An Adaptive Receiver Buffer Adjust Algorithm for VoIP Applications Considering Voice Characters

Jing Liu, Zhisheng Niu
Department of Electronic Engineering, Tsinghua University, Beijing 100084, CHINA

Abstract — In this paper, a new adaptive receiver buffer adjust algorithm is proposed for VoIP applications with the consideration of voice characters and network conditions. It divides network status into two modes: normal mode and spike mode, according to the delay of the coming packets and the size of buffer. In normal mode, the receiver adjusts the buffer delay at the beginning of every talk-spurt according to the delay jitters collected during the last talk-spurt. In spike mode, two buffer adjust methods are proposed. One is to adjust the buffer delay for every packet based on a prediction of the delay according to the delay of former packets; the other is to adjust the buffer delay based on a prearranged way. Simulations and experiments show that this strategy can well conceal the delay jitter and reduce the packet loss rate.

1. INTRODUCTION

A key challenge for the Internet is to guarantee quality of service (QoS) for real-time applications as voice and multimedia. As for voice applications, the sender transmits the audio packets in a regular interval and the receiver plays out them in the same regular interval. But because the Internet is a best-effort network, the end-to-end delays of different packets are varied by their different routes and the variational network traffic load. The difference of the end-to-end delays is called delay jitter. And if the network congestion occurred, some packets will be lost in the core network. Delay, delay jitter and packet loss rate are the most important factors to its quality. Delay is mostly decided by the core network, so, many methods are investigated in the receiver to reduce packets loss rate and conceal delay jitter.

There is always a finite buffer set in the receiver to absorb jitter and make the packets be played out in regular interval. But a fix buffer also has problems: Late-arriving packets, which arrive after its scheduled play-out time, are discarded; Early-arriving packets, will also be discarded since there is no place to hold it in the buffer, Delay cumulates during congestion and then remains large for a long time. So the buffer must be adaptive to solve those problems.

An approach is to adaptively adjust the silence length between talk spurts to adapt to the change of network situation[1][6]. And the adjustment only use the first several packet of a talk-spurt, so it isn't effective if a congestion happens in the middle of a talk spurt. Thus all the packets in the talk spurt will lose, and the quality of voice will decrease seriously.

Another approach is to predict the delay of every coming packets and adjust the buffer for them[2]. This approach can salvage most late-arriving packets instead of throwing them away; and reduce the jitter caused by the core network. But this method doesn't have some different strategy when network condition is very well and so brings heavy burden to the controller. This method simply trace the change by prediction, didn't consider the character of voice to adopt a good adjust method which can reduce artificial jitter caused by adjustments, as show by Fig 1:

Fig 1. Cause of artificial jitter

In this work, a new adaptive receiver buffer adjust algorithm is proposed for VoIP applications with the consideration of voice characters and network conditions. This algorithm has different buffer adjust methods in different network condition. The adjust method in each condition is decided with the consideration of voice character to maximally improve quality of the voice. In section 2 and 3, network condition and voice character is analyzed to choose the adjust methods. In section 4, the algorithm is described in detail. In section 5, simulation is done to show the performance and advantages of the algorithm. In section 6, a test-bed is set up and the abundant statistical experiment results can be gotten. The results will show the advantage of this adaptive buffer adjust algorithm with compare to other former algorithms.

II. DELAY ANALYSIS

Fig 2 shows the composition of end-to-end delay. Here, $D_{send}$ is the time that the sender needed to send out a packet, $D_{play}$ is the time that the receiver needed to play out a packet, generally
the two is the same. $D_{prop}$ is the end-end propagation delay in network's physical media. $D_{trans}$ is the transmission delay needs in the network, including the delay in the router and the addition resend delay caused by NACK in the MAC layer. They are decided by the network condition. $D_{proc}$ is the delay needs in the receiver to un-packet and decode, commonly, it is a constant. $D_{queuing}$ is the time in jitter absorption buffer. The whole end-end delay $d_i$ is composed by $D_{prop}, D_{trans}, D_{queuing}$ and $D_{proc}$. The difference of the end-end delays is called delay jitter. To eliminate the delay jitter, the end-end delay $d_i$ of different packet $i$ must be the same, but $D_{prop}$ and $D_{trans}$ is hard to be the same, so adjusting $D_{queuing}$ is the way to make the whole delay $d_i$ be a constant.


