonces que los efectos de las pérdidas y el *"jitter"* en la percepción del usuario son similares. Esta similitud condujo a realizar un estudio para comparar los efectos combinados de estos factores (experimento de Claypool and Tanner en 1999). El experimento consistía en generar secuencias de video distorsionadas con diferentes niveles de pérdida y *"jitter"* , y ordenarlas después en base a la calidad de las mismas. La conclusión que se obtuvo fue la siguiente: existe una alta correlación entre los valores MOS obtenidos y el número medio de distorsiones detectadas.

5. Modelo en Matlab

La finalidad de este proyecto es implementar un modelo en Matlab para simular la degradación en redes de fibra óptica producida al hacer "*streaming"* en función de los parámetros de red (retardo, *"jitter"*, probabilidad de error, ancho de banda, etc).

En este modelo, se consideran sólo los tres primeros de estos parámetros dado que por ejemplo, como se mencionó antes, gracias al protocolo RSVP, el ancho de banda disponible no afectará a la calidad del video. Estos tres parámetros pueden ser modificados, cambiando el valor de las variables "error_prob" , "average_jitter" y "average_delay".

Funcionamiento del modelo

1) Cálculo del tamaño de cada "frame" y el número de paquetes necesarios para enviar cada uno:

El tamaño total de un *"frame"* RGB se calcula obteniendo la resolución del video y multiplicándola por el número de colores que compone cada píxel en una imagen RGB, que se codifica con 3 bytes. Después, procedemos a calcular el tamaño de cada tipo de *"frame"* (I, P o B) ya que como sabemos de la documentación, uno del tipo I se obtiene comprimiendo una imagen RGB de forma similar a JPEG, por lo que esta ocupará aproximadamente veinte veces menos. Un *"frame"* P tendrá aproximadamente, un tamaño tres veces más pequeño que uno del tipo I, y uno del tipo B, seis.

A continuación, sabiendo el máxima carga útil en Ethernet (1500 bytes) y sumando a esta el tamaño de las cabeceras de los protocolos que utilizaremos (MPEG2: 12 bytes , RTP: 12 bytes, UDP: 8 bytes, IP: 20bytes), se puede conocer la máxima carga útil en cada paquete, que será 1448 bytes. Con esto, es posible saber cuántos paquetes necesitamos para transmitir cada tipo de *"frame"* y los paquetes totales.

De acuerdo a la documentación, una buena secuencia para transmitir MPEG2 a través de una red es IBBPBBPBBPBBPBBPBB, es decir, un *"frame"* del tipo I por cada 18 *"frames"*, y uno del tipo P cada dos *"frames"* B.

2) Extracción de cada "frame" del archivo de video:

Para realizar esta tarea, se procede a leer el video y almacenar cada *"frame"* en una estructura *"cell"* de Matlab utilizando la función *"read"* del paquete *"mmreader"* de Matlab.

3) Generación de la cabecera RTP:

Para generar la cabecera RTP, de acuerdo con la documentación, se fijarán los siguientes valores:

- Los campos de "padding" y extensión serán cero ya que no nuestro caso no se usa "padding" ni la cabecera de extensión.

- Hay solamente un emisor así que el campo CC está a uno y la lista de CSRC vacía.
- El bit M se activa cuando ese paquete contiene un nuevo *"frame".*
- El campo PT es 33 porque representa el tipo de carga útil en MPEG2.
- El campo SSRC de 32 bits es un identificador de emisor, es decir, un número aleatorio entre 0 y 2^{32}-1
- El número de secuencia es un número de 16bits que se incrementa al enviar un paquete.
- La marca de tiempo en el primer *"frame"* es un número aleatorio de 32bits. En los siguientes *"frames"* este valor es incrementado de acuerdo a la tasa de frames: (30*3000) / frameRate

4) Generación de paquetes UDP e IP:

Para empaquetar los frames usaremos a través de archivos Mex una librería externa escriba en código C llamada "pnet.c"

5) Cálculo de los paquetes de error o perdidos debidos a la probalidad de error, retraso o "jitter":

Para establecer cuáles serán los paquetes de error usaremos una función aleatoria de acuerdo a una probabilidad de error fijada por una variable. En el caso del retardo, se generan tiempos de retardo aleatorios para cada paquete de acuerdo a la variación media del *"jitter"* y del retraso medio, parámetros fijados por las respectivas variables. Después de esto, se comprueba si esos valores superan el máximo retardo permitido de acuerdo a la tasa de *"frames";* si es así, dicho paquete será considerado como perdido.

