QOS on the XScale Ipaq Over 802.11b
CSC 714 Term Project

Anita Nagarajan
Guru Ranganathan
Vasanth Asokan

Table of Contents

1. Abstract

Handheld devices are an example of devices with constrained hardware and software resources, which are nevertheless expected to perform soft real-time multimedia tasks. The 802.11b is the IEEE standard for wireless communication within short distances. A combination of the two is interesting in terms of performance and quality issues. This project aims to evaluate certain QOS parameters for an XScale Ipaq running over an 802.11b wireless network.

2. Introduction

The Ipaq features an Intel Xscale processor running at 400 MHz with a 48 MB Flash ROM and 64 MB of SDRAM. They run the Windows Pocket PC 2002 Operating System (based on Win CE). Win CE is the embedded version of the popular desktop version.
Streaming multimedia applications are an example of a soft real-time system.The basic idea of the project is to setup an experimental environment to test QoS parameters on the Ipaq. This is achieved by developing custom multimedia and background applications.

3. Experimental Setup

3.1 Multimedia Client:

A streaming video client is a good test bed for QOS measurements. This is due to two facts.

1) Video frames are typically larger samples when compared to pure audio samples.
2) Decoding and rendering video frames is a computationally complex task with real-time requirements.

3.1.1 MPEG Player

MPEG-1 is a popular video format which uses advanced compression techniques. MpegTV SDK is a freeware SDK which can be used by developers to quickly build video playing clients [MTV]. The daunting task of writing the decoding and rendering thread is abstracted away by the library. The SDK is also especially interesting due to the fact that it allows one to write a separate layer (as a DLL) which can take care of providing an input video stream to the player. With respect to this project, the network streaming layer was built within this DLL. The SDK also provides a hook for retrieving the frame rate being played. Thus, there is a *player thread* within the multimedia client which performs the decoding and rendering tasks.

3.1.2 RTP/RTSP

RTSP, the Real-Time Streaming Protocol [RTSPRFC], is an application-level protocol for control over the delivery of data with real-time properties. It establishes and controls either a single or several time-synchronized streams of continuous media such as audio and video. A server maintains a session labeled by an identifier. An RTSP client may open and close reliable transport connections to the server or use a connectionless protocol like UDP. RTP, the Real-Time Transport Protocol [RTPRFC] is the Internet standard protocol for the transport of real-time data such as video and audio. It provides end-to-end network transport functions suitable for applications transmitting real-time data. It can be implemented on top of standard UDP services over unicast/multicast networks. It provides payload type identification, sequence numbering, time-stamping and delivery monitoring services.

The client uses a mini-implementation of RTP/RTSP as the streaming protocol to talk to the streaming server. It first initiates an RTSP session to setup delivery of a particular media file. The server then uses RTP to transmit the media file

The RTP header has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The timestamp and sequence number fields are the main fields of interest as they aid in QOS parameter calculations. This is described further below. The RTP layer is implemented as a separate *RTP thread* in the multimedia client.

3.1.3 Calculation of QOS Parameters:

The following QOS parameters are computed from the RTP header of each packet received:

1. Packets lost
2. Packets lost per interval of time
3. Packets received
4. Packets received per interval of time
5. Delay
6. Jitter
7. Total bytes received

The methods used for calculating these parameters are as specified in the RFC for RTP.

3.1.4 Network Load

Artificial network load is simulated by another *load thread* within the multimedia application. It can be programmed to offer varying load across the network.

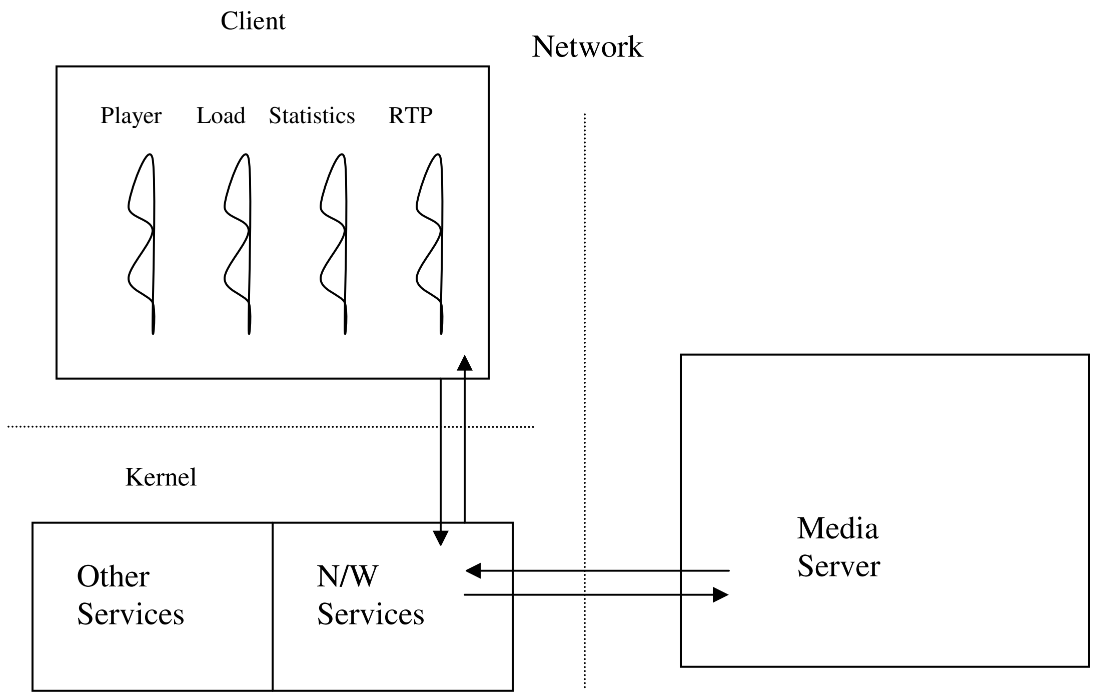
3.2 Multimedia Server

An evaluation version of the Real Networks Helix Streaming Server was used for a server. Helix supports RTP/RTSP streaming of MPEG files. The Server also provides logging and monitoring of served out media streams, which were useful for QOS measurements.