![Fig. 2. Composition of jitter in Internet.](image)


There are two modes to describe the different network status. One is normal mode. In normal mode, delays are changed below a reasonable delay bound $D_{up}$, i.e. $D_{prop} + D_{trans} + D_{proc} < D_{up}$. Set the $D_{up}$ as the scheduled end-end delay. These packets will wait in the buffer for $D_{up} - D_{prop} - D_{trans} - D_{proc}$ and then all be played out at the schedule time. The other is spike mode. In spike mode, the delays are very large and the delay jitters are changed violently. Some of the packets can't arrive at the receiver before the scheduled play-out time and be discarded.

Commonly, the receiver is in the normal mode. The receiver supervises the network condition according to the former packets' delay and delay jitter and switches to the spike mode when the network condition parameters satisfy the conditions of spike state. The receiver collect network condition though two ways: From the time stamp value of every packet received, and from the size of the jitter absorption buffer. So the conditions to enter into spike mode can be judged in two ways: One is judging through the packet delay. Every packet has its scheduled play-out time. If one packet didn't come at its scheduled play-out time, then it is considered to be lost. If continuous $k$ packets lost, then the system enters into spike mode. The other way is judging through the buffer size. This way is proposed by this paper: Two buffer threshold $L_{low1} < L_{low2}$. If the buffer size is below $L_{low1}$, the system enters the spike mode. Until the buffer size improves to exceed the $L_{low2}$, the system will remain in spike mode.


![Fig. 3. Silence period, talk-spurt and VAD of voice.](image)

III. Voice Characters Analysis

Adaptive buffer control method is to improve the quality of voice transmitted in Internet. So character of human voice is important for choosing the proper way to adjust the buffer delay.

Human voice is heard to be continuous. But in fact, if time scale is set to millisecond, the wave of human voice is not continue. Normal human speech consists of talk-spurt and silence periods. Silence period is the period with no or very low voice energy, it occurs both within a word and between words. In practical system, some hardware or software silence detectors is used to supervise the beginning of talk-spurt and silence period. With the detect strategy, there are some special periods which will be coded as VAD frame. These periods have very low or even no voice energy, but they are adjacent to talk-spurt or even in talk-spurt, so the detector treats them as the stretch of talk-spurt to avoid frequently switch between talk-spurt and silence period[4]. The composition of voice is show by Fig 3. ITU-T has given out a recommendations that the alternating active(talk-spurt) and inactive(silence) periods are exponentially distributed with average durations of 1.004 s and 1.587 s with such VAD detector. Talk-spurts occupy 38.53% of the whole time, while silence periods occupy 61.47% of the whole time[5].

Through experiment and data, we can found the delay of voice packets in Internet has its own special characteristic. In most occasion, the period of a network congestion can be divided into two part. In the first part, the arriving interval between adjacent packets increases quickly, so the network delay is become larger and larger, and if the delay reach to the maximum value, they will even be lost. When the network condition become better, the second part begins. The interval between adjacent packets become smaller gradually, it seems like the packets arrive in a rush, so, the buffer size will become great during the second part of spike mode, so some method must be adopt to reduce the buffer size to reduce the queueing delay. The delay curve of the congestion period is like a spike shown by packets from 12 to 21 in Fig 7.
A prearranged buffer adjust algorithm can be used according to the characters of the spike and the voice, just like the TCP sliding window flow control which is predefined to be additive increase and multiplicative decrease.

At the first part of congestion, the delay increased abruptly. So there are some methods can be used: additive, multiplicative and stepped. Experiments show the Stepped increase way will add obvious several interrupts into the voice stream which make the hearer not comfortable but if the duration between two interrupts is enough long, the interrupts do little harm to the comprehension of the meaning of the voice. If there are VAD frames in the buffer, then the large blank segment can be inserted adjacent to the VAD frames, the harm to the quality of the voice is even less. The experiments of the additive or multiplicative way show they will continuously distort the tone of the voice by change the interval irregularly, and the comprehension of the meaning of the voice will be affected when the distortion is serious just. So the stepped increase way is chosen in this work.

