Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 06, June-2018

De-Jitter Control Methods in Ad-Hoc Networks

[1] Peace Muyambo
[1] Ph D Student,
Department of Computer Science ,
University of Zimbabwe,
Zimbabwe

***Abstract* - In a bid to transfer either voice or video or some other application requiring real-time delivery over a packet network, we need a de-jitter buffer to eliminate delay jitters. A paramount design parameter is the depth of the de-jitter buffer this is because it determines two extremely important parameters controlling voice quality, namely voice-path delay and packet loss probability. In this paper, the author propose and scrutinize several schemes for optimally and dynamically adjusting the depth of the de-jitter buffer. Incrementing to de-jitter-buffer depth adjustments within an audio or video, the initial value and rates of changes of the de-jitter buffer depth are allowed to depend on the class of the audio or video and are adaptively adjusted (upwards or downwards) for every new file (audio or video) based on voice-path delay and packet loss probability measurements over at least one previous file transfer or call to be precise.**

Key words: De-jitter , Packet Loss , Delay Jitters.

INTRODUCTION

A major challenge rather draw back in transporting voice, video or more generally any application requiring real-time delivery over a packet network (using IP, ATM or some other packet-based protocol), is dealing with the delay jitter introduced by the packet network. In all cases real-time presentations cannot tolerate delay jitter, a de-jitter buffer needs to be used to eliminate it. In this paper, the author will mainly consider audio and formats although some of the work would also apply to a more general real-time delivery of voice calls. An important design parameter is the depth of the de-jitter buffer since it influences two important parameters controlling voice quality, namely end to-end voice-path delay and packet loss probability. The dejitter-buffer depth is defines as the maximum amount of time a packet spends in the de-jitter buffer before being played out or received at the destination. If it is too small, then many packets would miss the play-out deadline at the receiving end thereby increasing the packet loss probability. On the contrary, if it is too large, then the end-to-end voice-path delay would increase [1].

The key major challenge is to choose a de-jitter-buffer depth that acts optimally as a middle ground between too much packet loss and too much voice-path delay. A second aspect is static versus adaptive adjustment algorithms of play-out instant. In a static scheme, the play-out instant is set once and for all at the arrival of the first packet of the audio or video file format. In an adaptive scheme, the play-out instant may be shifted during the file format transfer based on the arrival instants of already transferred packets and thereby can improve the delay or packet loss behavior [1].

For this reason, it may be preferable if not suitable to use a static approach in some cases since it truly eradicated the delay jitter. A compromise between a static and an adaptive approach is to adaptively compute an ideal play-out instant with the arrival of every packet but use a static play-out instant (thereby avoiding delay jitters) for most of the different types of file transfers in real time. The static playout instant is synchronised to the adaptive ideal play-out instant at a few selected points in the call thereby limiting the impact on voice-call quality. Specifically when dealing with voice calls involving activity detection and silence suppression, the ideal synchronization points located at the very most beginning instants of every talk-spurt since that only involves shrinking or expanding the previous silence period slightly and has practically no impact on voice quality.

A jitter buffer temporarily stores arriving packets in order to minimize delay variations. If packets arrive too late then they are discarded. A jitter buffer may be mis-configured and be either too large or too small. If a jitter buffer is too small then an excessive and significant number of packets may be discarded, which can lead to tremendous degradation of quality of either video or audio formats. If a jitter buffer is too large then extra delay can lead to conversational difficulty.

Justification
This section seeks to answer the question: Why an Improved Adaptive algorithm as a solution for the De-jitter Control Methods

A typical jitter buffer configuration is 30mS to 50mS in size. In the case of an adaptive jitter buffer then the maximum size may be set to 100-200mS. Note that if the jitter buffer size exceeds 100mS then the additional delay introduced can lead to conversational difficulty.[2]

A de-jitterizer is a device that reduces jitter in a digital signal. A dejitterizer usually consists of an elastic buffer in which the signal is temporarily stored and then retransmitted at a rate based on the average rate of the incoming signal. A de-jitterizer is usually ineffective in dealing with low-frequency jitter, such as waiting-time jitter. [4]

In the Static Algorithm The end-to-end delay Di is set to a constant D for all packets. The only thing we can choose is the de-jitter-buffer delay, D1buf; for the first packet. After the play-out of the first or initial packet, each successive packet is played out strictly periodically unless of course if the packet is to be dropped for late or early arrivals.[1]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 06, June-2018

In the Adaptive Algorithm, the end-to-end delay Di (for i >1) is allowed to adapt based on the observed delays of previous packets unlike the static scheme where Di = D1 for all i). It is assumed that at the instant, a play-out decision is wanted to be made for the play-out of the ith packet, the arrival instants of all previous packets are known. [1]

In the Adaptive with Learning Algorithm we allow the scheme to learn from previous file formats either video or audio of the same type and accordingly adjust its parameters, we allow a simple learning in which the initial de-jitter-buffer depth is set to what was observed at the end of the last transfer file format, the second and subsequent calls or file formats show much better delay performance. [1]

The area of delay adaptation for packet voice and video has been receiving attention for many years. Early experimental systems for packetized voice on a LAN used a fixed scheme for managing the play out buffer. Subsequently, reactive approaches have become popular in the Internet, such as that employed in the vat audio tool and the related improvements proposed in. The recent general adaptation work has also appeared both for application design and middleware support for cooperative QoS management Synchronization between multiple related streams has also been a popular research topic. Some solutions assume that reasonable upper bounds on the delay across streams are available, while others set a fixed upper delay bound for a stream, discarding data (packet loss) which arrives late and would cause a loss of synchronization.