3.3 Wireless Infrastructure:

NC State University's 802.11b Nomad wireless computing environment was used as the wireless infrastructure for the experiment.

3.4 Experimental Setup Diagram



4 Experiments

The objective of the experiments can be stated as

1. Observing the effect of varying load on QOS.
2. Observing the effect of varying priorities of the threads on the QOS.

Objective(1) can be achieved by running the custom background thread along with the media player thread at the same priority and varying the background traffic parameters and observing its effect on the QOS parameters, which are measured in accordance with the methods stated in the RFC. Objective (2) can be achieved by creating the background and the media player thread with different priorities.

The following series of experiments were performed to evaluate QOS metrics.

1. Equal Priorities
    a. No Background load
    b. Background load of 400kbps
    c. Background load of 700kbps
2. Different Priorities(at 400kbps constant load)
    a. Equal priorities
    b. RTP thread having higher priority
    c. Background thread having higher priority.

5 Results

5.1 Varying background load, Equal priorities for RTP and Load threads

[Appendix A.1]

The graphs obtained as a result of varying the offered load (*noload*, *400kbps*, *700kbps*) are given in appendix A1.

Cumulative Packets Lost: With no background load, the cumulative number of packets lost is very low. As background load is increased, the number of packets lost increases. [Appendix A.1.b] This is due to the fact that both the RTP and load thread run at the same priority level , but the background load pumps in more data into the network, thus hogging the network and the Ipaq's network buffers, which leads to more packet loss experienced by the RTP thread. This behavior is seen in the graph (AppendixA1.a).

Similar conclusions can be arrived at by observing other graphs like

Cumulative Packets Received: [Appendix A1.d].
As can be observed in the graph , the *noload* line is above the other two lines at all instants of time indicating that more the amount of data is received (with less loss) for the entire course of time. Similar behavior is observed for the *400kbps* line as compared to

the *700kbps* line. This further strengthens our reasoning as stated for the Cumulative Packets lost graph.

Jitter: [Appendix A1.f]
From the graph, careful observation shows that the jitter is generally high when the offered load is more and decreases as the offered load is decreased, with minimum reaching

5.2 Constant Load, Varying Priorities for RTP and Load threads

The priorities of the *RTP* and *load* threads were varied and the effects on the QoS parameters of the *player* thread were observed. The offered load was not varied. The graphs are given in the appendix [Appendix A2.]

Cumulative Packets Lost: [Appendix A2.b]
A clear distinction between the various cumulative packets lost statistics can be seen when they are run with various priorities. As expected the *player* thread experiences more packet loss when the background *load* thread has more priority compared to the *RTP* thread, and comparatively lower packet loss when run with equal priority, which further decreases when run with higher priority. This seems inline with the expectation that a thread must run better if it is run with higher priority.

Cumulative Packets Received: [Appendix A2.d]
When the RTP thread is run with higher priority packets are received more often and thus the graph is raised as compared to the other two scenarios. The data values do represent a higher receive rate for the other two scenarios but due to the granularity of the graph it is not clearly visible.

Packets Received per Interval: [Appendix A2.e]

This follows the trend of the cumulative packets received graph**.**

Jitter: [Appendix A2.f]
Under close observation of the jitter graph, it is seen that the jitter experienced by the RTP thread decreases as the priority of the background load thread increases.

The results seem to be inline with the underlying concepts (within the usual experimental error margins).

6. Limitations

The approach taken here has many common faults.

- The experimental results presented here are not statistical averages of multiple runs as they should be ideally.

- The behavior of the wireless network due to varying loads due to other users and the various wireless specific network issues have not been considered.
- There were some issues with thread priorities and Win CE handling of them that have also been abstracted away.
- Virtual memory and other such aspects have also been ignored away in the evaluation.

7. Open/Unresolved Issues

7.1 QOS Measurement of MPEG Streams Using RTP

The timestamp field in the RTP header nominally represents the sampling instant of the corresponding packet and is normally monotonically increasing [RTPRFC]. Jitter and Delay calculations interpret this field in the header as the sampling and/or transmission time at the receiver. It is also the fact that the RTP timestamp may not be monotonically increasing in the case of interpolated MPEG B video frames [RTPRFC]. B frames are predictive frames which represent an interpolation of the last frame and a frame to come. These frames appear out of order by default in the RTP MPEG video stream. i.e., B-frames have an RTP timestamp greater than some packets that in reality actually follow them in the network [MPGFORMAT]. Therefore it is inconclusive how jitter and video calculations incorporate this quirk. It is quite clear though, that jitter and delay calculations are quite correct for other streams which do not have this property.

7.2 Merging Audio and Video Streams at the Client

RTP streaming of MPEG files works by splitting the audio and video streams and transmitting them as separate RTP streams. Usually decoders are designed so that there are separate video and audio decoding/rendering threads with some form of synchronization between them. This feature is abstracted away in MpegTV SDK. As a result, only a single composite MPEG-1 stream can be fed to the player. A way in which the original MPEG stream could be recomposed from two separate media streams was not found. Though a tractable problem, it is not a very sensible way of doing things. It is just a constraint imposed by using a common SDK.

8. Work Split

Get familiar with the device and the development environment	- All of us
Write/test small initial applications on eVC++	- Guru
Look for open source audio/video decoding engines for WinCE	- Vasanth
Research RTP, RTCP, RTSP	- Anita
Investigate MpegTV SDK	- Anita & Guru
Develop the media decoding/rendering application,	
- The SIH(Stream Input Handler) plug-in(DLL)	- Vasanth
- The Player	- Anita
- Logging information	- Vasanth

Implement RTP layer(client)	- Anita
Implement RTSP	- Vasanth
Evaluation of QOS metrics	- Guru
Develop application to simulate load - client and server	-Vasanth
Develop timeserver application	- Anita
Web Maintenance	- All of us
Analysis and Final Report	- All of us

8. Conclusion

These experiments give an insight into the variation of the QOS parameters with variation in other factors like background load and thread priorities. This work is not complete in all aspects and there is scope for more detailed and in-depth study.

