ENHANCING THE QOS OF A VOIP CALL USING AN ADAPTIVE JITTER BUFFER PLAYOUT ALGORITHM WITH VARIABLE WINDOW SIZE

Atri Mukhopadhyay[1], Tamal Chakraborty[2], Iti Saha Misra[2], Salil Kumar Sanyal[2]

[1]School of Mobile Computing and Communication, Jadavpur University Salt Lake Campus, Kolkata-700098, India
[2]Dept. of Electronics & Telecommunication Engineering, Jadavpur University, Kolkata-700032, India
atri.mukherji11@gmail.com, tamalchakraborty29@gmail.com, iti@etce.jdvu.ac.in, s_sanyal@ieee.org

ABSTRACT

Transmitting real-time voice over the Internet is a technological challenge. Variation in network characteristics introduces jitter to the propagating voice packets. Jitter hampers voice quality and makes the VoIP call uncomfortable to the user. Often buffers are used to store the received packets for a short time before playing them at equal spaced intervals to minimize jitter. Choosing optimum buffering time is essential for reducing the added end-to-end delay and number of discarded packets. In this paper, some established adaptive jitter buffer playout algorithms have been studied and a new algorithm has been proposed. The network used for analyzing the algorithms has been simulated using OPNET modeler 14.5.A. Further studies have been conducted for finding the optimum sliding window size for the proposed algorithm. The proposed algorithm kept jitter within a tolerable limit along with significant reduction of delay and loss compared to other algorithms analyzed in this paper.

KEYWORDS

VoIP, QoS parameters, Adaptive Algorithm, Congested Network, Jitter Buffer

1.INTRODUCTION

The transmission of real-time audio and video applications over the Internet is a challenging task. These applications have strict bounds in terms of delay and loss. This set of applications is called Real-time applications and they include facilities like Internet Protocol (IP) telephony, teleconference, etc. Of the various real-time applications, we have concentrated on Voice over IP (VoIP) since it has gained importance over the past few years owing to its low cost and ease of interfacing between data and voice traffic [1]. VoIP is a technology that transmits voice using the underlying Internet backbone. In order to provide agreeable Quality of Service (QoS), a VoIP service provider must strive to meet the stringent time requirements of VoIP. However, the characteristics of the Internet backbone are time-variant in nature. Its properties are so random and unpredictable that it is not an easy task to statistically determine the way the backbone is going to behave in a future point of time. The reason behind the lack of proper prediction of its characteristics is its dependency on the behavior of the other connections throughout the network

DOI : 10.5121/ijmnct.2012.2306

[2]. The connectivity may be hampered due to several reasons rendering networking applications ineffectual. The networks suffer from congestion when traffics exceeding the capacity of the network are routed through it. As a result, the data packets suffer high delay and loss while passing through the network. Such delay and loss are unacceptable when we want to utilize the network for transmitting voice traffic.

The unpredictability of the network characteristics may also cause the consecutive packets passing through the network to suffer from different extents of end-to-end delay. Generally, the VoIP applications send data at a constant rate. So, any alteration in the end-to-end delay suffered by two voice packets means that the time between two events occurring at the source and the time between the events perceived at the receiver are not equal. Such an event is not desired in a VoIP system as it degrades speech quality. This variation in delay suffered by two consecutive voice packets is termed as inter-arrival jitter [1], [3]. VoIP applications are not only sensitive to the extent of the delay and loss suffered by the voice packets but also to the inter-arrival jitter. Jitter is one of the main factors that degrade the QoS in IP networks [4]. Hence, we need some mechanisms to undo this variation in delay that is being incorporated into the voice packets by the network.

One of the feasible solutions is the use of some mechanisms which aim to reduce the network congestion, e.g. increasing the packet payload size [5]. A set of standard voice Coder-decoders (codecs) of different bit rates can also be employed. In this method, a lower bit rate codec is used when the network becomes congested so as to reduce the traffic introduced to the network. However, when the network conditions become favorable for transmission, higher bit rate codec is used to achieve better QoS [8]. A mechanism using a Fast-Startup Transmission Control Protocol (TCP) has been explored in [6] in order to find a possible way to enhance VoIP performance in congested networks. Another method that seeks to reduce congestion with the help of distributed routing has been proposed in [7] and [8]. Congestion can also occur in the intermediate access points. So VoIP performance can also be enhanced by optimizing the access point parameters [9].

However, the most effective solution to minimize jitter is to store the voice packets for a short time in the receiver buffer before playing it out, thus reducing the jitter [10]. However, using a fixed playout time for every packet is rendered useless if the network characteristics are variable and the voice packets suffer different extent of delay while passing through it [11]. Several algorithms have already been proposed so that the playout time for a voice packet is delayed in accordance with the variation in the network and thus provide better QoS for the VoIP applications. However, if the playout delay is too large, the end-to-end delay suffered by the voice packets is increased beyond acceptable limits and the VoIP performance may become irritating to the user because significant awkwardness occurs between speakers when delay exceeds 200 ms [11]. On the other hand, if the playout delay is too small, voice packets may be discarded due to late arrival [1]. This leads to loss in information and hampers the voice quality. It is not desirable that the voice packets suffer from either high delay or high loss. So, it is mandatory to obtain an optimum playout time to get the best performance out of a VoIP system.

In this paper, we have analyzed some of the already established adaptive jitter buffer playout algorithms and have tested for their efficiency in several network scenarios. Further, we have also taken a note of their shortcomings and have proposed a new adaptive jitter buffer playout algorithm that provides the optimum QoS to the VoIP application in terms of delay, loss and jitter. The performance of the new algorithm has been tested in varying network scenarios using OPNET Modeler 14.5.A. The effect of varying the sliding window size on the performance of the proposed algorithm has also been tested. Moreover, its performance has also been compared with the analyzed algorithms.

The remainder of the paper is organized in the following manner. Section 2 provides a brief description of VoIP. Section 3 reviews some of the works that have already been conducted on adaptive jitter buffer algorithms. Section 4 describes the simulation setup and the methods that have been employed during the course of this work. Section 5 focuses on a thorough analysis of the works mentioned in Section 3 with the help of above-mentioned simulation setup. Section 6 contains the details of the proposed adaptive jitter buffer algorithm. Section 7 illustrates the results achieved by employing the proposed algorithm, investigates the effect of the window size on the proposed algorithm's performance and also provides a comparative analysis of the proposed algorithms against the analyzed adaptive jitter buffer algorithms. The work is concluded after the extensive analysis in Section 6.

2. BACKGROUND STUDY

The Internet mainly relies on IP for proper routing of the packets passing through it. So it is clear that in the network layer, IP is the life force of a VoIP system. Both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) can be used in the Transport layer. However, owing to its lower bandwidth requirements, UDP is preferred over TCP [1], [3]. However, UDP lacks the reliability provided by TCP. So, another protocol, the Real-time Transport Protocol (RTP), rides on top of UDP and IP to ensure that the voice packets are able to meet their stringent time requirements [12]. RTP provides timestamps and sequence numbers to voice packets. The receiver can use the sequence number to determine whether the packets have arrived in order and the timestamps to assess the inter-arrival jitter suffered by the packets en route to the VoIP receiver. Real-time Transport Control Protocol (RTCP) works in tandem with RTP. It provides control information to the VoIP source and the VoIP receiver, allowing them to work efficiently. The primary function of RTCP is to provide feedback on the quality of data distribution [12].