The rise of jitter and de-jitter

Jitter Defined

In contrast to the constant algorithmic and processing delay, transmission delay varies over time. The reason is that the transit time of a packet through an IP network will vary due to queuing effects. The transmission delay is divided into two parts, first being the constant or slowly varying network delay and the second one being the rapid variations on top of the basic network delay, usually referred to as jitter. The jitter is defined as a smoothed function of the delay differences between consecutive packets over time. Jitter can also be precisely defined as a free variation in packet delays which inhibits transfer of voice and video of real time delivery in packet switched network. In this paper Real Time Protocol will be abbreviated to RTP

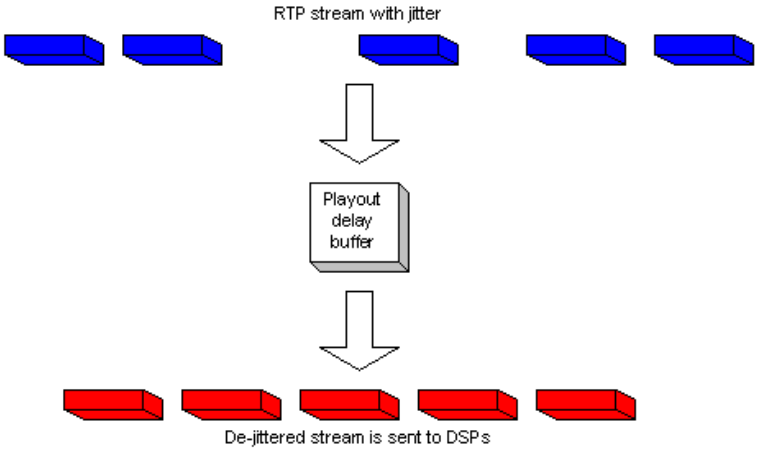

Fig 1 RTP stream with jitter

If the jitter is so large that it causes packets to be received out of the range of the buffer specified, the out-of-range packets are discarded and lost and dropouts can be heard in the audio and seen in cases of video. For dropouts as small as one packet, the Digital Signal Processors (DSP) interpolates what it thinks the audio should be and no problem is audible. When jitter exceeds what the DSP can do to make up for the missing packets, audio problems are heard.

This diagram illustrates how excessive jitter is handled

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 06, June-2018

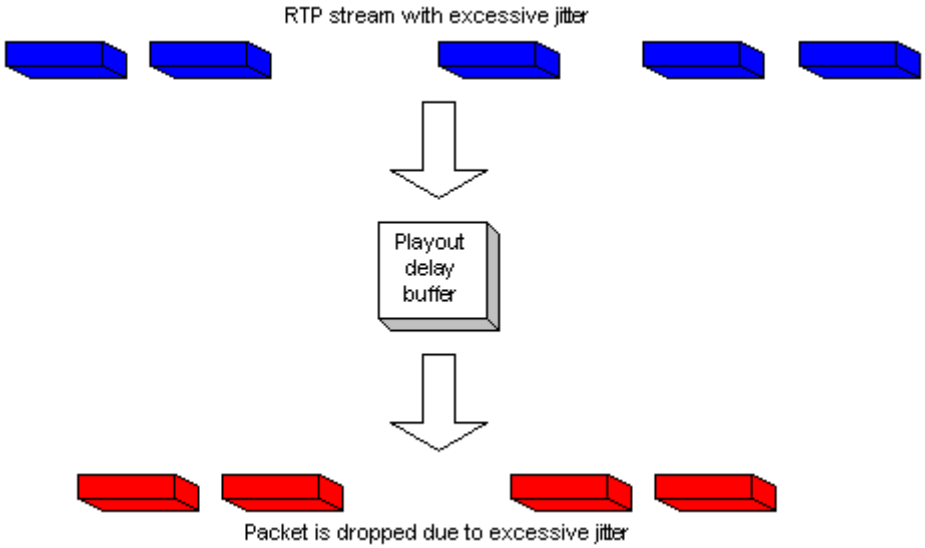

Fig 2 RTP with excessive jitter

What Causes Jitter?
Jitter is generally caused by congestion in the Internet Protocol network. Either the congestion can occur at the interfaces of router, which acts as a gateway device or in a provider or carrier network if the circuit has not been provisioned the proper way.