Acknowledgements

The following people and resources were of great help.

People,
David Jebousek, Real Networks.
Ross Finlayson, Live.com

Resources

RTP Tools	http://www.cs.columbia.edu/IRT/software/rtptools/
MpegTV Forum	http://forum.pocketmovies.net/
Live.Com	http://www.live.com

References

[RTPRFC] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[RTSPRFC] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[MPGFORMAT] D. Hoffman, G. Fernando, V. Goyal, M. Civanlar "RTP Payload Format for MPEG1/MPEG2 Video" January 1998

[MTV] http://mpegtv.com/sdk

Appendix A: Results

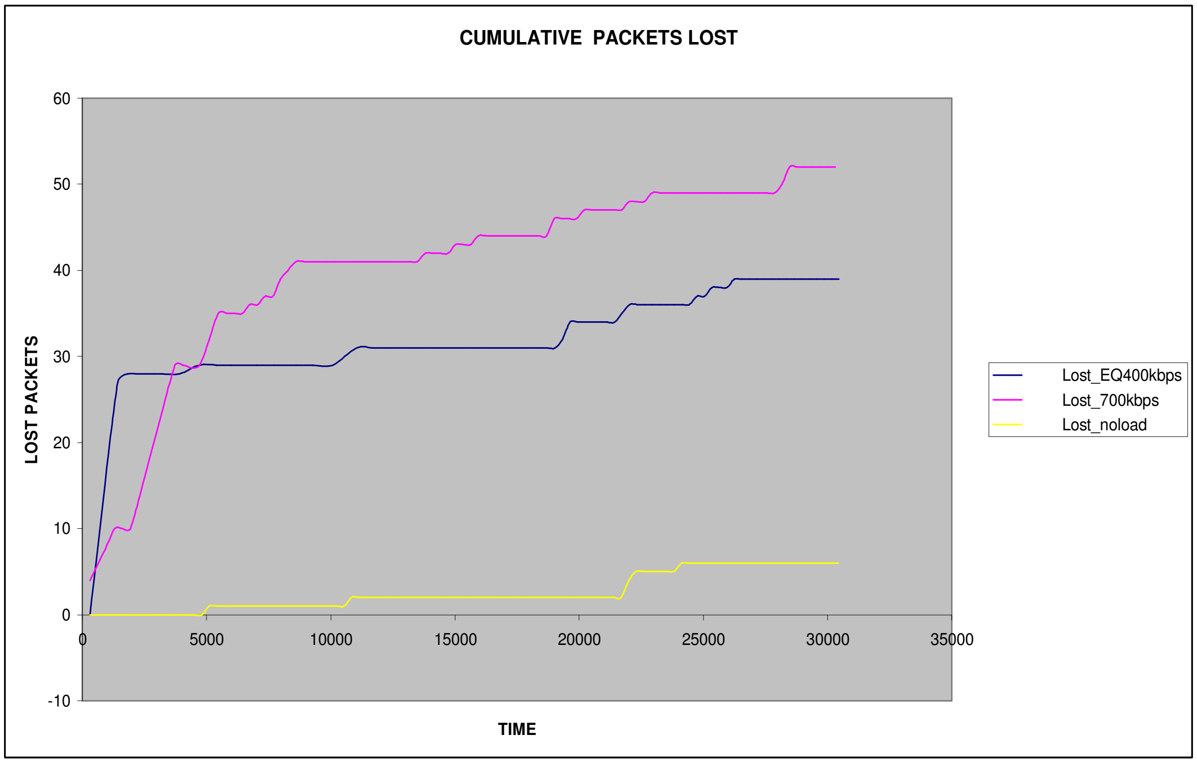
These are the results that was obtained as the part of running the experiment for various values of the offered load, priorities of the processes running.

A.1. Varying the offered Background Load with Equal priorities

a.) Legend:

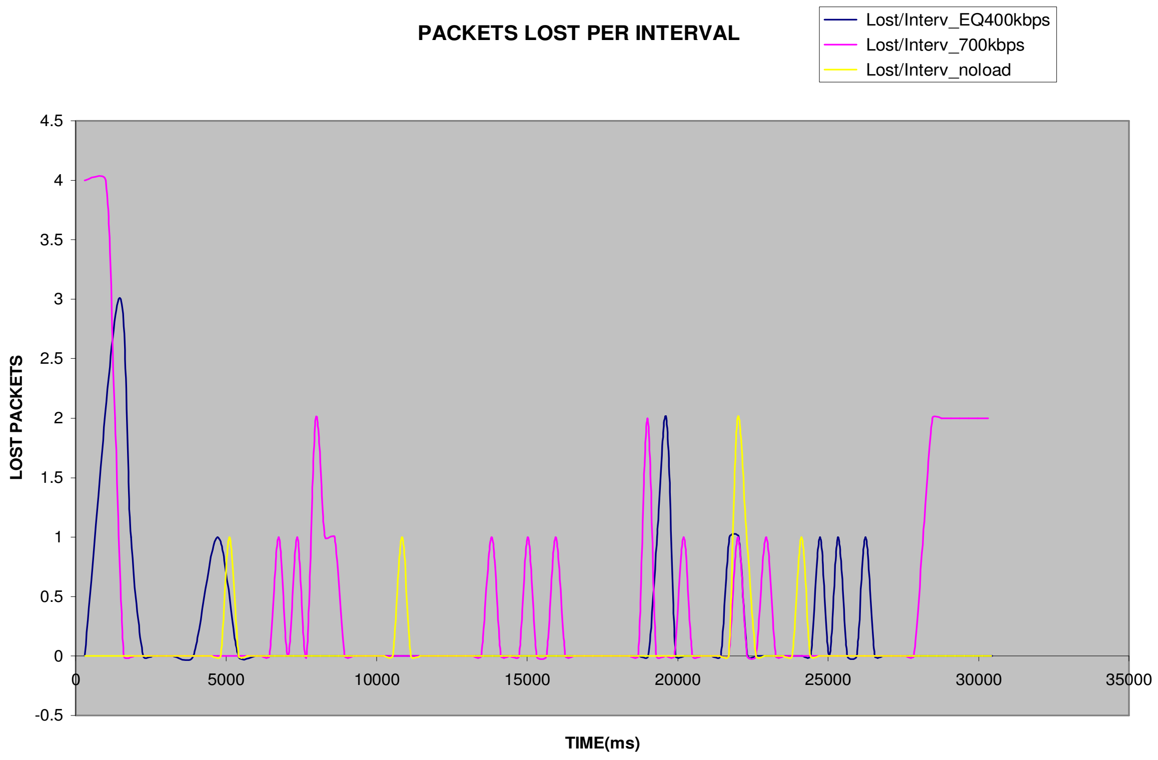
1. <Statistics>_EQ400kbps represents the statistics graph when the offered load is 400 Kbps. (BLUE COLOR GRAPH)
2. <Statistics>_700kbps represents the statistics graph when the offered load is 700 kbps (PINK COLOR GRAPH)
3. <statistics>_noload represents the statistics graph when no load is offered . (YELLOW COLOR GRAPH)