In a VoIP system, inter-arrival jitter is an estimate of the statistical variance of the voice packet inter-arrival time. The inter-arrival jitter is defined to be the mean deviation of the difference in packet spacing at the receiver as compared to the sender for a pair of packets. The inter arrival jitter is given by (1) [12].

$$J(i) = J(i-1) + \{|D(i-1), i| - J(i-1)\}/16 \qquad (1)$$

where, *J(i)* is the jitter for the i^{th} voice packet and for two packet *i* and *j*, *D(i,j)* is given by (2) [12].

$$D(i, j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i) \qquad (2)$$

where, S_i and S_j are the RTP timestamps for packets *i* and *j*, and R_i and R_j are the time of arrivals in RTP timestamp units for packets *i* and *j* .

It is evident from (2), that the inter-arrival jitter can be easily calculated from the delay suffered by two consecutive voice packets while travelling from the VoIP source to the VoIP receiver.

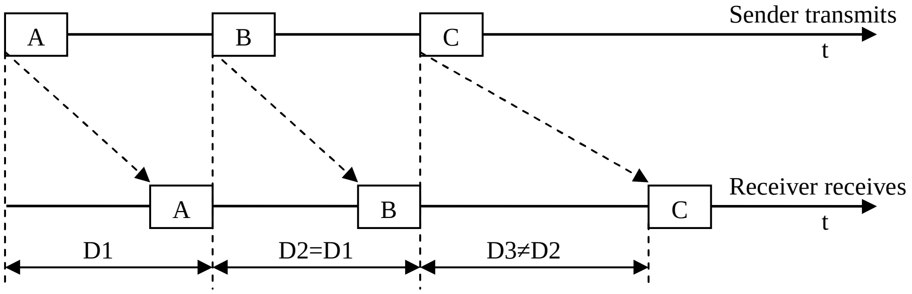


Figure 1. Time diagram showing jitter

Figure 1 illustrates that the time required for both packet A and packet B to travel from the source to the receiver are same. However, packet C encounters delay while traveling through the network and arrives after its expected time. The reverse case may also happen, where a packet may be routed quicker due to sudden increase in the network capacity and the packet reaches its destination before its expected time of arrival. This is the reason that we require a jitter buffer to smooth out the inter-arrival jitter. It may be noted that for non-real-time systems, packets can be stored indefinitely at the local buffer. However, for real-time applications like VoIP, delayed packets may become useless after a certain point of time. The jitter buffer holds the delayed packets in an attempt to neutralize the effects of packet inter-arrival jitter. This helps maintain the liveliness of real-time communication over packet-switched networks. The jitter buffer must be neither too small nor too large. If the jitter- buffer is too small, it will not serve its purpose on the other hand, if it is too large, it may remain filled with useless packets and hence waste memory space [3].

The need to evaluate the voice quality is important for both technical and commercial reasons. One of the major challenges faced by a network service provider is to measure the voice quality efficiently and accurately for QoS monitoring. The estimation of the perceived QoS forms the basis of all the control mechanisms that seek to enhance the performance of a VoIP session even after the detoriation of the underlying network condition. The voice sample that is obtained from the voice packets is tested for quality at the receiving point. The voice can be tested for quality in two ways, namely, subjective and objective. Humans perform the subjective voice testing by listening to the voice sample, whereas, objective tests are performed by computers [1]. A common subjective benchmark for quantifying the performance of the voice session is the Mean Opinion Score (MOS) [1]. For performing MOS test, a voice sample is given to a group of listeners. They listen to the sample and give a rating on a scale where "excellent" quality is given a score of 5, "good" a 4, "fair" a 3, "poor" a 2, and "'bad" a 1. The ratings given by every member of the group is then averaged to get the MOS.

The E-Model is most commonly used for objective measurements. The basic result of the E-Model is the calculation of the R-Factor. The R-factor is defined in terms of several parameters associated with a voice channel across a mixed Switched Circuit Network and a Packet Switched Network. The parameters included in the computation of the R-factor are fairly extensive covering such factors as echo, background noise, signal loss, codec impairments, and others. R-factor can be expressed by (3) [13].

$$R = 94.2 - I_d - I_e \quad (3)$$

Where, I_d is the impairment associated with the mouth-to-ear delay of the path, I_e is an equipment impairment factor associated with the losses within the gateway codecs. I_d is given by (4).

$$I_d = 0.024T_d + 0.11(T_d - 177.3)H(T_d - 177.3) \tag{4}$$

where, $H(x)$=0 if x<0

$H(x)$=1 if x≥0

and T_d is the end-to-end delay suffered by the voice packet en route from the VoIP source to the VoIP receiver. I_e is given by (5).

$$I_e = A \times \ln(1 + B \times lossratio) + C \tag{5}$$

where A, B and C are constants depending on the codec used.

MOS is related to R-Factor by (6) [13].

For R<0: MOS=1

For R>100: MOS=4.5

For 0<R<100: $MOS = 1 + 0.035R + 7 \times 10^{-6} R(R - 60)(100 - R)$ (6)

3. Related Work

A significant research has already been conducted in the quest of finding a suitable adaptive jitter buffer playout algorithm. As already mentioned, finding the proper playout delay is of utmost importance. In order to find out the efficiency of some of the already existing adaptive jitter buffer algorithms, we have studied five algorithms mentioned in [14] and [15].

3.1. Exponential Average Algorithm (EXP-AVG) [14]

In this algorithm, the delay estimate for the i^{th} packet is computed based on RFC 793 algorithm [16] and the variation in the delays is calculated as suggested by Van Jacobson in the calculation of round-trip-time estimates for the TCP retransmit timer [17]. In this algorithm the estimate of the playout delay for packet i is evaluated by the equation (7).

$$P_i = d_i + 4v_i \tag{7}$$

where, d_i and v_i can be figured out by (8) and (9).

$$d_i = \alpha d_{i-1} + (1 - \alpha)n_i \tag{8}$$

$$v_i = \alpha v_{i-1} + (1 - \alpha)|d_i - n_i| \tag{9}$$

where, n_i denotes the one-way delay of the i^{th} packet and the value of α is 0.998002 [14].

3.2. Fast Exponential Average Algorithm (F-EXP-AVG) [14]

This algorithm is similar to the previous one. The only difference being that if the current packet's network delay 'n_i' is greater than d_{i-1}, then d_i is given by equation (10).

$$d_i = \beta d_{i-1} + (1-\beta) n_i \qquad (10)$$

where, the value of β is 0.75 [14].

3.3. Minimum Delay Algorithm (Min-D) [14]

The primary objective of this algorithm is to minimize the delay. So it uses the minimum value of the network delay suffered by the packets in the current talkspurt to estimate the playout delay of the next talkspurt. Let S_i be the set of all packets received in the talkspurt prior to the one initiated by i. So, the delay estimate for packet i is calculated by (11). Apart from this modification, this algorithm is similar to the EXP-AVG algorithm.

$$d_i = \min_{j \in S_i} \{n_i\} \qquad (11)$$

β is 0.75 [14].