The jitter present in packet networks complicates the decoding process in the receiver device because the decoder needs to have packets of data readily available at the accurately correct time instants. If the data is not available, the decoder will not be able to produce smooth, continuous speech or continuous video stream. Thus, in addition to adding to the delay, jitter leads in a timing problem for the receiver. A jitter buffer is normally used to make sure that packets are available when needed

Packet loss

Packet loss takes place either if a router in the network drops a packet or if a packet arrives too late to be handled by the decoder. By allowing for a long delay in the jitter buffer, the latter type of packet loss can be almost completely removed, but at the price of increased system delay.

When packet loss occurs due to either a packet being dropped somewhere in the network or arriving too late, some mechanism for filling in the missing speech must be incorporated. Such solutions are usually referred to as error concealment algorithms or packet loss concealment (PLC) algorithms. For best performance, these algorithms have to accurately predict the speech signal and make a smooth transition between the previous decoded speech and inserted segment.

Since packet loss occurs mainly when the network is heavily loaded and congested, it is not uncommon for packet losses to appear in bursts. A burst usually consists of a series of consecutive lost packets or a period of high packet loss rate. It is however common, when several consecutive packets are lost, even the best algorithm will have problems producing acceptable audio or speech quality and high video quality.[5]

To improve useful utilization of bandwidth, multiple audio and video frames are in most cases carried in a single packet, so a single lost packet may result in multiple lost frames. Though packet losses occur randomly, the listening experience is then similar to that of having the packet losses occur in bursts.

End-Point Issues
The choice of audio or video codec has a major effect on speech or video quality. There are several other, not so obvious; problems related to implementing speech-processing functions in the actual end-point devices that can have significant impact on the quality of both file formats. Examples include the de-jitter buffer emulator, the analog-to-digital converter, voice or video activity detection, echo cancellation algorithm, among other voice and video processing issues.

Whether the communication end-points are gateways or other devices, low-frequency clock drift between the two can cause receiver buffer overflow or underflow in either cause which leads to packet loss. In simple cases, speaking this effect can be described as the two devices talking to each other having different time references. For instance, the transmitter might send packets every 20 ms according to its perception of time, while the receiver's perception is that the packets arrive every 20.5 ms, having a delay of 0.5ms.[5]

Typical Solutions
The designer of a Voice over Packet (VoP) gateway solution has to implement efficient and effective solutions to the jitter, packet loss, and clock drift problems. In addition, every possible aspect of delay has to be taken into consideration. [5]

What is Jitter Buffer
A jitter buffer is required to make sure that packets are available when needed for play-out. It removes the jitter in the arrival times of the packets at the cost of an increase in

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 06, June-2018

the overall delay. The objective of a jitter buffer algorithm is to keep the buffering delay as short as possible while minimizing the number of packets that arrive too late to be used. A large jitter buffer causes an increase in the delay and decreases the packet loss. A small jitter buffer decreases the delay but increases the packet loss.[1]

The ordinary traditional approach is to store the incoming packets in a buffer (packet buffer) before sending them to the decoder. Since packets can arrive out of order, the jitter buffer is not a strict first-in-first-out (FIFO) buffer, but also reorders packets if necessary. The most straightforward approach is to have a buffer with a fixed number of packets. This result in a constant system delays and requires no computations and provides minimum complexity. The major disadvantage with this approach is that the length of the buffer has to be made sufficiently large that even the worst case can be accommodated.[5]

In order to keep the delay as short as possible, it is important that the jitter buffer algorithm adapt rapidly to changing network conditions. Therefore, jitter buffers with dynamic size allocation, so-called adaptive jitter buffers, are now most common. Inserting packets in the buffer when the delay needs to be increased, and removing packets when the delay can be decreased achieve the adaptation. Packet insertion is usually done by repeating the previous packet. Unfortunately, this will almost always result in audible distortion, so most adaptive jitter buffer algorithms are very cautious when it comes to reducing the delay. [1]

This traditional packet buffer approach is limited in its adaptation granularity by the packet size, since it can only change the buffer length by adding or discarding one or several packets.

Another major limitation of traditional jitter buffers is that, in order to limit the audible distortion of removing packets, they can only function during periods of silence or video streaming. Hence, delay builds up during a talk spurt, and it can take several seconds before a reduction in the delay can be achieved. Also, high delay at the end of a talk spurt will have a severe effect on the conversation because it increases the probability of the speakers stepping on each other's talk. [5]

Packet Loss Concealment

Only recently, two simple approaches to deal with lost packets have prevailed. The first method, referred to as zero stuffing (ZS), involves simply replacing a lost packet with a period of silence of the same duration as the dropped or packet. [5]

The second method, referred to as packet repetition (PR), assumes that the difference between two consecutive speech frames is quite small and replaces the lost packet by simply repeating the previous packet. It is important to note that this is not the case in practice, since even a minor change in pitch frequency, is easily detected by the human ear. Subsequently, it is virtually impossible to achieve smooth transitions between the packets with this approach. [5]

