COMPLEXITY-AWARE ADAPTIVE JITTER BUFFER WITH TIME-SCALING

Liyun Pang[1], *Anisse Taleb*[1], *Jianfeng Xu*[1] *and Laszlo Böszörmenyi*[2]

[1] Huawei European Research Center, Munich, Germany
{liyun.pang, anisse.taleb, antonio.xujianfeng}@huawei.com
[2] Department of Information Technology, University Klagenfurt, Klagenfurt, Austria
office-lb@itec.uni-klu.ac.at

ABSTRACT

In VoIP applications, packet loss, delay and delay jitter are inevitable and have a large impact on the perceived speech quality. Jitter buffers are commonly deployed to compensate for jitter in order to play out the received packets continuously. For mobile devices, due to limited battery power, computational complexity has to be kept to a minimum. In this paper, we propose a jitter buffer management which takes complexity into consideration. The algorithm adaptively adjusts the play-out delay under certain complexity constraints. Simulation results show that our proposed algorithm can keep complexity under specified constraints while still optimizing jitter induced packet losses and average delay.

Index Terms— Adaptive jitter buffer, time-scaling, computational complexity

1. INTRODUCTION

Packet-switched networks such as local area networks LANs or the Internet, are used to carry speech, audio, video or other continuous signals, such as Internet telephony or audio/video conferencing signals and audiovisual streaming such as IPTV. However, communications in packet-switched network has several challenges such as latency, delay jitter and packet losses. In a VoIP application, the sender transmits a series of packets over the network to the receiver at regular time intervals. These packets encounter variable and unpredictable propagation delays over the network mainly due to network congestion, improper queuing, or configuration errors. This results in the so-called delay jitter where packets arrive at the receiver with variable and usually unpredictable inter-arrival time or even out of order. An adaptive jitter buffer is usually deployed at the receiver in order to compensate for the effects of jitter.

In a jitter buffer, packets are buffered for a certain time after arrival. At predefined times, each packet is sent to the source decoder and thereafter played out. Using time-scale modification, the length of a decoded speech frame can be changed at the receiver adaptively in response to the jitter buffer to ensure continuous play-out [1-3]. Such time-scale modification can either be external or internally integrated in the decoder [4].

Many algorithms for achieving good performance of jitter buffering have been reported in the literature. Most algorithms are either based on achieving a trade-off between late packet losses and buffering delay [1] or maximizing the perceived quality [3]. However, to the best knowledge of the authors, little work has been done to address the issue of the instantaneous power consumption in jitter buffer management algorithms. This is especially true for handheld devices which have limited battery lifetime. In [4], it is pointed out that jitter buffer management could affect the computational complexity at the receiver, and a possible solution is proposed which might reduce the complexity, however, at the expense of large delay.

In this paper, we propose an adaptive jitter buffer with a complexity control mechanism. Furthermore, we explore the relationship between a complexity control parameter and the resulting delay statistics. The paper is organized as follows. A brief introduction of jitter buffer management is provided in Section 2. The computational complexity of the receiver is introduced in Section 3. Section 4 describes the proposed complexity-aware jitter buffer management to control the peak complexity, and also keep the average complexity low. Experimental results illustrating the performance of the proposal are presented in Section 5. Conclusions and insight into future extensions are provided in Section 6.

2. JITTER BUFFER MANAGEMENT

A Jitter buffer management system located at the receiver usually includes the actual jitter buffer, an Adaptation Control logic and optionally a Time Scaling, as illustrated in Fig. 1. Once the jitter buffer contains several packets, it begins sending the packets to the decoder at a certain rate based on the buffer status and current network conditions.

In the following, for the sake of simplicity, we will assume that each packet contains a single speech coded frame, so "packet" and "frame" are used interchangeably in this paper. Extensions to cases where a packet might contain more than one frame are easily derived.

When packet i is received and unpacked, the jitter buffer management analyzes the current network condition