3.4. Spike Detection Algorithm (Spike-Det) [14]

One of the most common phenomena that can be observed in a VoIP system is that some of the packets may suddenly suffer from high end-to-end delay. The result of this effect is that no voice packet reaches the receiver for some time followed by the arrival of a large number of voice packet reaching almost simultaneously. We describe this phenomenon as the 'spike'. The above mentioned algorithms do not take care of this problem. However, this algorithm seeks to overcome the problem with the incorporation of a spike detection mechanism. When, a spike is detected, the algorithm switches to 'SPIKE' mode and later reverts back to 'NORMAL' mode when the network condition becomes normal. The basic concept of this algorithm is similar to the EXP-AVG algorithm. However, the value of α is set to 0.875 here. The Spike-Det algorithm can be summarized by the following steps, where d_i is the delay estimate of the i^{th} packet and *var* is the variation in delay.

```
1.  $n_i = i^{th}$ packet network delay;
2.  IF (mode == NORMAL) {
            if (abs($n_i – n_{i-1}$) > abs($v_i$) * 2 + 800)
            var = 0; /* Detected beginning of spike */
            mode = SPIKE;
    }
    ELSE {
            var = var/2 + abs((2$n_i$ - $n_{i-1}$ – $n_{i-2}$)/8);
            if (var <= 63) {
                    mode = NORMAL; /* End of spike */
                    $n_{i-2} = n_{i-1}$;
                    $n_{i-1} = n_i$;
                    return;
            }
    }
```

```
3.  IF ($mode$ == NORMAL)
            $d_i$ = 0.125 * $n_i$ + 0.875 * $d_{i-1}$;
    ELSE
            $d_i = d_{i-1} + n_i – n_{i-1}$;
    $v_i$ = 0.125 * abs($n_i$ - $d_i$) + 0.875 * $v_{i-1}$;
4.  $n_{i-2} = n_{i-1}$;
    $n_{i-1} = n_i$;
    return;
```

3.5. Window Algorithm [15]

This algorithm collects the network delays of previously received packets and uses them to estimate the playout delay of the incoming packets. The network delays of last few packets are collected and the delay distribution is updated with every incoming talkspurt. The playout delay of the incoming packet is chosen by obtaining a delay that represents a given percentile among the last few received packets. The determination regarding the playout delay is made with the help of an incrementally updated histogram. When a new packet arrives, its delay is added to the histogram, and the delay of the oldest packet is removed. This algorithm also detects spikes. On detection of a spike, the algorithm stops collecting packet delays. If a talkspurt starts during a spike, then the delay of the first packet of the talkspurt is used as the playout delay for that talkspurt. The efficiency of determination of playout delay for this algorithm depends on the window size, i.e. the number packets considered for recording their delay. If the window is too small, then the estimation of playout delay is likely to be poor. On the other hand, if the window size is too large, large memory is wasted for keeping tracks of long and unnecessary history. The window algorithm is defined by the following steps. Here, d_i is the network delay of the i^{th} packet and p_i is the playout delay of the i^{th} packet. Two parameters *head* and *tail* are used in this algorithm in detecting the beginning and end of a spike respectively. c*urr-delay* is the given percentile point based on the previous statistics of packet delays.

```
IF ($mode$ == SPIKE)
        IF ($d_i \leq tail$ * old_d)        /* the end of a spike */
                $mode$ == NORMAL;
ELSE
        IF ($d_i > head * p_{i-1}$) {       /* the beginning of a spike */
                $mode$ = SPIKE;
                $old\_d = p_{i-1}$;         /* save $p_i$ to detect the end of a spike   later */   }
        ELSE {
                IF (delays[curr_pos] ≤ curr_delay)
                        count = count - 1;
                distr_fcn[delays[curr_pos]] = distr_fcn[delays[curr_pos]] - 1;
                delays[curr pos] = $d_i$;
                curr_pos = (curr_pos+1) % $w$;
                distr_fcn[$d_i$] = distr_fcn[$d_i$] + 1;
                IF (delays[curr_pos ] < curr_delay)
                        count = count + 1;
                WHILE (count < $w$ $x$ $q$) {
                        curr_delay = curr_delay + unit;
                        count = count + distr_fcn[curr_pos];
                }
                WHILE (count > $w$ $x$ $q$) {
                        curr_delay = curr_delay - unit;
                        count = count – distr_fcn[curr_pos]; }        }
```

4.THE SIMULATION SETUP

We have created the congested network scenario used for the analysis of the above mentioned adaptive jitter buffer playout algorithms and to assess our new algorithm with the help of OPNET Modeler 14.5.A. The set up consists of four nodes. Two ethernet4_slip8_gtwy_adv gateways are used to interface IP cloud to the communicating nodes. The IP cloud serves the purpose of simulating the presence of an IP backbone in the communication path of the nodes. The gateways and the IP cloud are connected with PPP_adv link whose data rate can be altered.

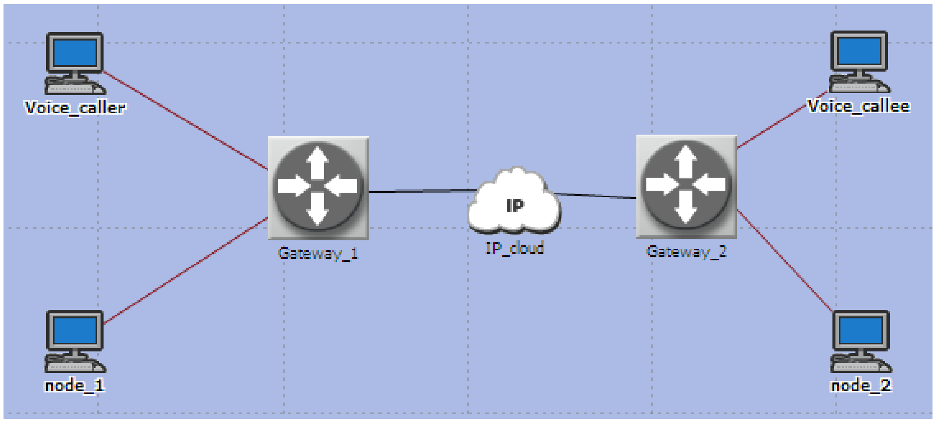


Figure 2. The OPNET Simulation Setup

It is seen from Figure 2, that one of the nodes acts as the Voice caller whereas another node acts as the Voice callee. These two nodes exchange voice packets between each other. Both these nodes are configured to use G.726 ADPCM coder with 32 kbps bit rate and it produces traffic at a constant rate. The other two nodes, i.e. node 1 and node 2 interchange packets unrelated to the VoIP communication, i.e. the cross traffic. Their communication bit rate varies randomly every second between the lower and upper extremes of 0 kbps and 1000 kbps respectively. The basic purpose of these nodes is to congest the links between the gateways and the IP cloud. It is worth mentioning that in order to simulate various network behavior, we have simulated the network several times with the capacity of the PPP_adv link having the values of 600 kbps, 800 kbps, 1000 kbps, 1200 kbps and 1400 kbps. These values enable us to study how the VoIP communication behaves when the cross traffic bit rate exceeds the link capacity and then again reduces below the capacity of the link, as in case of 600 kbps and 800 kbps. In case of link capacity of 1000 kbps, 1200 kbps and 1400 kbps, the cross traffic is always below the link capacity since the maximum value that can be attained by the bit rates of node 1 and node 2 is 1000 kbps. This is how we have created a varying network, so as to induce variable end-to-end delay to the voice packets exchanged between the pair of voice nodes. The IP cloud serves to simulate the routing functionalities and can also increase the delay and packet loss rate. For simplicity, only the results with network capacities 600 kbps, 1000 kbps and 1400 kbps are shown as they cover the three types of jitter conditions, i.e. a network with high jitter (600 kbps network), a network with moderate jitter (1000 kbps network) and a network with low jitter (1400 kbps network).