b) Cumulative Packets Lost



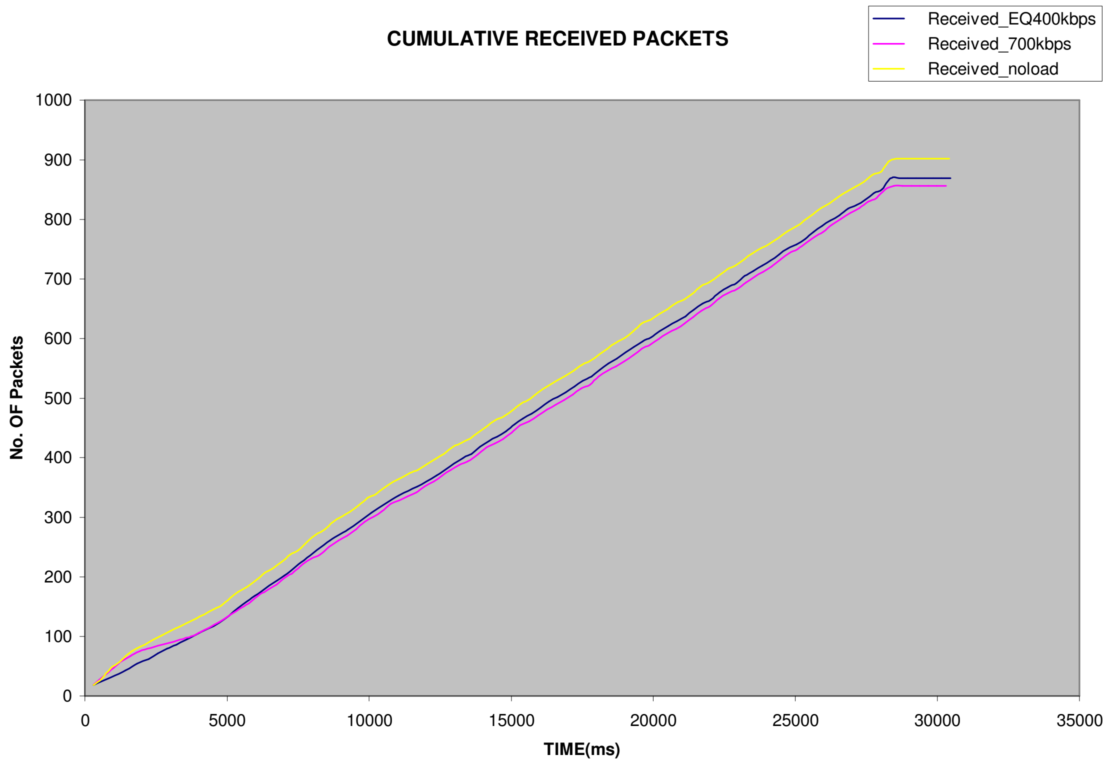
PINK: 700Kbps, BLUE: 400Kbps Yellow: NO Load