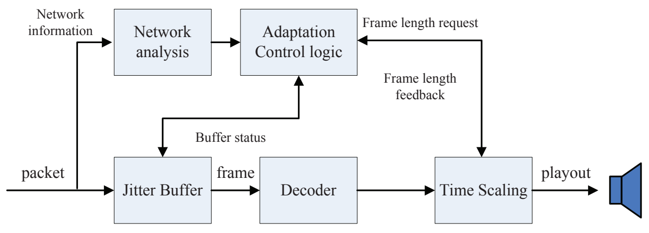

Fig. 1. Jitter Buffer Management with Time-Scaling

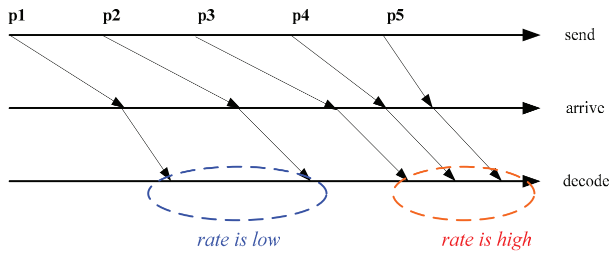

Fig.2. Processing rate of variable frame length

based on packet header information and puts frame i in the jitter buffer. In the Adaptation Control logic, the play-out time of packet i+1 is estimated before its arrival. Frame i is sent to the decoder at its scheduled play-out time. Since the frame length of frame i is the difference between $\hat{d}_p^{i+1}$ the estimated play-out delay of packet i+1 and d_p^i the play-out time of packet i, the estimated play-out time is converted to the expected frame length of i, as

$$L^i = \hat{d}_p^{i+1} - d_p^i \tag{1}$$

Reducing the frame length of i means decreasing the play-out time of packet i+1. The frame length modification request is thus sent to the Time Scaling unit. Based on the request, the decoded samples are stretched or compressed accordingly. The actual time-scaled length is fed back to the control logic. The signal obtained at the output of the Time Scaling unit is ready to be consumed and played out.

3. TIME SCALING AND ITS IMPACT ON COMPUTATIONAL COMPLEXITY

In many applications, computational complexity is a key parameter to take into account to ensure good performance and correct platform dimensioning. It is directly related to the CPU load when running the decoder, and therefore it is always desirable to limit the worst-case complexity. The worst-case complexity measures the resources of an algorithm required in the worst case. The average complexity is also an important parameter as it relates to the energy consumption of the processor and is strongly correlated to the battery lifetime of the handheld device.

Complexity can be measured in terms of wMOPS (Weighted Millions of Operations per Second) or MIPS (Millions of Instructions per Second). The wMOPS value is typically obtained from the ITU-T simulation software [5] in terms of basic arithmetic operations, and these operations are then weighted to estimate the complexity.

In our context, the computational complexity of decoding frame i is expressed as

$$C^i = NumOP^i / L^i \tag{2}$$

where $NumOP^i$ is the number of operations that the speech decoder performs to decode the given frame i, and L^i is the duration of frame i, which is normally constant, and typically 20ms in most currently deployed speech codecs. While this definition is accurate when expressing the complexity of a speech codec alone, it fails in reflecting the actual load on the processor.

In fact, since complexity C^i and frame length L^i are inversely proportional, when using time-scaling the length of the decoded speech frame L^i will vary accordingly. When no time-scaling is applied, the frame length is constant and is therefore consumed from the jitter buffer to the decoder at a constant rate, for instance, one frame every 20ms. When time-scaling is used to change the frame length, the speech decoding processing rate is variable. We illustrate this with a time-line example in Fig. 2:

1) time-stretching (shown in blue) renders frames taken from the jitter buffer and decoded at a lower rate thus CPU load and complexity are decreased;

2) time-compressing (shown in red) renders frames taken from the jitter buffer and decoded at a higher rate, so the CPU load and the complexity are increased.

In conclusion, when time-scaling is used in jitter buffer management, the actual frame length of the output signal will be changed and the worst-case complexity will be increased due to shorter frame. Therefore, this worst-case complexity should be controlled; otherwise the CPU can be overloaded and leads to undesirable effects such as a loss of synchronicity which would lead in the case of voice signals to annoying clicks in the perceived quality.

4. COMPLEXITY-AWARE JITTER BUFFER MANAGEMENT

To avoid the problem of complexity overload described in Section 3, the proposed complexity-aware jitter buffer management takes the complexity information into account before sending the time-scaling request. As shown in Fig. 3, two additional complexity parameters are needed by the Adaptation Control logic unit. The first parameter, denoted as C_{max}, represents a complexity upper bound, i.e. the maximum acceptable complexity. This parameter is externally supplied and would depend on the processing power of the target device. The algorithm would strive to always be below this worst-case complexity.