5. ANALYSIS OF THE EXISTING ADAPTIVE JITTER BUFFER PLAYOUT ALGORITHMS

The link capacity has been set in accordance with each of the above mentioned values and VoIP call simulations have been carried out between the two voice nodes to study their behaviour. The end-to-end delay values for each of the voice packets were noted. Later, these set of readings have been used to implement the various algorithms and then we have compared the results to find out the improvement in VoIP performance, i.e. reduction in jitter.

It is observed from Table 1 that for a network which induces high jitter to the voice packets passing through it (network capacity of 600kbps), the F-EXP-AVG algorithm discards least number of packets, and hence the lowest discard ratio. However, it increases the playout delay to such an extent that the average delay increases beyond a tolerable value and hence the voice quality degrades. The other algorithms induce lower average delay, but discard a large number of voice packets since the packets arrive after the estimated playout time. As a result, the voice call standards go below tolerable limits. The average Mean Opinion Score (MOS) reflects the voice quality offered by each of the algorithms. It is evident that none of the algorithms perform satisfactorily under high jitter conditions.

Table 1. Results for the algorithms for different network capacities

Network Capacity	**Algorithm**	**Avg. delay (ms)**	**Discard ratio (%)**	**Avg. MOS**
600 kbps	EXP-AVG	493.94	6.289	1.0232
	F-EXP-AVG	1446.76	0.482	1.0041
	Min-D	417.80	8.867	1.0191
	Spike-Det	338.17	6.318	1.0461
	Window	341.49	5.524	1.0320
1000 kbps	EXP-AVG	119.83	3.928	1.9114
	F-EXP-AVG	276.46	0.453	2.1900
	Min-D	112.12	6.547	1.6462
	Spike-Det	115.12	6.867	1.6143
	Window	103.63	3.865	1.9386
1400 kbps	EXP-AVG	89.52	0.756	2.8233
	F-EXP-AVG	102.93	0.049	3.4264
	Min-D	88.53	2.681	2.1664
	Spike-Det	87.69	4.728	1.8486
	Window	86.10	0.516	2.9840

In a network with moderate congestion (network capacity of 1000 kbps) and consequently moderate jitter, the average delay induced by the algorithms decreases considerably. However, the F-EXP-AVG still imparts higher delay to the voice packets whereas, the Min-D and Spike-Det discards a large number of packets, thereby, suffering from large losses. Further, the performances of the algorithms in a network with low congestion (network capacity 1400 kbps) are also tabulated. Here, we can say that since the inter-arrival jitter for the packets is low, the playout algorithms do not incorporate a significant playout delay to the voice packets. Hence, the end-to-end delay does not increase much. However, we can see that the Spike-Det and Min-D algorithms discard a high percentage of packets and as a result the call quality provided by them gets degraded.

6. Proposed Adaptive Jitter Buffer Playout Algorithm

After extensive analysis of some of the existing jitter buffer algorithms, we have come to the conclusion that, when the jitter imparted by the network to the voice packets is very high, the playout delay increases considerably. Moreover, the packet discard ratio also increases beyond tolerable limits. The net result of the above two factors is degradation in the quality of voice in the VoIP session. Our algorithm seeks to reduce the playout delay and packet discard ratio. Our algorithm can be summarized in the following steps which are to be followed as long as the VoIP call continues.

```
1. $n_i = i^{th}$ packet network delay, $\alpha$ =0.875;
2. DD = abs($n_i$ - $n_{i-1}$);        /* DD indicates the absolute value of the difference in
                                       network delay of 2 consecutive packets*/
3. IF ($i < w$)                        /* w indicates the number of packets to be considered or
                                       the window size */
           find out the inter-quartile range of 'i-1' packets;
   ELSE
           find out the inter-quartile range of the last 'w' packets;
4. IF (inter-quartile range < 5)
           IF ($\alpha$ +0.01< 0.998002)
                   $\alpha = \alpha + 0.01$;
           ELSE
                   $\alpha = 0.998002$;
   ELSE
           IF ($\alpha$ – 0.05 > 0.75)
                   $\alpha = \alpha - 0.05$;
           ELSE
                   $\alpha = 0.75$;
5. IF(mode == NORMAL)
           IF (DD > $(1 - \alpha)$ x 100)
                   mode = SPIKE;
           ELSE
                   goto step 6;
   ELSE IF ($n_i < \alpha$ x $n_{i-1}$)  /* that is, mode = SPIKE */
           mode = NORMAL;
   Else
           goto step 6;
6. IF(mode == SPIKE)
           $d_i$ = 0.75 x $d_{i-1}$ + (1-0.75) x $n_i$;
   ELSE                                        /* that is, mode = NORMAL */
           IF(new talkspurt)
                   $d_i = n_i$;
           ELSE
                   $d_i = \alpha$ x $d_{i-1} + (1 - \alpha)$ x $n_i$;
7. $Y$ = abs($d_i$ - $n_i$) ;
8. $v_i$= 0.998002 x $v_{i-1}$ + (1-0.998002) x $Y$;
9. Playout delay = $d_i$ + 4 $v_i$;
```