De-Jitter Buffer Operation

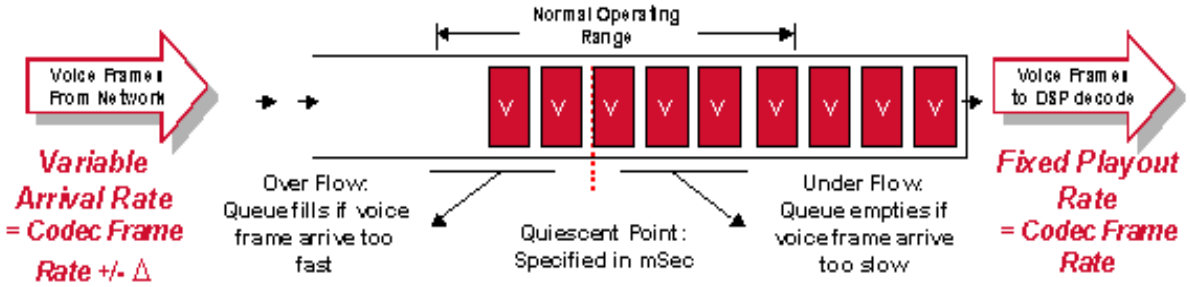

Fig 3 de-jitter buffer operation [6]

The de-jitter buffers can be adaptive, but the maximum delay is fixed. When adaptive buffers are configured, the delay becomes a variable figure. In this case, the maximum delay can be used as a worst case for design purposes. Additionally, the optimum initial play out delay for the de-jitter buffer is equal to the total variable delay along the connection. It is strictly important to handle properly the de-jitter buffer. If packets are held for too short a time, variations in delay can potentially cause the buffer to under-run and cause gaps in the audio or video. If the packets are held for too long a time, the buffer can overrun, and the dropped packets again cause gaps in the audio or video. Lastly, if packets are held for too long a time, the overall delay on the connection can rise to unacceptable levels, which lead to serious compromise to both audio and video quality. [6]

De-jitter-buffer algorithms

The successive voice or video packets are transmitted strictly periodically with a time interval equal to the packetization interval. Let Di^{net}, Di^{buf} and Di represent the network delay, de-jitter-buffer delay and end-to-end delay, respectively, experienced by the i^{th} packet. All delays are unidirectional. Additionally, throughout this paper the author will use seconds as the unit of time. Clearly, $Di = Di^{net} + Di^{buf}$ .In most cases it is not possible to accurately estimate the one way network delay without requiring elaborate synchronization procedure between the transmitter and the receiver.[1]

We assume that no such procedure is available, but we can accurately obtain the relative network delay among packets as explained below. Specifically, if dp represents the packetization delay and t_i; t_j represent the arrival instants at

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 06, June-2018

the de-jitter buffer of packets i and j; respectively then due to the strict periodic nature of packet transmission, we get

$D_j^{net} - D_i^{net} = t_j - t_i - (j- i)\, dp$ may be obtained accurately at the receiver without requiring any synchronization with transmitter. For any de-jitter algorithms, we only need relative delays among packets except in one case where we need an approximate estimate of the absolute one-way delay in order to slowly adapt upwards the arrival instant of the minimum-delay packet following an internet route change that increases the end-to-end propagation delay. If a synchronization procedure exists between transmitter and receiver in order to accurately estimate the absolute one way delay then that may be used. If not then an estimate may be obtained by taking the minimum of several round-trip delay measurements and dividing it by two. [1]

Static algorithm

The end-to-end delay Di is set to a constant D for all packets. The only thing optional to choose is the de-jitter-buffer delay, $D1^{buf}$, for the first packet. After the play-out of the first packet, each successive packet is played out strictly periodically unless of course the packet is to be dropped for late or early arrivals .This implies that the end-to-end delay of every packet (that is not dropped) is given by $Di = D = D1 = D1^{net} + D1^{buf}$. For the i^{th} packet if $Di^{net} = D$ then it is a late packet and is dropped. The amount of time the i^{th} packet stays in the de-jitter buffer (provided it is not late) is $Di^{buf} = D = Di^{net}$: If this time is too long or large, then the physical holding capacity of the buffer may be exceeded requiring the packet to be dropped for being too early. The assumption is that physical holding capacity of the buffer is large enough so that no packet is dropped for early arrival. In Figs below shows how the static algorithm as a function of the only tunable parameter, $D1^{buf}$ the de-jitter-buffer depth set for the first packet. The de-jitter-buffer depth is in sec. In all cases, however, the de-jitter-buffer depth in number of packets may be obtained by dividing this quantity by the packetization interval. Shows that the same de-jitter buffer depth may give significantly different delay and packet loss probability depending on the type of stream. [1]

Adaptive algorithm

In this algorithm, the end-to-end delay Di (for i > 1) is allowed to adapt based on the observed delays of previous packets as opposed to the static scheme where Di = D1 for all i. The assumption is at the very instant, a play-out decision is needed to be made for the play-out of the i^{th} packet, the arrival instants of all previous packets are known however if some previous packet has not shown up yet, and its delay is thus assumed infinity. Such a packet will be a late packet and will be dropped any way once it shows up. In most cases if not all times, we estimate a minimum-delay packet and a buffer depth which is defined as the delay experienced by the minimum-delay packet in the De-jitter buffer. For i >1; let Di^{min} represent the network delay of the minimum delay packet when play-out decision is made for the i^{th} packet.