The second parameter $\hat{C}_{DEC}^i$ is the estimated complexity of the source decoder corresponding to frame i. There are

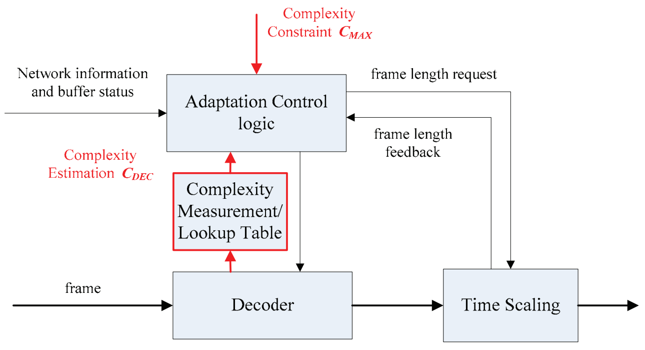
Complexity Constraint C_{MAX}
Network information and buffer status
Adaptation Control logic
frame length request
frame length feedback
Complexity Estimation C_{DEC}
Complexity Measurement/ Lookup Table
frame
Decoder
Time Scaling

Fig. 3. Complexity-Aware Jitter Buffer Management

many alternatives to obtain such an estimate. For instance, one could use a lookup table which stores related worst-case complexity values based on an offline characterization of the codec corresponding to different sampling rates, bitrates and in general to various operating points of the codec. A more accurate approach is to count, during run-time, the number of operations used by the processor for decoding a frame. This is similar to the use of profiling tools such as the wMOPS tools [5] to obtain the estimated complexity per frame. While this approach is more accurate, the counting of the number of operations itself can introduce additional complexity. In the remainder of this paper, we will make use of the wMOPS tools to validate the proposed algorithm and simulation results. Other complexity estimation methods are currently being envisioned for future works.

Based on these two parameters, $\hat{C}_{DEC}^{i}$ and C_{max}, the Adaptation Control logic unit includes an additional complexity control part, the processing of which is as follows:

First the complexity of the decoder including jitter buffer management C^i is defined as

$$C^i = \left(NumOP_{DEC}^{i} + NumOP_{TS}^{i}\right)/L^i \tag{3}$$

where $NumOP_{DEC}^{i}$is the number of operations in the source decoder implemented for frame i, and $NumOP_{TS}^{i}$ is the number of operations in the Time Scaling implemented for frame i, which is a rather constant and small value. L^i is the expected frame length based on network information and buffer status analysis. We further define

$$C_{DEC}^{i} = NumOP_{DEC}^{i}/L_o \tag{4}$$
$$C_{TS}^{i} = NumOP_{TS}^{i}/L_o \tag{5}$$

where L_o is the original frame duration, e.g. 20ms, and C_{DEC}^{i} and C_{TS}^{i} are the complexity values for frame i over L_o. We assume $C_{DEC}^{i} \approx \hat{C}_{DEC}^{i}$ and use a constant value C_{TS} obtained from simulation to represent C_{TS}^{i}, then (2) is converted to

$$C^i = (\hat{C}_{DEC}^{i} + C_{TS})\, L_o/L^i \tag{6}$$

The target of the adaptation unit is to limit the complexity load according to the following constraint:

$$C^i \le C_{max} \tag{7}$$

Therefore, L^i must fulfill:

$$L^i \ge L_{min} = \frac{\hat{C}_{DEC}^{i}\, L_o}{C_{max} - C_{TS}} \tag{8}$$

Then the expected length request $min\,(L^i, L_{min})$ is sent to the Time Scaling. By controlling that the length of modified voiced frames never goes below L_{min} the worst-case complexity will be ensured not to exceed the externally set computing ability of the device. This makes the adaptive jitter buffer more conservative when it comes to decreasing the play-out delay.

5. EVALUATION AND DISCUSSION

In the experiments conducted four algorithms have been implemented: the algorithm mentioned in [4] to compress frames only during silence as *Alg.1*, the proposed algorithm with complexity control using wMOPS tools, referred to as *Alg.2,* the proposed algorithm with complexity control using lookup table, referred to as *Alg.3*, and the algorithm without complexity control as *Alg.4*. For the jitter buffer adaptation and time-scaling, we implemented an algorithm similar to that reported in [1]. It should be noted that our framework, illustrated in Fig. 3, is quite general, and can accommodate various jitter buffer models with time-scaling as well as various speech codecs. In this evaluation, the AMR-WB speech codec [6] operated at the 12.65kbps mode was used. The five trace files from [7], profile 1-4 and profile 6, each containing 7500 packets or frames, were investigated. Profile 5 is not used because it is for two frames per packet. Due to space limitation, we mainly represented the results of profile 6 here. Profile 6 has the largest jitter among all the files, and thus represents the worst case scenario for a jitter buffer management. Similar results could be derived for all of them.

