An Adaptive Receiver Buffer Adjust Algorithm for Voice & Video on IP Applications

Jing Liu, Zhisheng Niu
Tsinghua-Hitachi Joint Lab on Ubiquitous IT, Room 3-329, FIT Building, Tsinghua University, Beijing 100084, CHINA

Abstract—In this paper, a new adaptive receiver buffer adjust algorithm is proposed for Voice and Video on IP(V2oIP) applications with the consideration of voice and video’s characters and network conditions. This algorithm take the voice stream as the main stream, adjust the voice buffer according to the voice characters, and synchronize the video stream with it considering video’s characters. This algorithm 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 voice's talk-spurt. In spike mode, buffer is adjusted when every packet arrives. The adjust can based on a prediction of the delay according to the delay of former packets, or based on a prearranged way. Simulations and experiments show that this strategy can well conceal the delay jitter and reduce the packet loss rate.

I. INTRODUCTION

A key challenge for Internet is to guarantee quality of service(QoS) for realtime applications as voice, video and other media. For those realtime application, the sender transmits the packets in a regular interval and the receiver should play out them in the same regular interval. Because of Internet’s best-effort and competitive access mechanisms, the end-end delays of different packets are varied according to their different routes and the variational network traffic load. The difference of the end-end delays is called delay jitter. Another form of jitter is inter-stream jitter or skew, which measures the difference in delay as seen by separate streams pertaining to the same application (such as voice and video). In order to ensure proper intra-stream synchronization, low skew is often required. And if the network congestion occurred, some packets will be lost in the core network. Delay, delay jitter, skew and packet loss rate are the most important factors which destroy the QoS of realtime multimedia application. Delay and packet loss is mostly decided by the core network, delay jitter and skew can be reduced in the receiver.

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 discard 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 for voice 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. The above Adjust methods may all induce artificial jitter. They simply trace the network condition, didn’t consider which kinds of adjustment can reduce artificial jitter according to the character of voice.

The adjust algorithms which have been proposed before for video jitter buffer are much simple contrast to those for voice, because human being is less sensitive to the distortion of video than that of voice, and more information is contains in voice in most communication. General method is only use a fix jitter buffer without adjustment.

We have proposed an adaptive buffer adjust methods for VoIP in [9]. In this work, a new adaptive receiver buffer adjust algorithm is proposed for V2oIP applications with the consideration of voice and video’s 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 media characters to maximally improve QoS. 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 1 shows the composition of end-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. $D_{queueing}$ is the time in jitter absorption buffer. The whole end-end delay $d_i$ is composed by $D_{prop}$, $D_{trans}$, $D_{queueing}$ 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 $D_{queueing}$ is the way to make the whole delay $d_i$ be a constant.

Through experiment and data, we can found the delay of packets in Internet has its own special characteristic. In most occasion, the delay curve of the a network congestion is like an spike which can be divided into two part. In the first part, congestion happens, so the packet delay becomes larger and larger, and if delay beyond the up bound, packets will be lost. When the network condition become better, the second part begins. The packets arrive in a rush, the buffer size becomes large during the second part of spike mode, so some method must be adopt to reduce the buffer size to reduce the queuing delay. The delay curve of the congestion period is like a spike shown by packets from 12 to 21 in Fig 5.

There are two modes to describe the different network status. One is normal mode, The other is spike mode. 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.

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. 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.

The receiver decides the network mode in two ways: One is based on 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.

### III. Media Characters Analysis

Adaptive buffer control method is to conceal or reduce jitter and skew, improve the quality of voice transmitted in Internet. But the adjust will add artificial jitter. So character of voice and video is important for choosing the proper way to adjust the buffer delay, and move the artificial jitter to the place where the jitter less impacts the QoS of the media stream.

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-spurts 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 2. 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].

The character of video on IP is decided by the given video code and the selected parameters. The rate of video using H.263 can different from 64kbps to 2Mbps. and video stream doesn’t have silence gap as the voice stream, it is consecutive.

The codec mechanisms of video is different from that of voice, if one frame is lost or received with errors, the error will extend until the next I frame comes. So we can’t discard frames like voice does.

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 will obviously infect the voice communication [3]. For video,
large experiments shows the end-end delay can be withstand between 200-500ms. for television quality video, play out jitter should be kept below 10ms, and can be withstand below 50ms in lower quality video. The skew is required to below 80ms for inter-stream synchronization.[8] So, for applications with certain QoS requirements, we can give out:

\[ P(|J_{a}| > J_{\text{max}}) < \varepsilon_{a} \]
\[ P(|J_{a} - J_{i}| > S_{\text{max}}) < \varepsilon_{s} \]
\[ P(|J_{i}| > J_{\text{max}}) < \varepsilon_{v} \]

Here \( J_{a} \) is the jitter of the voice stream, and because the voice stream contains more information than the video stream, so the importance of the above three equation is from the most important to the least important.

IV. ALGORITHM DESCRIPTION

Our adjust algorithm are different in different network situation.