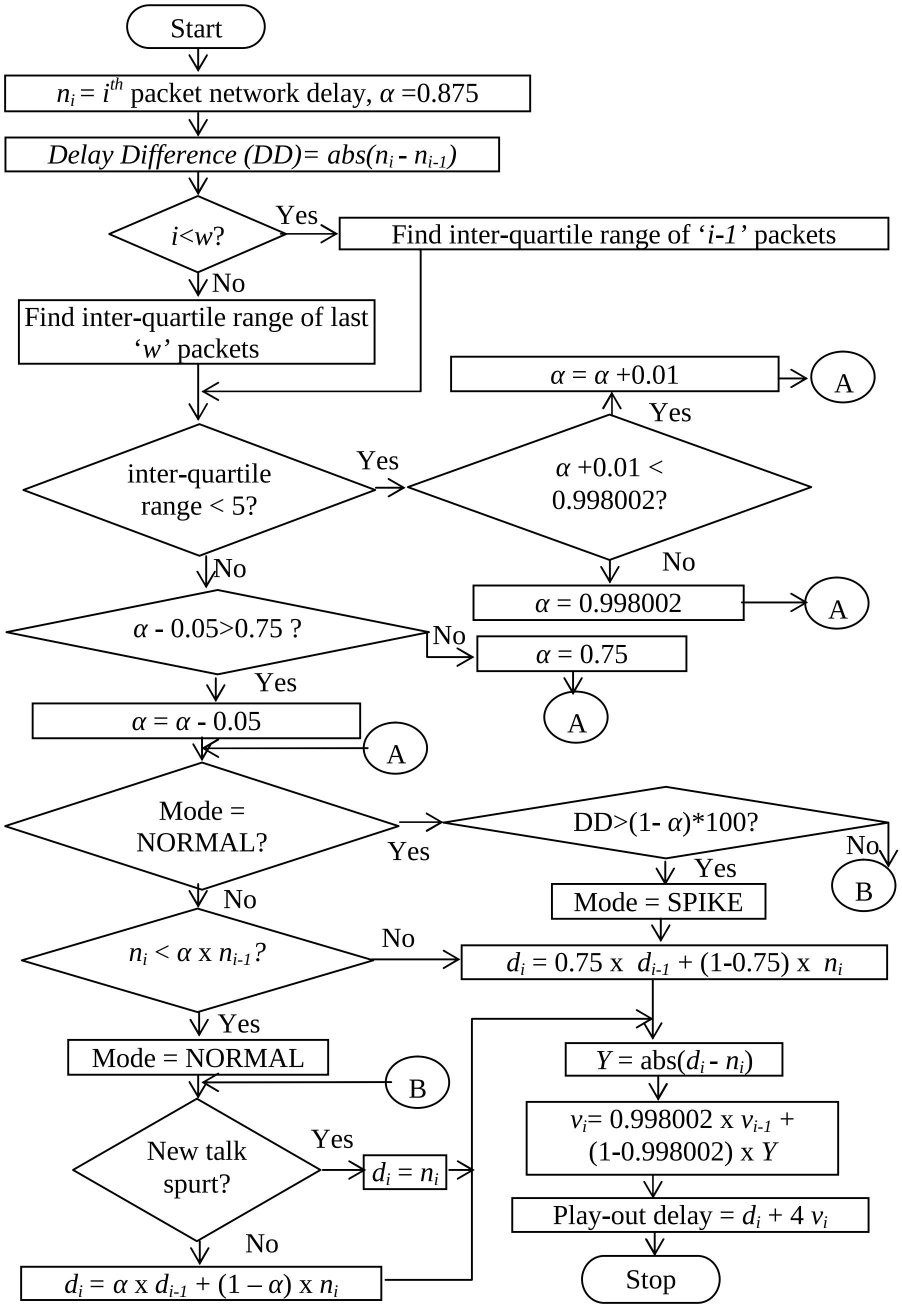

Start
$n_i = i^{th}$ packet network delay, α =0.875
Delay Difference (DD)= abs($n_i - n_{i-1}$*)*
$i<w$?
Yes
Find inter-quartile range of ‘*i-1*’ packets
No
Find inter-quartile range of last ‘*w*’ packets
$\alpha = \alpha + 0.01$
A
Yes
inter-quartile range < 5?
Yes
α +0.01 < 0.998002?
No
No
α = 0.998002
A
α - 0.05>0.75 ?
No
α = 0.75
Yes
A
$\alpha = \alpha - 0.05$
A
Mode = NORMAL?
DD>(1- α)*100?
Yes
No
Yes
B
Mode = SPIKE
No
$n_i < \alpha$ x n_{i-1}?
No
d_i = 0.75 x d_{i-1} + (1-0.75) x n_i
Yes
Mode = NORMAL
Y = abs($d_i - n_i$)
B
v_i= 0.998002 x v_{i-1} + (1-0.998002) x Y
New talk spurt?
Yes
$d_i = n_i$
Play-out delay = d_i + 4 v_i
No
$d_i = \alpha$ x d_{i-1} + (1 – α) x n_i
Stop

Figure 3. Flowchart of the Proposed Adaptive Jitter Buffer Playout Algorithm

We estimate the network characteristics by keeping track of the last ‘*w*’ received packet. ‘*w*’ represents a sliding window. This sliding window helps the receiver to estimate the playout delay of the future packets from the end-to-end delays of the previous ‘*w*’ received packets. Choosing a small value for ‘*w*’ may lead to improper estimation of playout delay whereas a high value of ‘*w*’ may lead to wastage of memory by keeping track of unnecessary packets. Hence, the value of ‘*w*’ should be chosen carefully to get the best results. We use this accordingly to vary the value of α, where α is a parameter that determines how much a newly received packet depends on the previously received packets. The algorithm is pictorially represented in Figure 3.

7. RESULTS

The QoS of a VoIP call can be best described by the MOS value as both the end-to-end delays of the packets and the packet loss are considered for the calculation of the MOS. The algorithm was initially implemented by setting the value of '*w*' as 100. Following this development, the effect of varying window sizes on the performance of the algorithm was measured. Finally, a comparative analysis of the performance of the proposed algorithm with the performances of the analyzed algorithms has been made in order to assess the improvement offered by the implementation of the proposed adaptive jitter buffer playout algorithm. The results obtained have been included in the following subsections.

7.1. Implementing the Algorithm with a Window Size of 100

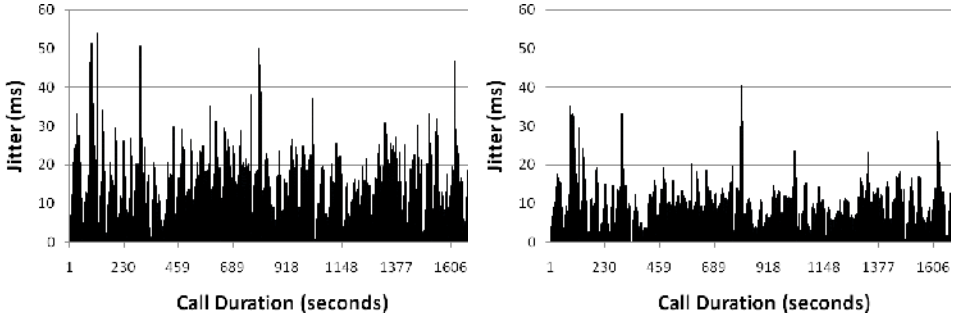
Initially, the algorithm used the data from the latest 100 packets in order to estimate the recent trends in the network characteristics. Several simulations were being carried out to find out the effectiveness of the proposed algorithm and it gave a better MOS than the other discussed algorithms. The results also show that it reduced jitter considerably.

Table 2. Results for the proposed algorithm for different network capacity with a window size of 100

Bandwidth (kbps)	Avg. delay (ms)	Packet Discard Ratio (%)	Average MOS	Percentage Reduction in Jitter
600	276.93	2.635	1.4508	35.34
800	161.42	2.846	2.0657	38.96
1000	107.80	2.806	2.1059	44.86
1200	92.50	0.664	2.8709	50.19
1400	87.68	0.171	3.2932	54.49

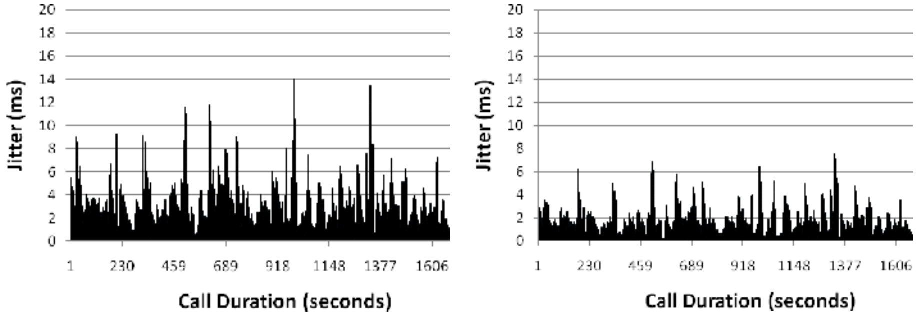
It can be seen from Table 2 that the proposed algorithm performs satisfactorily for all of the above network scenarios. However, the MOS is low for a network bandwidth of 600 kbps because the network congestion is very high in this case. As a result there is a very large jitter between the consecutive voice packets. So, in order to compensate for jitter, the algorithm incorporates a significant playout delay, as a result of which, end-to-end delay is increased. Moreover, due to the high unpredictable nature of the network, some packets arrive later than it is estimated and hence they are rejected. So the discard ratio is also high. Both these factors degrade voice quality and hence the average MOS rating is also reduced. However, as the capacity of the network increases, the probability of the network becoming congested is also lowered. As a result, the jitter also decreases with increase in network capacity. This enhances VoIP performance. The algorithm, however, still incorporates a small playout delay, in order to reduce the jitter further.