In the second part of congestion, The logarithmic decrease strategy is adopted by this work. If the receiver found the delay reduced obvious, it will reduced the buffer size according to the increment of that step, then remain in the size to conduct the network condition. If delay keeps on to reduce, then it will reduce the buffer size again, but if it founds the delay is increase again, it will return to the largest acceptable delay.

ITU-T has given out recommendations on end-end delay of voice application. Human ears are almost unconscious to end-end delay of 0 to 150 ms. They can also accept end-end delay between 150 and 400 ms. But end-end delay over 400 ms is not acceptable for human ears[3].

In normal mode, because the network delay isn't large in normal mode, we can found a delay bound below 150ms, which all the packets' delay won't beyond. This will avoid the artificial jitter caused by adaptive adjust, without affecting the quality of voice.

Experiment also shows that if the length of the silence gap changes a little, the quality of voice will not be affected. So in normal mode, the buffer only be adjusted through silence gap.

IV. ALGORITHM DESCRIPTION

This paper give out a strategy to adjust the buffer delay according to different mode as show by Fig 4:

In normal mode, after every talk-spurt, we calculate the end-end delay according to the formulation below and adjust the buffer size:

\[ b = E(v_i) + 4 \cdot \sqrt{D(v_i)} \]

Here \( b \) is scheduled constant end-end delay for the coming talk-spurt,

\[ v_i = D_{prop} + D_{trans} \]

is the network delay of the \( i \)th packets of the former talk-spurt. \( E(v_i) \) is the statistical mean value of \( v_i \), and \( \sqrt{D(v_i)} \) is the root mean square of the \( v_i \). The number “4” in the formulation can be replaced by a variable coefficient. It is a experiment parameter[2] to make sure that almost all the packets' delay is involved by \( b \).

In spike mode, there are two different ways to adjust buffer for every packet. One is to use some prediction algorithms to prediction the packets' delay for every packet. Then adjust the scheduled play-out according the predict delay to reserve the delayed packets. The other is to adjust the buffer size according to the prearranged way: stepped increase and logarithmic decrease. As show by Fig 5.

Increase the buffer size was done by insert empty frames adjacent to the place of VAD frames. The steps to reduce the buffer size gradually is:

1. The RTP receiver module refuses to receive VAD frame if buffer size is too large;
2. Discard the VAD frame which is already in the receiver.
buffer.

3. If the congestion is very serious and at that time there are very less VAD frames to reduce the buffer size. Then the method to decrease the buffer size is to squash the play-out time of every packet. This method will change the tone of the voice a little, but properly control the squashing degree will not affect the comprehension of the voice's meaning.

V. SIMULATIONS AND RESULTS

To realize the first kind of per packet buffer size adjust algorithm, NLMS algorithm is chosen to predict the packet delay for every packet. NLMS is proved to be a robust prediction algorithm for a stochastic process with limited quadratic mean deviation. When enter into the spike mode, NLMS algorithm uses the delay of the arrived packets to prediction the delay of the following packets and so decided the play-out time for it. If the packets can't arrive before the predict time, it will be discard.

![Fig. 6. Flow chart of NLMS](image)

The flow chart of the algorithm is show in Fig 6. And the formulation of the algorithm is:[7]

\[ \hat{d}_i = \hat{w}_i^T \cdot \hat{u}_i \]

\[ e_i = \hat{d}_i - \hat{d}_i \]

\[ \hat{w}_{i+1} = \hat{w}_i + \frac{\mu}{\hat{w}_i^T \cdot \hat{u}_i + \alpha} \cdot \hat{u}_i \cdot e_i \]

Here \( \hat{w}_i \) = \( \{d_{i-1}, d_{i-2}, d_{i-3}, ..., d_{i-k}\} \) is the input of the algorithm, \( \hat{d}_i \) is real delay of the \( i \)th packet, it's a scalar gotten when the packet arrives the receiver. Then \( k \) such scalars build up a vector \( \hat{u}_i \) to stand for the delays of former \( k \) packets. \( \hat{d}_i \) is the output of the algorithm, it's the prediction of the delay of the \( i \)th packet according to the \( \hat{u}_i \). \( \hat{w}_i = \{w_1, w_2, ..., w_k\} \) is the coefficients of the filter. The \( w_j \) is weight of the \( (i-j) \)th packet's impact on the predict delay of the \( i \)th packet.