c) Packets Lost Per Interval

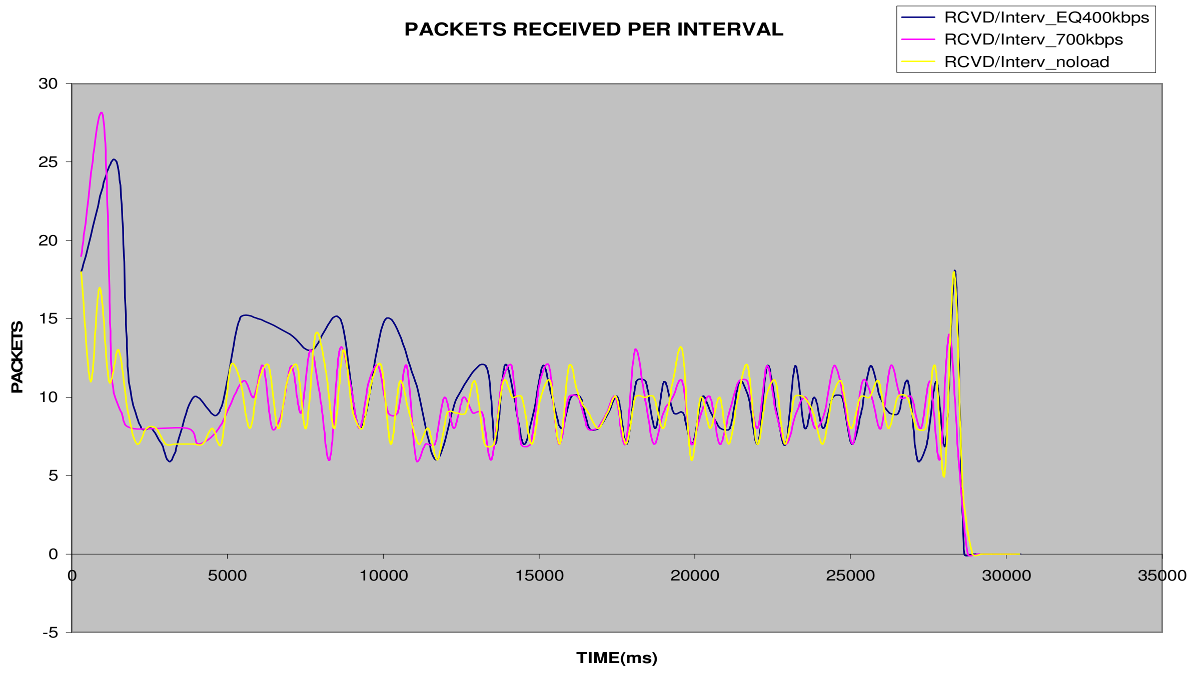


PINK: 700Kbps, BLUE: 400Kbps Yellow: No Load

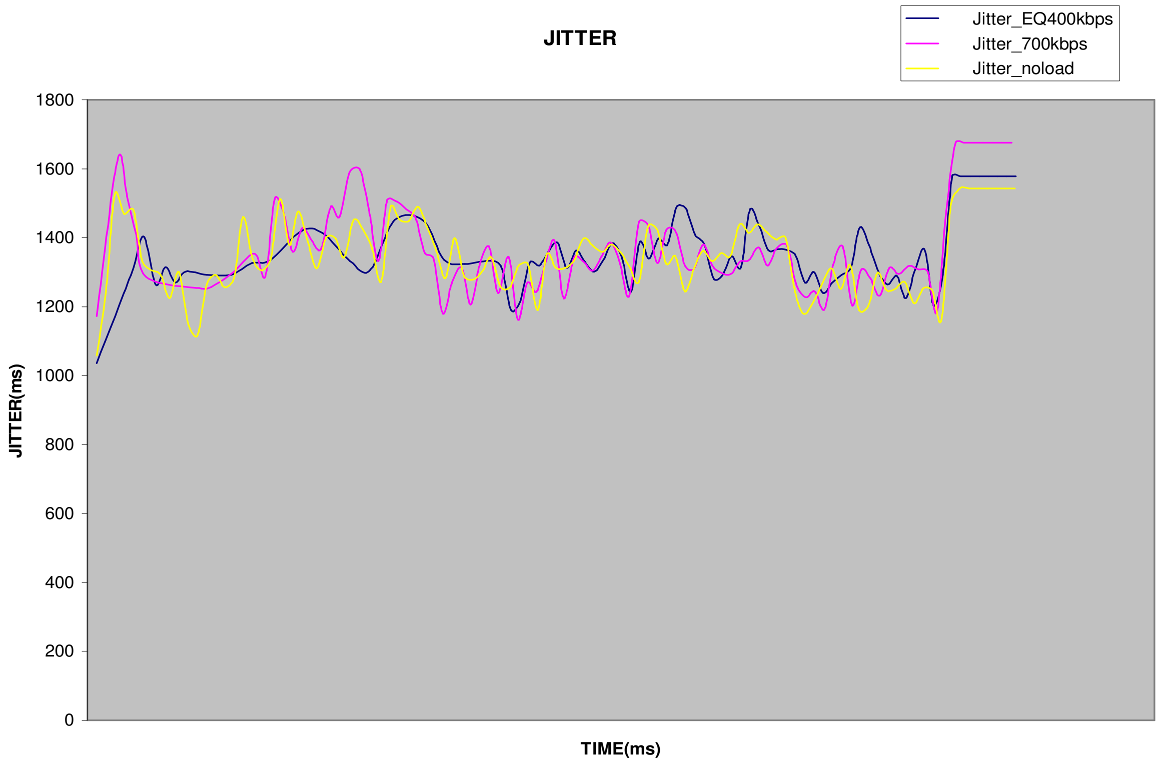
d) Cumulative Packets Received



e) Packets Received Per Interval
PINK: 700Kbps, BLUE: 400Kbps Yellow: NO Load

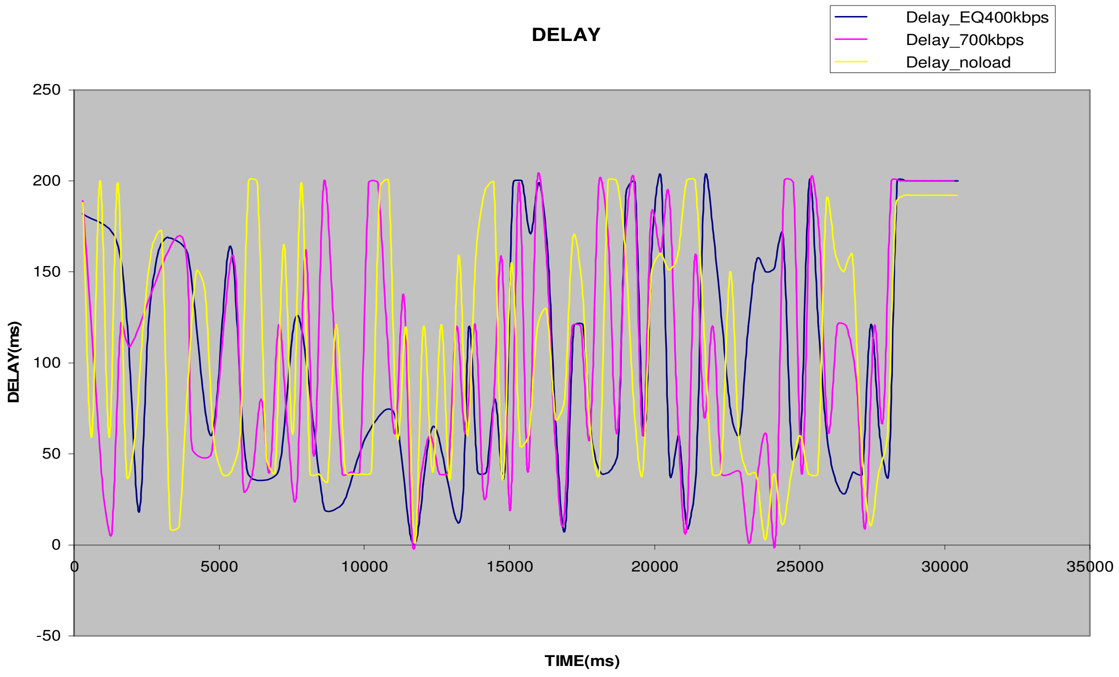


f) Jitter :