Then $Di^{min} = Min_j \{Dj^{net}\}_j = 1\ldots i\text{-}1$. In order to slowly adapt the minimum-delay packet upwards in case there is a constant upward shift in network delay in most cases due to change in network route caused by a failure we change the minimum-delay calculation slightly. [1]

Methodology

The approach to this study is through an experimental setup which involves the design and implementation of the De-jitter Control Methods in computer networks. Data analysis components, developed in this thesis, are all done and developed under pure simulation. Each of the following sections describes the practical aspects of the developed analysis components.

This recommendation defines a set of performance parameters for packet networks and end terminals that can assist in quantifying the end-to-end quality of speech and other voice band applications. It is largely focused on quality impairments resulting from delay variation and packet loss which are peculiar to IP and other packet-based technologies, and which do not appear in traditional networks. It discusses the interactions and trade-offs among these packet impairments, and describes mechanisms such as de-jitter buffers and packet loss concealment for reducing their effects on the quality of speech and other applications specifically in this paper audio and video. [7]

Packet Loss processing in a real Packet Switched Network

The following diagram is process through which IP packet parameters/impairments, which are transfer delay, delay variation, can be mapped to application layer performance in terms of overall loss and delay.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 06, June-2018

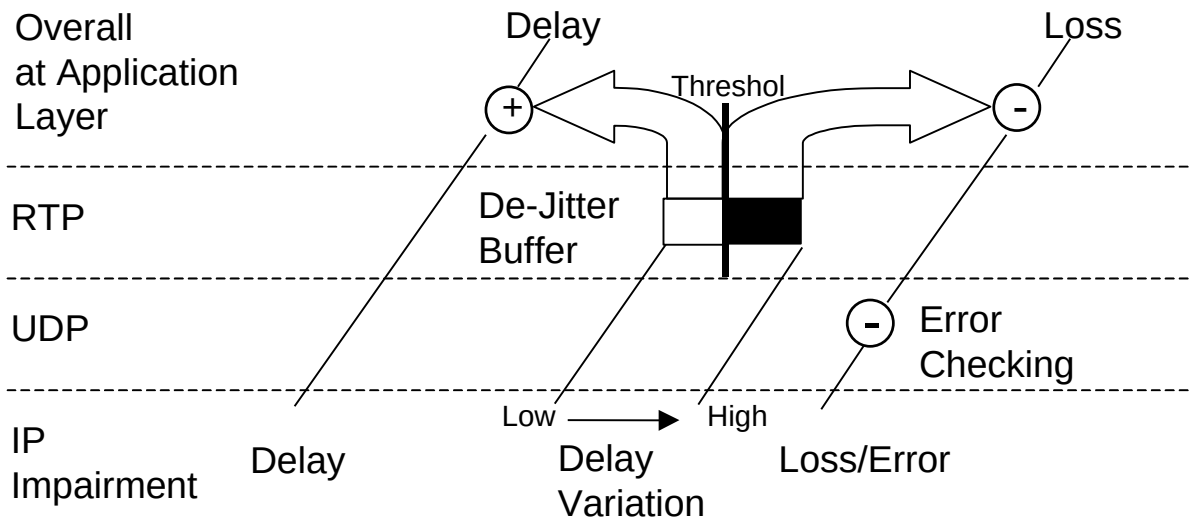

Fig 4 Mapping Internet Protocol Packet to the Application Layer [7]

At the very bottom of the figure, packets arrive with different impairments due to the source terminal and network, or they are lost. Packets are serviced accordingly in the order of their arrival as they move up the protocol stack to remove as much impairment as possible. Packets with delay variation in the "acceptable" range are accommodated, while packets with larger variation (in the "unacceptable" range) would be discarded, however it is imperative to note that the range of acceptability or unacceptability is left for the user to define.