6) Degradación de "frames" :

Para cada paquete perdido, se comprueba qué tipo de *"frame"* contenía para saber de qué *"frame"* anterior podemos extraer la información que falta, qué *"frames"* siguientes se verán afectados por esta pérdida, y el número de de bytes que tendremos que modificar en esos *"frames".* En el caso de que se pierda un *"frame"* B, sólo se verá afectado dicho *"frame"*, pero en el caso de uno

del tipo P, los siguientes *"frames"* hasta la recepción de uno nuevo del tipo P o I tendrán el mismo error. Lo mismo sucede si el error ocurre en un *"frame"* I, ya que el error se extenderá al resto de los siguientes *"frames"* hasta que recibamos uno nuevo del tipo I (durante los siguientes 17 *"frames"*).

Después de esto, se comprueba qué parte del *"frame"* se perdió para saber a partir de qué byte hay que empezar a cambiar los datos; y dependiendo del tipo de *"frame"* y del número de paquetes necesarios para transmitir dicho *"frame"* , se calculará el número total de bytes afectados por el error (por ejemplo si es el último paquete que contiene un trozo de un *"frame"* puede haber menos datos ya que dicho paquete puede no estar lleno). Una vez que se tenga toda esta información, se puede empezar a modificar los bytes afectados en las imágenes. En el caso de perder el primer *"frame",* no se puede recuperar nada de información de un *"frame"* previo, así que esto se traducirá en una visualización en color negro de la zona afectada.

Finalmente, procederemos a almacenar todos los *"frames"* en un archivo de video usando las funciones de Matlab *"avifile"* y *"addframe".*

6. Conclusiones

Las conclusiones que se pueden obtener sobre los efectos del retardo y el *"jitter"* en la calidad del video son:

- Los efectos del *"jitter"* en la degradación de calidad percibida son muy similares a los efectos de las pérdidas. Además, la degradación es considerable en el caso de niveles bajos de pérdida o *"jitter",* mientras que con valores altos de estos parámetros no se degrada la calidad proporcionalmente.
- Las secuencias con menos variación temporal fueron más robustas frente al *"jitter"* que aquellas con una variación mayor. Esto significa que el *"jitter"* afecta más a la calidad que el retardo.

8. References

[1]http://www.geocities.com/intro_to_multimedia/RTP/index.html Date: October 2008
[2] www.isi.edu/div7/rsvp/overview.html Date: October 2008
[3] http://www.hep.ucl.ac.uk/~ytl/qos/rsvp_01.htm Date: October 2008
[4] www.ietf.org
[5] http://iie.fing.edu.uy/investigacion/grupos/artes/publicaciones/hetnet05.pdf Date: October 2008
[6] www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/rsvp.htm Date: October 2008
[7-]http://www.tml.hut.fi/Opinnot/Tik110.551/1997/iwsem.html Date: November 2008
[8] www.cs.columbia.edu/~hgs/rtp/ Date: November 2008
[9]http://stackoverflow.com/questions/361943/why-does-rtp-use-udp-instead-of-tcp Date: November 2008
[10] www.freesoft.org/CIE/RFC/1889/13.htm Date: November 2008
[11] www.mpeg.org Date: November 2008
[12]digitais.ist.utl.pt/ec-ris/video%20streaming/Basic%20Streaming%20Technology.doc Date: November 2008
[13]www.csee.umbc.edu/~pmundur/courses/CMSC691M-04/manoj1.ppt Date: November 2008
[14]http://www.cse.wustl.edu/~jain/cis788-97/ftp/ip_multimedia/index.htm Date: November 2008
[15]http://www.bbc.co.uk/rd/pubs/papers/paper_14/paper_14.shtml Date: November 2008
[16]http://www.cs.columbia.edu/~hgs/rtsp/draft/draft-ietf-mmusic-stream-00.txt Date: November 2008
[17]http://helios.etsit.upv.es/asig/5%BA/tel_espa/pract_16/mpeg/mpegi.htm Date: October 2008
[18] http://docs.hp.com/en/5992-1950/ch01s02.html Date: October 2008
[19] http://www.freesoft.org/CIE/RFC/1889/13.htm Date: November 2008
[20]http://stackoverflow.com/questions/361943/why-does-rtp-use-udp-instead-of-tcp Date: November 2008
[21]http://medusa.sdsu.edu/network/tcpip/documents/cisco-rsvp.pdf Date: November 2008
[22]http://searchunifiedcommunications.techtarget.com/generic/0,295582,sid186_gci1310584,00.html Date: November 2008
[23]http://www.callcentermagazine.com/GLOBAL/stg/commweb_shared/shared/article/showArticle.jhtml?articleId=8700868&pgno=6 Date: November 2008