PINK: 700Kbps, BLUE: 400Kbps Yellow: NO Load

g) Delay

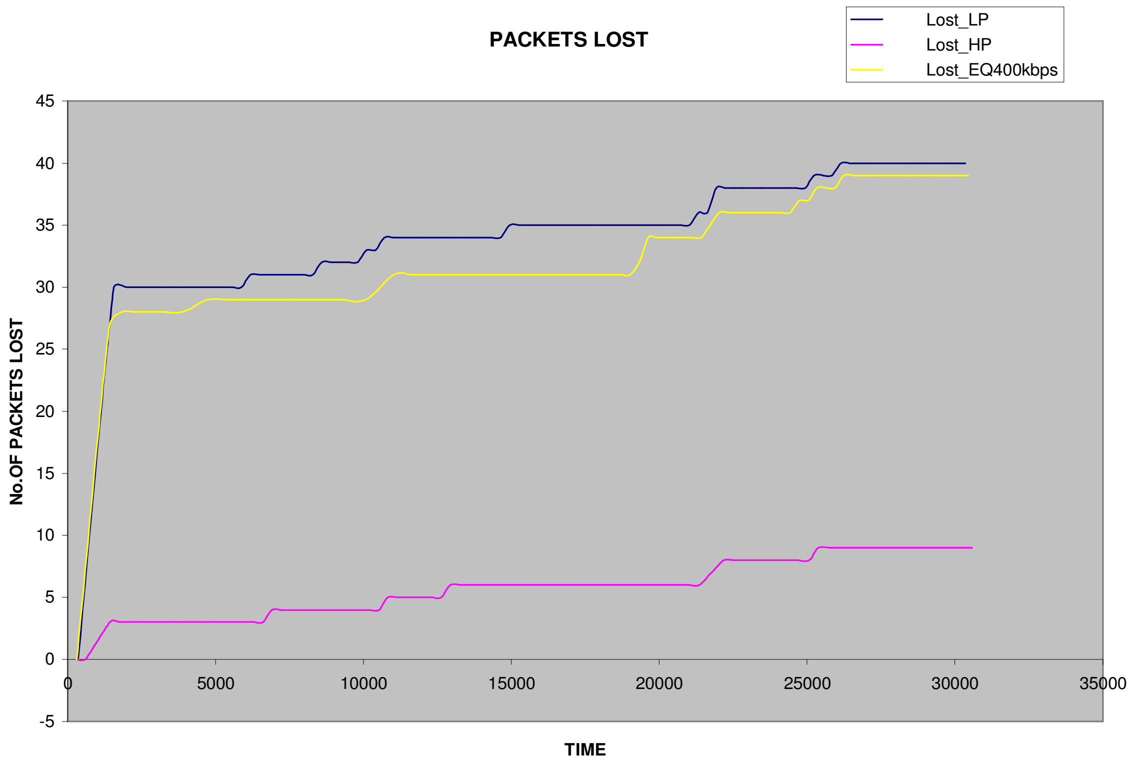


A.2. Results for varying the Priorities of the processes with constant Load (400 kbps)

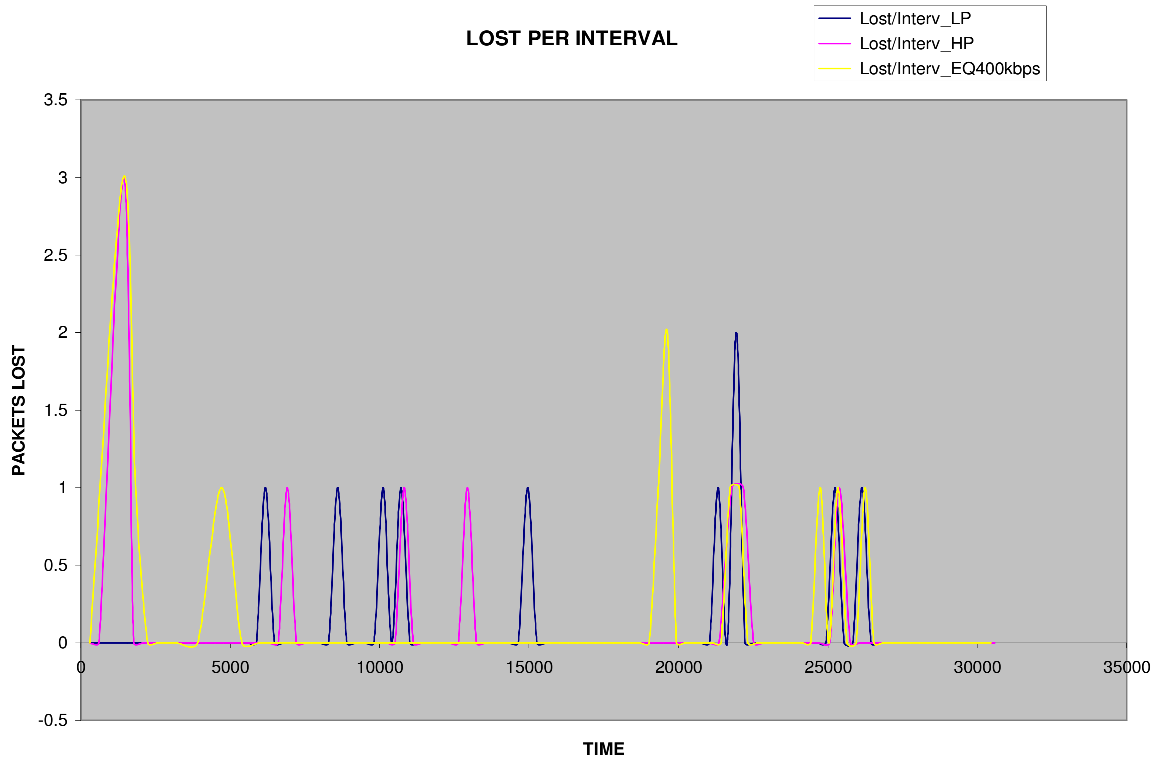
a) Legend

1. **<Statistics>_EQ400kbps** represents the statistics graph when the offered load is 400 Kbps.
2. **<Statistics>_HP** represents the statistics graph when the RTP process runs with higher priority as compared to the background load.
3. **<Statistics>_LP** represents the statistics graph when the RTP process(thread) runs with lesser priority as compared the background load.