A significantly large

De-jitter buffer can accommodate packets with greater delay variation; this means that fewer packets would be lost overall at the expense of larger overall delay. On a different note , a smaller de-jitter buffer will produce less overall delay, but expose a larger fraction of packets to be discarded by the terminal and increase the overall loss.[7]

Methods of Construction of De-jitter Buffers

Two main types of de-jitter buffers exist, fixed (static) length and adaptive length. De-jitter buffers can be constructed in many different ways, including the following attributes identified in Table below. [8]

Methods of coming up wth de-jitter parameters

Type	Attributes	Possibilities	
Fixed (and Adaptive)	Size (configure maximum and nominal or minimum)	Integer number of packets	Fractional number of packets
Adaptive	Control	Timed decay if no over/under flow	Evaluate Loss Ratio (configure lowest acceptable threshold, and minimum packet count between adjustments)
	Adjustment	Timed	Silence Gaps only
	Initialization	First Packet	Small sample
	Adjustment Granularity	Packet size	Fraction of packet size
	Restores packet order	Yes	No
	Voice band data mode	Detect 2100 Hz tone, set to maximum length	None

Fig 5 the measurement of de-jitter buffers and parameters. [8]

What is Packet Loss?

The type of de-jitter buffer there will always depend on some criteria for determining whether each specific packet in a flow is accommodated or discarded. The result can completely change the distribution of overall packet losses. Suppose we assume that random bit errors are causing packets to fail the UDP checksum, and then packet losses will have a random distribution as they proceed up in all cases to the application layer. Instead if several consecutive packets experience excessive delays, then the additional discards due to the limitations of the de-jitter buffer will make the overall loss distribution appear busty rather than random. [8]

What is Packet Delay?

The average network delay should be summed with the average de-jitter buffer occupation time and other delays to obtain an overall average delay. This criterion allows for buffer adaptation, needing only the average queuing time for

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 06, June-2018

all packets in the assessment time interval. On the contrary, if only the minimum network delay is assumed or known, it should be added with the maximum de-jitter buffer occupation to give an overall delay. In this regard initial fixed size de-jitter buffer, if the first packet to arrive has the minimum transfer delay, then the receiving end will buffer the packet for the entire period requested, and buffering size will be as anticipated. It's fortunate however that, many packets arrive at or near the minimum transfer time, so this case has a fair likelihood. On the contrary, if the first packet has a rather prolonged delay, then more buffer space will be needed to accommodate the "early" arrival of packets at or near the minimum transfer time, and the de-jitter buffer will contribute more than the expected delay to the overall calculation.[8]

Comparison of the Fixed and Adaptive De-jitter Buffers

In simple terms, the effectiveness of the model of loss due to a fixed de-jitter buffer is to designate as discarded all packets whose delay is greater than the minimum transfer delay for the packet stream plus the (fixed) de-jitter buffer length.

During the calculation of the overall impairment contribution of a fixed de-jitter buffer; the distribution of delay variation determines the proportion of packets that would be lost. The distribution of packet delays that are accommodated can be used to calculate the average de-jitter buffer occupation delay, as follows:

Average delay Occupation = [De-jitter Buffer Size] - (Average Delay of Accom. Packets - Average. Delay)

This average/mean delay can be summed with other destination terminal delay constants to give an output that estimates the average destination terminal delay. If the exact delay distribution is not reachable, then there is consensus that a value of half the de-jitter buffer size may be substituted for the average occupation delay in calculations supporting network planning. [8]

In the Adaptive scheme the time series of packet arrivals may be used with an adaptive de-jitter buffer emulator to determine the buffer size dynamics and the average de-jitter buffer occupation time (delay) over the series. This average delay can be summed with other destination terminal delay constants to produce an estimate of the average destination terminal delay. [8]

Standard Calculation of Overall delay

Incrementing the analysis of the de-jitter buffer, the following formulas are acceptable and correct, and their use is determined by the specific delay statistics available for computation. When the mean delays for all components are at hand:

Overall-Average-Delay = average-source-delay + average-net-delay + average-destination-delay)

It is imperative that when the minimum delays for source terminal and network can be added with the maximum destination terminal delay to obtain an estimate using constants which leads to the formulae of calculating the Overall network delay as follows:

Overall Delay (constant) = min (source-delay) + min (net-delay) + max (destination-delay)

It is important to note that the author did not necessary use the above formulas to determine overall delay since in most cases when measured directly, the Overall Delay is the interval defined from the time that a signal enters the Mouth Reference Point and ending at the time when the corresponding signal exits the Ear Reference Point or equivalents referent points depending on the type of application where is simple terms is the time from source plus network plus destination.[8]

Possible Errors in Parameter Calculation

A simplified style or approach to obtain the overall end-to-end delay has been to use the average IP packet transfer delay, and sum it up with constants for other elements in the sender to receiver path. This criterion may output errors because of non constant delays in some terminal components or because the variable delays elements are ignored. Seconding this pitfall would be to use the packet loss ratio as measured by a physical device called a test receiver that allows, for example, 3 seconds before declaring a packet lost, thereby underestimating the actual accurate loss ratio. Conclusively, typical de-jitter buffer would have much less tolerance for long delays beyond the norm.

Improved Adaptive De-jitter Buffer Algorithm

Due to the draw backs of both the Static(fixed) and Adaptive Algorithms stated in the literature review the author designed the below algorithm in a bid to improve what has already been done and this algorithm has been adapted as the methodology in determining the results in the following chapter.