1) Collect the network condition and decide the system mode. If in normal mode, go to step 2, else to step 3;
2) Calculate delay for voice buffer if at the beginning of a talk spurt, then adjust video buffer to synchronize with it, go to step 1;
3) Calculate delay for voice packets, try to reduce jitter and move artificial jitter to the silence gaps and vad frames, go to step 4;
4) Adjust video buffer to synchronize with the voice, reduce the skew and avoid packet loss, go to step 1;

A. Adjust methods for voice in normal mode

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;

\[ v_{i} = D_{\text{prop}} + D_{\text{trans}}, \] 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 normal mode, because the network delay isn’t large, we can found a delay bound \( b \) below 150ms, which all packets’ delay won’t beyond. All the packets be play out by the delay bound, thus the artificial jitter caused by adaptive adjust is avoid, without affecting the quality of voice and video. 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 voice buffer only be adjusted at the beginning of talk spurt, and make the change in the silence gap.

B. Adjust methods for video in normal mode

If the adjustment of voice buffer make the skew of the voice and the video beyond a predefined bound(for an example: 20ms), then the video buffer will start adjustment to synchronize with voice buffer. Other wise, the video buffer will not act when voice buffer adaptively adjust.

C. Adjust methods for voice in spike mode

In spike mode, there are two different ways to adjust buffer for every packet.

1) Prediction based adjust algorithm:

One way 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.

NLMS prediction algorithm is chosen. 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.

The flow chart of the algorithm is show in Fig 3. And the formulation of the algorithm is:[7]

\[ \hat{d}_{i} = w_{i}^{T} \cdot \bar{u}_{i} \]
\[ e_{i} = d_{i} - \hat{d}_{i} \]
\[ \bar{w}_{i+1} = \bar{w}_{i} + \frac{\mu}{w_{i}^{T} \cdot w_{i} + \alpha} \cdot \bar{u}_{i} \cdot e_{i} \]

Here \( \bar{u}_{i} = \{d_{i-1}, d_{i-2}, d_{i-3}, \ldots, d_{i-k}\} \) is the input of the algorithm, \( d_{i} \) 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 \( \bar{u}_{i} \) 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 \( \bar{u}_{i} \). \( \bar{w}_{i} = \{w_{1}, w_{2}, \ldots, w_{k}\} \) 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 \( \bar{w}_{i} \), the size and positive or negative of the error will influent the change of the \( \bar{w}_{i} \). Coefficient \( \mu \) decides the degree of adjustment. Large \( \mu \) will induced to quickly change of the \( \bar{w}_{i} \). 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.

2). Prearranged adjust algorithm:

The other way is to adjust the buffer size according to the prearranged way: stepped increase and logarithmic decrease. As show by Fig 4, just the TCP sliding window flow control.

![Fig. 4. Prearranged buffer adjust method](image)

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 will 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.

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.

D. Adjust methods for video in spike mode

For video buffer, we can set a skew band: \( S_{up} \), which is below the \( S_{max} \). If the skew beyond the band, then the video buffer will adaptively adjust to synchronize them. So the skew beyond \( S_{max} \) is avoided.

The video buffer adjust method is to change the play out rate of the video frame. When the buffer size is need to be increase, then slower the play out rate for several, otherwise, accelerate the play out rate. If the rate adjust is below 10%, human being generally can’t detect.

V. SIMULATIONS AND RESULTS

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 5. Here we only show the TCP-like performance of the simulation. It is more sensitive when the jitter occurs, but the int-NLMS prediction strategy trail the variety of the delay better. Both of them avoid the packets loss and still trade off the packets loss and the packets delay and jitter in spike mode.

![Fig. 5. Comparison of two kinds of adaptive buffer adjust method](image)

VI. EXPERIMENT AND RESULTS

An experiment system is set up to test our algorithm. 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 any V2oIP equipment with adaptive buffer algorithm.

One scenario is that one end equipment 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 communicate with each other, as show by Fig 7.

Another scenario is that two end equipment access through different WLAN BSS, and they set up communication with the help of SIP server through a gateway, as show by Fig 7.
Experiments have been done to test the performance of the adaptive buffer adjust methods. Fig 8 shows their average performance after a long time experiments 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 voice 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 show the variance of video jitter, the two algorithm is almost the same. The third and fourth sub figure shows mean delay of voice and video in spike mode. It shows both of the two kinds of adjust algorithm will improve the average delay a little. The fifth sub figure shows packets loss rate in spike mode. It shows prearranged buffer adjust make the data can react to the spike quickly and save more delayed packets. The sixth sub figure shows the skew, the NLMS algorithms reduce the skew more. The seventh sub figure shows 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 V^2oIP application. This paper gives out an adaptive buffer adjust algorithm which combined network conditions and media characters. The simulation and experiment results prove that it can observably reduce the jitter, skew 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, delay jitter and skew.

ACKNOWLEDGMENT

The authors would like to express their sincere thanks to Hitachi R&D Headquarter for the continuous supports and cooperations in this work.

REFERENCES