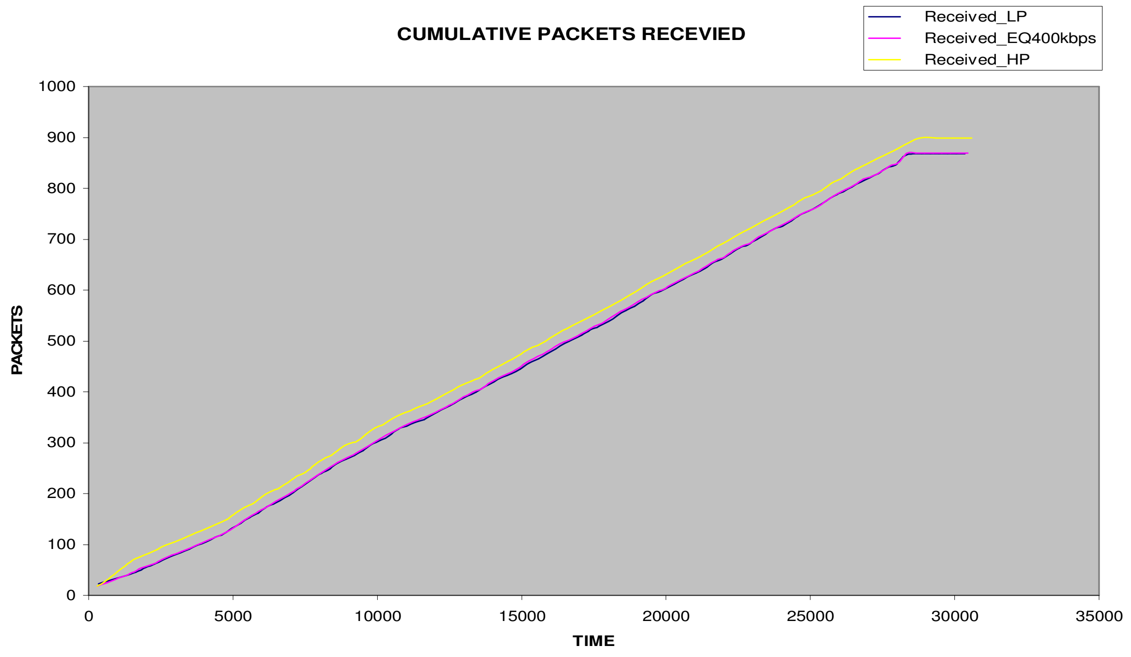
b) Cumulative Packets Lost



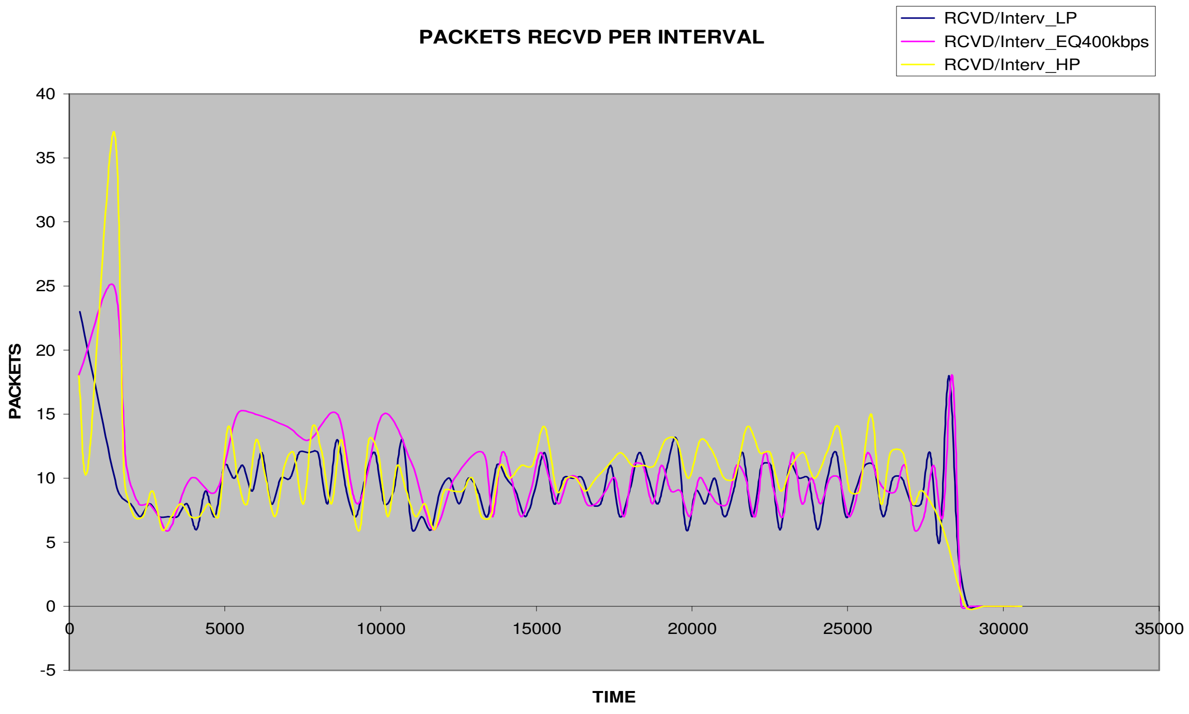
c) Packets Lost Per Interval



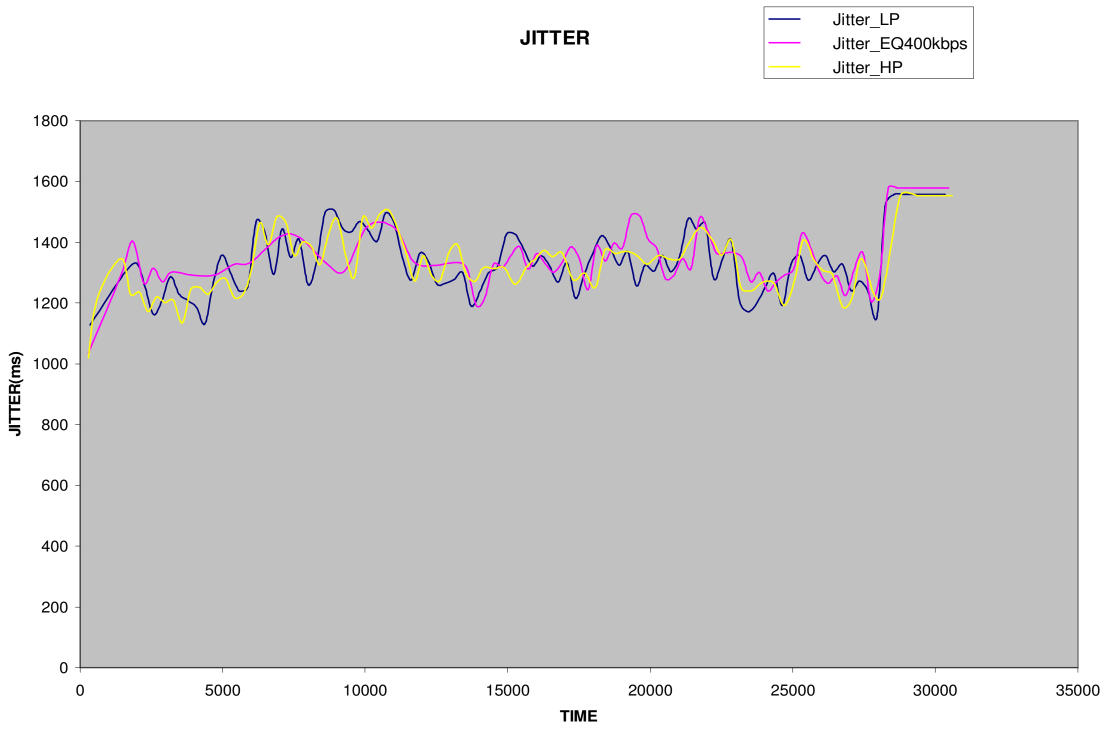
d) Cumulative Packets Received



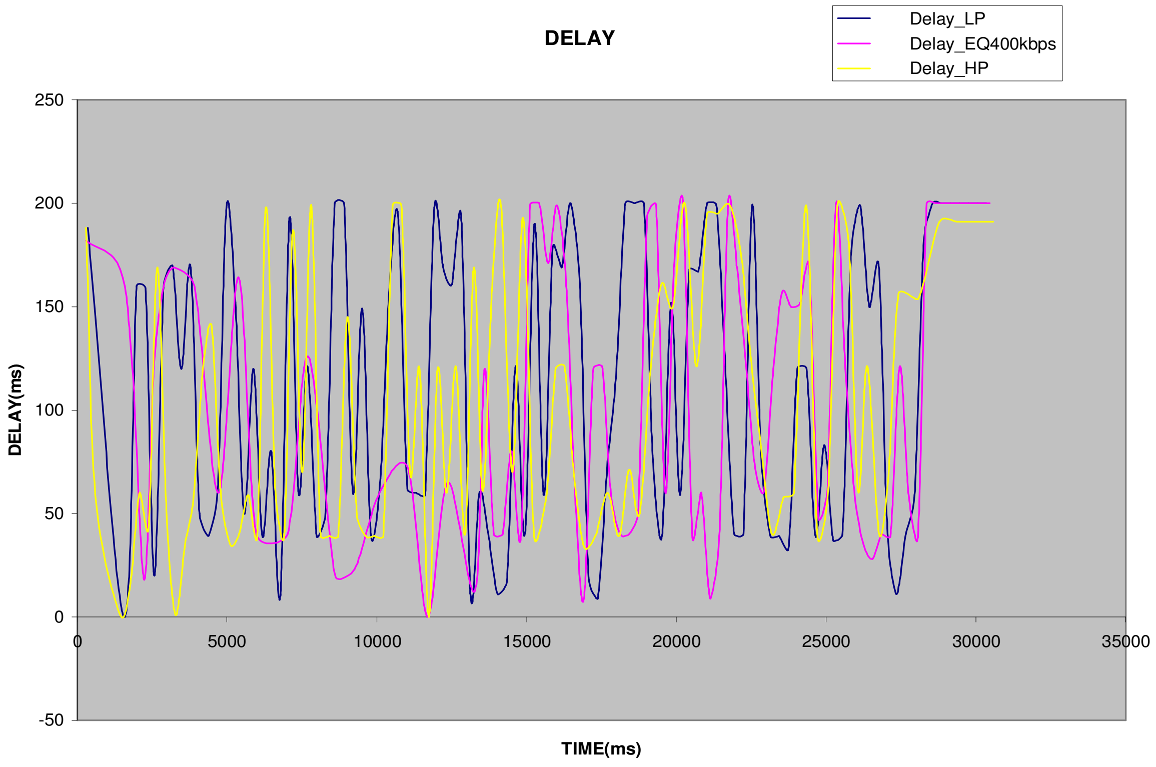
e) Packets Received Per Interval



f) Jitter



g) Delay



A3. Frame Rate Statistics

a) Effect Of Load on the Frame Rate: (With equal priority)

LOAD	Frame Rate
No Load	23.350254
400 Kbps	8.203800
700 Kbps	3.310573

b) Effect of priorities of threads on the Frame rate:

Priority	Frame Rate
Equal	8.203800
RTP High	3.031834
RTP Low (Background High)	6.564551