Algorithm Research design

- The researcher have designed an Improved Adaptive Algorithm whose depth depends on traffic type(audio /video)
- The Depth of buffer is based on both observed delays of previous packets and the size of packets
- Send packets according to priority specified by user
- User can specify the size of file and the algorithm splits the file into packets sent to the buffer
- Optimum depth buffer which avoids both packet loss probability(too small) and end path delay increase (too big)

Algorithm:

Select the type of file format to be downloaded

Either video or audio

1. *If audio ,then choose the audio type written in order of service priority*

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 06, June-2018

Mp3 , Mp4, Mid (where priority is a configurable parameter)

Else if

2. *Video , then choose the video type written in order of service priority*

3gp , Asf , Avi (where priority is a configurable parameter)

Upon selection of a file format number, then specify size to be downloaded in MB

Else if go back to main menu and repeat process 1 and 2 or exit the program

3. *When size is specified in MB, divide size into number of packets= (vid/audfile.megasize/temp)*

Where temp is a float variable and vid/aud.megasize is a function that depends on network traffic and availability of path packets travel to destination.

If 3 above is satisfied allocate buffer depth (configurable parameter)

4. *For buffer depth allocated to each packet > T1 a threshold for audio or video, discard packet and calculate time taken to download file (a difference of start and end time) and also calculate network delay based on network congestion.*

5. *While buffer depth allocate to each packet < T2 a threshold for audio or video ,*

Do accept / service packet and calculate time taken to download file (a difference of start and end time) and also calculate network delay based on network congestion.

Finally, go back to main menu and repeat steps 1 and 2

Else exit the program.

RESULTS AND ANALYSIS

In this section of study, the Improved Adaptive Algorithm is used to test its efficiency and efficacy if any as compared to the standard which is the Adaptive Algorithm. This study is based on an actual data input by the author (simulations) though the size of the audio and video file formats has been 5MB and 10MB, this was meant to simplify and for speed the analysis. The results and implementations of the run simulations are used to illustrate the advantage of using the author's proposed algorithm.

File type(5mb)	Total packets split	Download time	Total packets dropped	Delay in sec
Asf	9	6min 30sec	2	2.0sec
Avi	5	4min 0sec	0	0.1sec
Mid	5	2min 0sec	1	1.0sec
Mp3	16	2min 7sec	4	6.0sec
Mp4	8	2min 2sec	6	8.0sec

Fig 5. Table of results with for 5MB

File type(10mb)	Total packets split	Download time	Total packets dropped	Delay in sec
Asf	11	8min 6sec	2	2.0sec
Avi	7	5min 7sec	0	0.0sec
Mid	7	3min 0sec	3	3.0sec
Mp3	23	8min 0sec	4	5.0sec
Mp4	11	3min 0sec	7	7.0sec

Fig 5. Table of results with for 10MB

The fewer the packets on the network the less the packets are dropped because they are serviced faster this is the case of Avi and mid on both 5MB and 10MB since in general their rate of packet loss probability is slightly lower than others file formats. Special cases of Avi having an acceptable delay of 0.0 sec where none single packets are dropped at all.

It is also important to note that file formats with relatively large depth buffer no matter the packets are divided approximately the same their rate of dropping packets is acceptable as compared to file formats with low depth. This is the case of Asf and Mp4 in the 5MB the packets are 9 and 8 respectively and in 10MB, packets are 11 and 11, however since the buffer range of Asf is larger than that of Mp4 rate of packet loss and delay is rather accepted in Asf (video file) rather than Mp4 (audio file).

Another point to note from the results is that as a file format is segmented into many small packets in the case of Mp3, delay can increase though however the rate of packet loss can remain lying in the acceptable range, this is because by taking many path (packet switched network), time can be lost (increase in delay) as many packets can reach at the decoder and wait for the "late" ones for purposes of re-order and assembly.

Finally, priority plays an important role since no matter the existence of good advantages a file format has; it has to wait for the one with higher priority to be serviced first.

Evaluation of Improved Adaptive Algorithm

The following is an abstract the author took as a general comparison of the proposed Improved Adaptive algorithm and the Adaptive algorithm using an Mp3 audio file format to

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 06, June-2018

test the validity of the author's algorithm is the graph below is a fair and true representation of the performance of the two algorithms. It is important to note that in general the author contributed something to the De-jitter Algorithms.