In Figure 4, it can be seen that the jitter reduces significantly following the application of the proposed adaptive jitter-buffer playout algorithm. The average jitter throughout the duration of the call falls from 18.38 ms to 11.88 ms. The above result reflects the effectiveness of the algorithm under congested network that impart high jitter to the voice packets passing through it. Figure 5 and Figure 6, further illustrates the behaviour of the algorithm to moderate and low jitter, where the network capacity is 1000kbps and 1400kbps respectively. It is seen from the figures that the proposed algorithm reduces jitter considerably and performs well, in all three network scenarios.



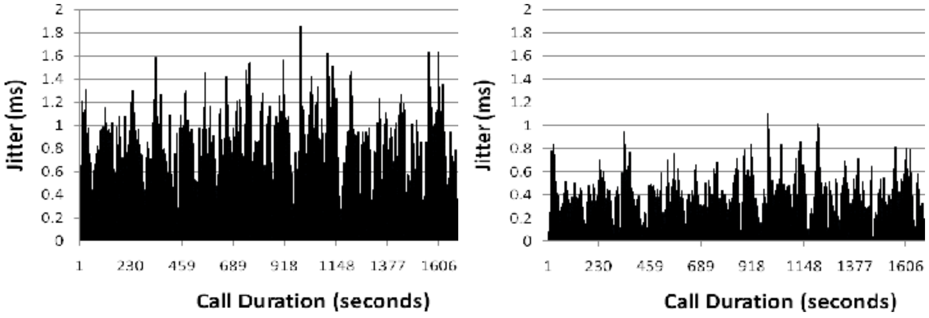
(a) (b)

Figure 4. The Inter-arrival jitter for network capacity of 600 kbps (a) Without playout buffer (b) With proposed algorithm (window size 100)



(a) (b)

Figure 5. The Inter-arrival jitter for network capacity of 1000 kbps (a) Without playout buffer (b) With proposed algorithm (window size 100)



(a) (b)

Figure 6. The Inter-arrival jitter for network capacity of 1400 kbps (a) Without playout buffer (b) With proposed algorithm (window size 100)

7.2. Effect of various Window Sizes

After establishing the effectiveness of our proposed adaptive jitter buffer algorithm, we conducted further experimentations in order to find out the optimum window size for all network conditions. For this purpose, we have conducted exhaustive simulations using various window sizes. We have primarily focussed on maximizing the MOS of the VoIP call as it reflects the call quality from the user's point of view. However, we have also considered the other important parameters that reflect the QoS of a call from the technical point of view. These parameters include Average Delay, Packet Discard ratio and Percentage improvement in jitter. The effects of window size on the VoIP call performance have been illustrated in Table 3.

Table 3. Effect of Window Size on the Performance of the Proposed Algorithm for different Network Capacities

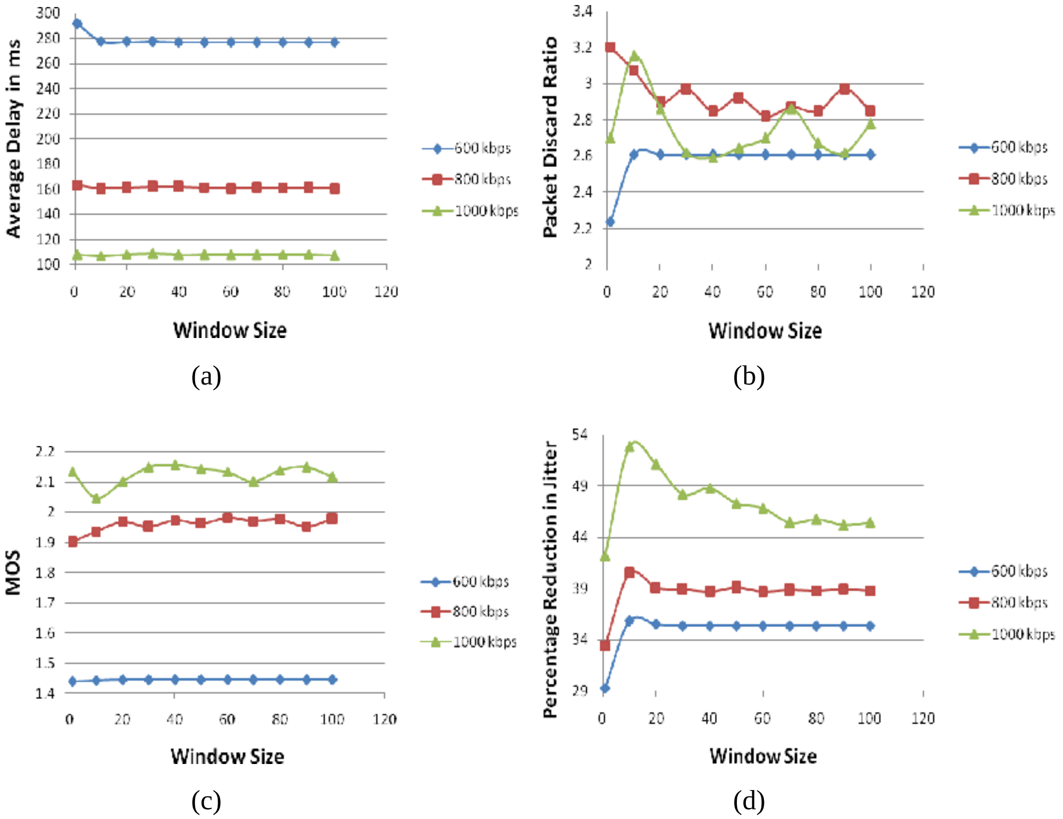
Network Capacity	Window Size	Avg. delay (ms)	Discard ratio (%)	Avg. MOS	Percentage Reduction in Jitter
600 kbps	0	291.71	2.238	1.4405	29.27
	20	277.13	2.635	1.4447	35.51
	40	277.02	2.635	1.4451	35.37
	60	276.97	2.635	1.4508	35.35
	80	276.93	2.635	1.4508	35.34
	100	276.93	2.635	1.4508	35.34
800 kbps	0	163.55	3.201	1.9922	33.56
	20	161.27	2.896	2.0571	39.26
	40	161.83	2.846	2.0540	38.84
	60	160.98	2.820	2.0712	38.86
	80	161.47	2.846	2.0656	38.93
	100	161.42	2.846	2.0657	38.96
1000 kbps	0	107.96	2.701	2.1333	42.17
	20	108.14	2.860	2.1012	51.10
	40	107.93	2.592	2.1561	48.80
	60	108.04	2.699	2.1335	46.85
	80	108.28	2.672	2.1387	45.74
	100	107.80	2.806	2.1059	44.86

From Table 3 we can clearly see that for a network capacity of 600 kbps, the best improvement in jitter is obtained for a window size of 40. However, the best average MOS value is obtained with a window size of 60. As the network capacity increases, the optimum window size seems to decrease. For a network with capacity of 800 kbps and 1000 kbps the highest improvement in jitter is obtained for a window size of only 20. But, the best MOS is observed for a window size of 40 and 60 respectively. When looking at the least average delay and packet discard ratio for various window sizes, we can clearly see the algorithm gives the best results for the window sizes between 40 and 60.