\( e_i \) is the difference between the real delay and the predict delay of the \( i \)th packet, it can be calculated out when the \( i \)th packet arrives.

When get the prediction error \( e_i \), the algorithms use the error to self adjust the weight vector \( \hat{w}_i \), the size and positive or negative of the error will influence the change of the \( \hat{w}_i \). Coefficient \( \mu \) decides the degree of adjustment. Large \( \mu \) will induced to quickly change of the \( \hat{w}_i \). Coefficient \( \alpha \) make sure that the denominator will not be so small that a little error will lead to large change of \( \hat{w}_i \).

Then when get the prediction value, the value can be converted into the packets number to adjust the buffer size. Thus the algorithm is a integer-NLMS prediction algorithm.

Both the int-NLMS prediction and the prearranged buffer control methods were used to do some simulation with the collected real VoIP delay data. Then the following simulation result is gotten. A chosen spike segment is shown as Fig 7, and a little long time slice with several spurt is show as Fig 8.

![Fig. 7. Comparison of two kinds of adaptive buffer adjust method](image)

![Fig. 8. Behavior in spike mode and normal mode](image)

In normal mode, if spike occurs in the former talk-spurt, the constant delay of the following talk-spurt will be large, whereas, the constant delay will be small. In spike mode, we can see that the prearranged adjust method responses to the spike more quickly but the NLMS prediction strategy trail the variety of the delay better. Both of them avoid the packets loss and well trade off the packets loss and the packets delay and jitter in spike mode.

VI. EXPERIMENT AND RESULTS

To get real VoIP delays and more data, a experiment system is set up. The system uses SIP to setup communication. The
system is composed by client terminal, gateway, SIP server, edge router and core Internet. And the terminal can be special VoIP equipment (such as a computer with VoIP software) or common phone add gateway. If the terminal is common phone add gateway, the function module is show in Fig 9. But if the terminal is special VoIP equipment, the 2line-4line module will not need, and all other module will in the special VoIP equipment.

Fig. 9. Terminal composition

Then one such end equipment is access through the PSTN, another is access through the LAN, and then they found out each other and set up communication with the help of a SIP server in the same LAN whose address is known by all the client. Then the two end equipments communicated with each other.

Long time experiments have been done to test the performance of the two kinds of buffer adjust methods. Fig 10 shows their performance with compare to reference basic buffer adjust algorithm. The results show that different adjust method has different advantage, but both of them can improve the quality of the application observably. There are three columns in every sub figure below, they stand separately for reference basic buffer adjust, int-NLMS prediction buffer adjust and TCP-like prearranged buffer adjust method from left to right.

The first sub figure shows their difference in variance of jitter in spike mode. It shows the NLMS prediction can well trace the change of delay and minimize the range of jitter. The second sub figure shows their difference in mean delay in spike mode. It shows both of the two kinds of adjust algorithm will improve the average delay a little. The third sub figure shows their difference in packets loss rate in spike mode. It shows prearranged buffer adjust way can react to the spike quickly and save more delayed packets but caused a larger jitter at the beginning of the spike mode. The fourth sub figure shows their difference in quality of voice. The prearranged buffer adjust makes the tone of the voice more stable, so the quality is a little higher.

VII. CONCLUSION

Buffer adjust method is important to improve the QoS of VoIP application. This paper gives out an adaptive buffer adjust algorithm which combined network conditions and voice characters. The simulation and experiment results prove that it can observably reduce the jitter and packets loss rate with increasing a little delay which is small relative to the whole delay. During the experiments, it was found out that only using the end solutions to improve the jitter is not enough. Future work must be done to combine the ends and the routes, and find out better methods to reduce the delay and delay jitter.

ACKNOWLEDGMENT

The authors would like to thank Hitachi Ltd. for their support in this work, and special thanks to Dr. Ojima, Dr. Tanabe, Ikeda, Ruonian Li, Wen Du, Sheng Jing, Shen Yong and Ming Wan for their valuable comments and suggestions.

REFERENCES