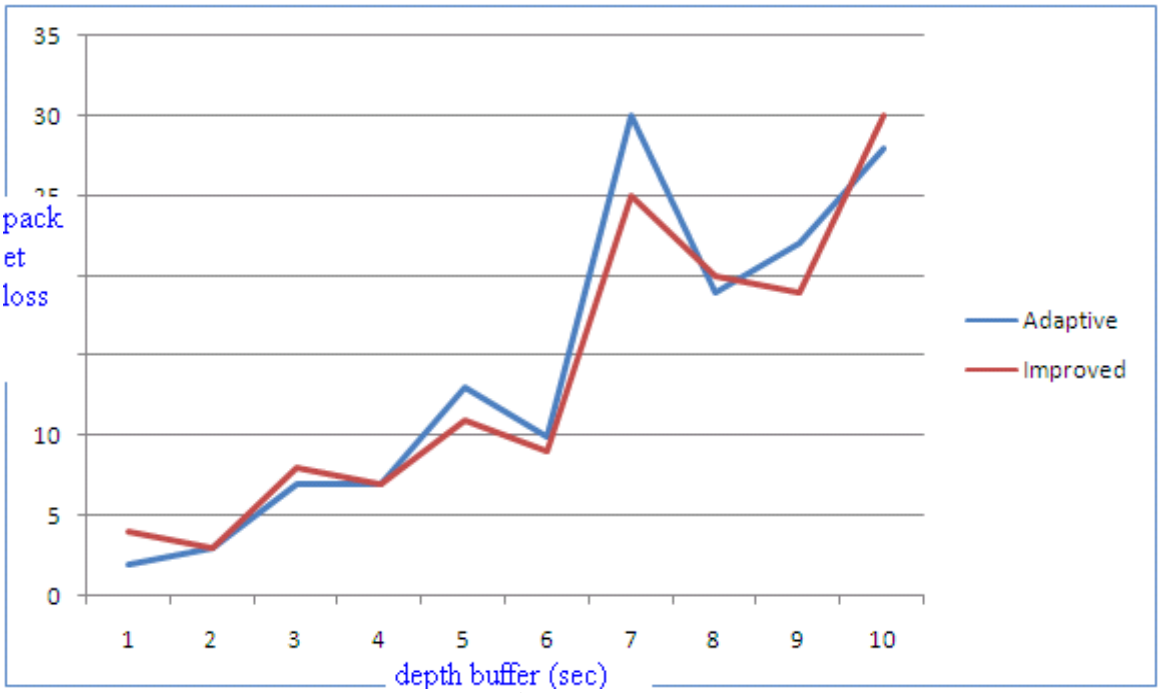

Fig 6. Comparison of Improved and Adaptive Algorithm

SUMMARY AND CONCLUSION

In this final section of study, the author among other contributions and observations the following are the main contributions that can be presented in this thesis. In order to achieve good speech quality invoice over Packet (VoP) gateways, issues related to typical packet network properties, such as delay, jitter, and packet loss, have to be considered. More specifically, a good design has to include:

- The fewer the packets on the network the less the packets are dropped because they are serviced faster
- Network performance places a major if not significant role in determining the effectives of any of the three de-jitter algorithms which were under study
- The faster the network the faster and better adaptivness of the de-jitter buffer depth and allocation of packets , hence better performance presented
- jitter buffer has no general good recipe since its effectiveness is solely hinged and very much dependent on the target application and the environment where the application is used since in most if not all the cases it is not necessary to have a complex and/or rather memory consuming jitter buffer implementation if the delay variation for the target network is very low
- Efficient jitter buffer algorithms that can deal with network jitter and clock drift.
- An efficient packet loss concealment solution residing on the DSP to achieve acceptable quality
- A jitter buffer implementation is dependent on the target application resources and the ecosystem where the application is deployed. In general, the better the algorithm estimating the play out time for a packet the more resources are needed.
- A simpler algorithm may be used for a network with low delay variation and packet reordering or at least a network with a predictable behavior. However, the applications are not deployed to a single system, and different systems may have different specifications in terms of network behavior.

REFERENCES

[1] Gagan L. Choudhury, Robert G. Cole "Design and analysis of optimal adaptive de-jitter buffers" (2003) 1-12
[2] ITU-T G107 Recommendation, "The E-model, a computational model for use in transmission planning" (2004)
[3] R. Ramjee, J. Kurose, D. Towsley, H.Schulzrinne, "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks
[4] http://en.wikipedia.org./wiki/jitter/dejitterizer
[5] GIPS NetEQ™ - A Combined Jitter Buffer Control/Error Concealment Algorithm for VoP Gateways (2002) 2-17
[6] http://www.cisco.com/playout delay enhancement for VoIP
[7] Altman, E., Avrachenkov, K., Barakat, C., TCP in the Presence of Bursty Losses, Performance Evaluation 42 (2[2] Berger J. M., Mandelbrot B. A New Model for Error Clustering in Telephone Circuits. IBM J R&D July 1963000) 129-147

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 06, June-2018

[8] Draft new Recommendation G.1020- Performance Parameter Definitions for Quality of Speech and other Voice band Applications Utilising IP Networks (2003)9 -21

[9] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, Adaptive playout mechanisms for packetized audio applications in wide-area networks, Proceedings of the IEEE Infocom, Totonto, Canada, June 1994, pp. 680–688.

[10] Miroslaw Narbutt, Mark Davis, "Assessing the quality of VoIP transmission affected by playout buffer scheme"2000 pp17-21