It may be noted that the results for network capacities of 1200 kbps and 1400 kbps have not been included in Table 3 because the values remain constant even though there is change in the window size. This is primarily due the fact that these two scenarios induce very low jitter to the passing voice packets. Hence, the playout delay mostly depends on the end-to-end delay of the received packet only. So, the window size does not have any effect on the performance of the algorithm.

The results can be further verified by Figure 7(a), 7(b), 7(c) and 7(d). Figure 7(a) reflects the effect of window size on Average Delay of the voice packets for the duration of the voice call. It is seen that the delay is high if a zero window size is used. However, as the window size is increased to 10 the delay decreases considerably and remains fairly constant with further increase in window size. Figure 7(b) on the other hand shows that the window size has a profound effect on the packet discard ratio. It is clearly seen that the best results in terms of packet discard ratio is obtained for a window size of 40. Figure 7(c) provides illustration for the most important of the VoIP QoS parameters, the MOS. It is seen that the MOS depends on the size of the window used for the operation of the proposed adaptive jitter buffer algorithm. It is seen that the best MOS values are obtained for around two window sizes, i.e. 40 and 80. Since, choosing a lower window size helps to reduce the memory consumption and the processing delay, it is better to choose a window size of 40 as an optimum value.

Figure 7(d) provides the analysis results of the dependency of percentage improvement of jitter with respect to changing window sizes. Here, we find that the best results are obtained for a window size of 20. However, the best MOS value is not obtained for a window size of 20 and hence the algorithm will fail to provide its full potential if we resort to a window size of 20. Hence, we sacrifice a little improvement in terms of jitter in order to provide a more satisfactory user experience. Thus, after a thorough analysis we come to the conclusion that a window size of 40 can be considered as the optimum value for a seamless operation of the proposed algorithm.



(a)

(b)

(c)

(d)

Figure 7. A comparison of various parameters w.r.t. window size (a) Average end-to-end delay, (b) Packet Discard ratio, (c) MOS, (d) Percentage reduction in Jitter.

7.3. Comparative Results with the other Analyzed Algorithms

Further endeavours have been taken in order to find out where the proposed adaptive jitter algorithm stands, when compared with the performances of the other discussed algorithms. Our algorithm has given the lowest end-to-end delay among all the algorithms especially when the network is more congested and the extent of jitter in the voice packets is very high. The end-to-end delay results can be observed in Figure 8. When examined for packet discard ratio it has been found that our algorithm performs quite well. However, the F-EXP-AVG algorithm has even lower packet discard ratio. The packet discard ratio comparison has been illustrated in Figure 9. Upon further examinations, it is observed that for congested mediums, our algorithm gives the best MOS values. However, for network with very low jitter, F-EXP-AVG gets the edge because of its lower packet discard ratio. In a nutshell, we can say that the existing algorithms aim to minimize either end-to-end delay or packet loss. While one of the above mentioned factors is reduced, the other gets worse. As a result, voice quality gets degraded. Our algorithm aims to find a trade-off between these two factors. Hence, it provides the best voice quality. The comparative results of MOS values have been included in Figure 10.

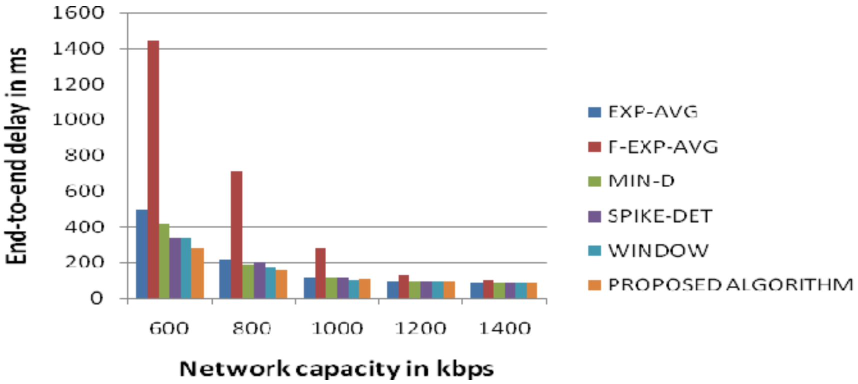


Figure 8. Comparison of the end-to-end delays of the different algorithms

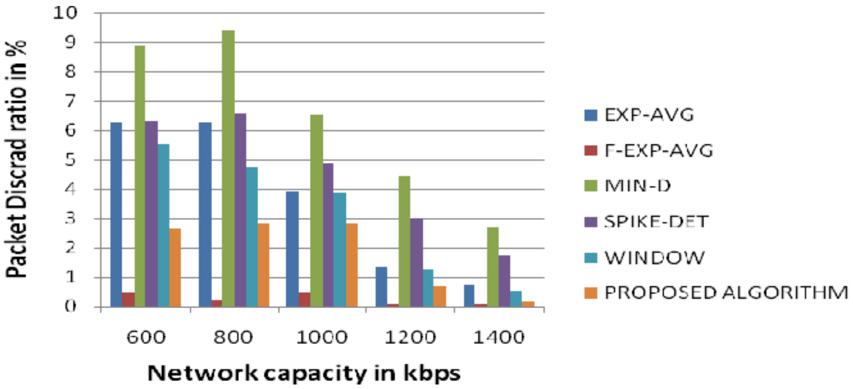


Figure 9. Comparison of the packet discard ratio of the different algorithms

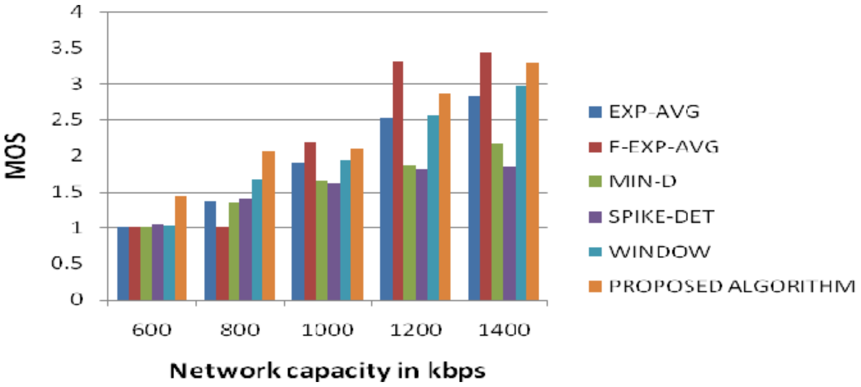


Figure 10. Comparison of MOS of the different algorithms

8. Conclusion

A congested network transports voice packets with uneven delay. The result of this unevenness is incorporation of jitter in the consecutive voice packets. Jitter is not desirable during a voice call as it leaves the user dissatisfied. Several, algorithms have already been proposed to add a further playout delay to the voice packets in hope of minimizing the jitter. However, selecting the optimum playout delay is a tricky part. These algorithms often under-estimate or over-estimate the network delay of future incoming voice packets, resulting in discarding of the packets or long undesirable end-to-end delay respectively. We have proposed an algorithm that addresses to this problem and properly estimates the network delay of the future incoming voice packets. Our algorithm takes the help of a sliding window in order to get an estimate of the variation in network conditions. After conducting extensive analysis, we have found out that in order to get the best performance the optimum window size should be 40. Our algorithm aims to enhance the QoS of the VoIP session. It seeks to decrease the end-to-end delay and packet discard ratio while allowing a tolerable amount of jitter to be present in the voice packets. The primary aim of our algorithm is to enhance user experience by improving the MOS of the call. We are conducting further studies in order to get even better QoS for the voice calls in a congested network scenario.

Acknowledgements

The authors deeply acknowledge the support from DST, Govt. of India for this work in the form of FIST 2007 Project on "Broadband Wireless Communications" in the Department of ETCE, Jadavpur University.

References

[1] J. Davidson, J. Peters, A Systematic Approach to understanding the basics of VoIP, Voice over IP Fundamentals, CISCO press, 2000.

[2] J. C. Bolot, A. Vega-Garcia, "Control mechanisms for packet audio in the Internet," in Proc. IEEE Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation, pp.232-239, San Francisco, USA, 24-28 March 1996.

[3] B. Khasnabish. Implementing Voice over IP, 12 June 2003, John Wiley & Sons, Inc.

[4] Chi Sanghyun, B.F. Womack, "QoS-based adaptive playout scheduling based on the packet arrival statistics: Capturing local channel characteristics," in Proc. 2010 IEEE International Workshop

Technical Committee on Communications Quality and Reliability, June 2010. doi: 10.1109/CQR.2010.5619942.

[5] A. Mukhopadhyay, T. Chakraborty, S. Bhunia, I. Saha Misra, S.K. Sanyal, “Study of enhanced VoIP performance under congested wireless network scenarios,” in Proc. Third International Conference on Communication Systems and Networks (COMSNETS 2011), pp.1-7, 4-8 Jan. 2011, Bangalore, India. doi: 10.1109/COMSNETS.2011.5716509

[6] M.F. Hong, H.W. Hsu, W.L. Du, Y. H. Hung, M.H. Lee, “A Fast-Startup TCP Mechanism for VoIP Services in Long-Distance Networks,” in Proc. International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp 185-188, December 2006. doi: 10.1109/IIH-MSP.2006.264976

[7] J. Yan, M. May, B. Plattner, “Distributed and Optimal Congestion Control for Application-layer Multicast: A Synchronous Dual Algorithm,” in Proc. 5th IEEE Consumer Communications and Networking Conference, pp. 279-283, 10-12 January 2008. doi: 10.1109/ccnc08.2007.69.

[8] A.B. Rus, M. Barabas, G. Boanea, Z. Kiss, Z. Polgar, V. Dobrota, “Cross-Layer QoS and its application in congestion control,” in Proc. 2010 17th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN), pp. 1-6, May 2010. doi: 10.1109/LANMAN.2010.5507149

[9] T. Chakraborty, A. Mukhopadhyay, I. Saha Misra, S.K. Sanyal, “Optimization Technique for Configuring IEEE 802.11b Access Point Parameters to improve VoIP Performance”, in Proc. International Conference of Computer and Information Technology, 23-25 December 2010, Dhaka, Bangladesh. doi: 10.1109/ICCITECHN.2010.5723919.

[10] P. Gournay, K.D. Anderson, “Performance Analysis of a Decoder-Based Time Scaling Algorithm for Variable Jitter Buffering of Speech Over Packet Networks,” in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, May 2006. doi: 10.1109/ICASSP.2006.1659946.

[11] K.M. McNeill, M. Liu, J.J. Rodriguez, “An Adaptive Jitter Buffer Play-Out Scheme to Improve VoIP Quality in Wireless Networks,” in Proc. IEEE Military Conference, October, 2006. doi: 10.1109/MILCOM.2006.302119.

[12] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, RFC 3550, July 2003.

[13] R.G. Cole, J.H. Rosenbluth, “Voice Over IP Performance Monitoring” ACM SIGCOMM Computer Communication Review, Volume 31 Issue 2, April 2001.

[14] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, “Adaptive Playout Mechanism for Packetised Audio Application in Wide-area Networks,” in Proc.of INFOCOM '94. Networking for Global Communications, pp.680-688 vol.2, 12-16 Jun 1994 doi: 10.1109/INFCOM.1994.337672.

[15] S. B. Moon, J. Kurose, D. Towsley, “Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms,” Acm/Spring Multimedia Systems, Vol. 6, No. 1, February 1998, pp. 17-28. doi:10.1007/s005300050073.

[16] J. Postel, editor, Transmission Control Protocol specification: ARPANET Working Group Requests for Comment (September 1981), RFC 793.

[17] V. Jacobson, “Congestion avoidance and control,” in Proc. 1988 ACM SIGCOMM Conf., August 1988, Stanford, pp. 314-329.

Author

AtriMukhopadhyay (atri.mukherji11@gmail.com) has studied Electronics and Communication Engineering from Meghnad Saha Institute of Technology under West Bengal University of Technology, Kolkata, India and received his Bachelor of Technology Degree in 2009. He has received Master of Technology in Distributed and Mobile Computing from Jadavpur University, Kolkata, India in 2011. His research interests focus on Voice over IP and Wireless Communication.

Tamal Chakraborty

(tamalchakraborty29@gmail.com) has done Master of Technology in Distributed and Mobile Computing from Jadavpur University, Kolkata, India. He has studied Computer Science and Engineering from Future Institute of Engineering and Management under West Bengal University of Technology, Kolkata, India and received his Bachelor of

Technology Degree in 2009. After the completion of his M.Tech, he has joined the Department of Electronics & Tele-communication Engineering, Jadavpur University as a Senior Research. Fellow. His research interests include Voice over IP and Wireless Communication

Dr. Iti Saha Misra (itisahamisra@yahoo.co.in; iti@etce.jdvu.ac.in) is presently holding the post of Professor in the Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata, India. After the completion of PhD in Engineering in the field of Microstrip Antennas from Jadavpur University (1997), she is actively engaged in teaching since 1997. Her current research interests are in the areas of Mobility Management, Network Architecture and protocols, Integration Architecture of WLAN and 3G Networks, Call Admission control and packet scheduling in cellular and WiMAX networks. Her other research activities are related to Design Optimization of Wire Antennas using Numerical Techniques like GA, PSO and BFA.

She is the recipient of the Career award for Young teachers by All India Council for Technical Education (AICTE) for the financial year 2003-2004 and obtained the IETE Gowri memorial award in 2007 for being the best paper in the general topic of 4G networks: Migration to the Future. She has developed the OPNET, QualNet and VoIP laboratories in the Department of Electronics and Telecommunication Engineering of Jadavpur University to carry out advanced research work in Broadband wireless domain.

Mrs. Saha Misra is the Senior Member of IEEE and founder Chair of the Women In Engineering, Affinity Group, IEEE Calcutta Section.

Dr. Salil K. Sanyal (s_sanyal@ieee.org) received his Ph.D (Engineering) Degree from Jadavpur University, Kolkata – 700032, West Bengal, India in 1990. He joined the Department of Electronics and Telecommunication Engineering, Jadavpur University as Lecturer in 1982 where currently he holds the post of Professor. He is the immediate past Head of the Department of Electronics and Telecommunication Engineering of Jadavpur University. Mr. Sanyal is a Senior Member of IEEE and also the past Chair of IEEE Calcutta Section. His current research interests include Analog/Digital Signal Processing, VLSI Circuit Design, Wireless Communication and Tunable Microstrip Antenna.