5.1. Complexity Comparison

Table 1 shows the results of worst-case complexity, average complexity, late loss rate and average buffering delay of four algorithms for profile 6. *Alg.1* has the lowest peak complexity, since no compression is used in active speech, thus almost no additional complexity is required. However, as is clearly seen, this comes at the expense of larger buffering delay, this will be further illustrated in Section 5.3. Our proposed complexity-aware jitter buffer management algorithms are denoted as *Alg.2* and *Alg.3*, and we use 12 wMOPS as C_{max} for both. The results obtained are quite close, showing that the table lookup approach was quite efficient in determining the complexity of each frame when compared to the more accurate use of wMOPS tools.

TABLE 1. Complexity for profile 6

	worst wMOPS	avg. wMOPS	Late_loss Rate (%)	Avg. buffering delay(ms)
Alg.1	7	6.4	0.70	59
Alg.2	12	6.4	0.82	39
Alg.3	12	6.4	0.83	40
Alg.4	14	6.4	0.84	38

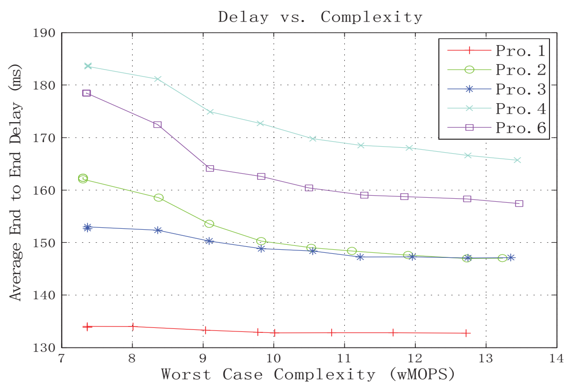

Fig. 4. End-to-End delay compared with complexity

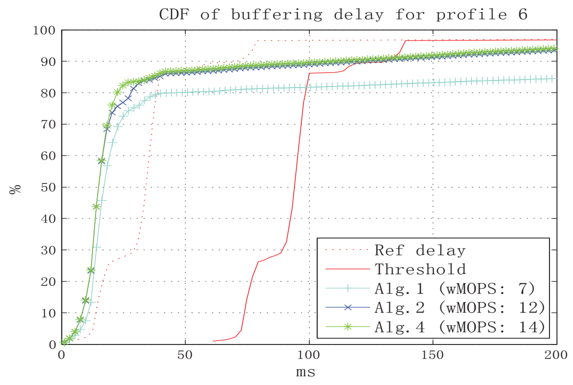

Fig. 5. CDF of buffering delay for profile 6

Alg.4 is the jitter buffer management without complexity control mechanism, and it has the highest observed worst-case complexity. The average complexity values of the four algorithms are almost identical, and this is due to the fact that although compression increases complexity, stretching which decreases complexity will cancel this resulting in a similar average complexity.

5.2. Delay and Complexity (*Alg.2*)

Fig. 4 shows the relationship between the average end-to-end delay and the worst-case complexity in *Alg.2* by varying the value of C_{max}. For all the five trace files, the average end-to-end delay can be reduced at the expense of additional complexity. Among these files, profile 1 has zero packet loss in the network and relatively small jitter when compared to other four trace files, therefore, the benefit of increasing complexity on the delay is not apparent, since the average end-to-end delay is already very small.

5.3 Cumulative Distribution Function of buffering delay

Fig. 5 shows the CDF of buffering delay for profile 6. It is a requirement in [7] that at least 90% of the buffering delay must be below the delay threshold (the red line). Profile 6 has also zero network packet loss, but very large delay spikes (around 300 ms). It is shown that both *Alg.2* and *Alg.4* can fulfill the requirement. *Alg.1* can only reduce the play-out delay in silence, thus it cannot reduce the delay under the threshold and then is unable to fulfill the CDF requirement. All of these algorithms have a late loss rate lower than 1% which is in line with the requirements [7].

6. CONCLUSION AND FUTURE WORK

We have proposed a jitter buffer management taking the complexity information into account, and controlling the worst-case complexity under specified constraints. We show that it is possible to reduce complexity of the overall system while still fulfilling the requirements on jitter buffer management specified by 3GPP in [7].

For future work, it is envisioned to use other complexity measurement tools instead of the wMOPS. Furthermore, in order to consider the overall perceived speech quality, a combination of this approach with a quality based jitter buffer management algorithm [3] is also a topic of investigation.

7. REFERENCES

[1] Y.J. Liang, N. Färber, and B. Girod, "Adaptive Playout scheduling using time-scale modification in packet voice communications," in *Proc. ICASSP'01*, pp. 1445-1448, May 2001.

[2] G. Zhang, H. Lundin, and W.B. Kleijn, "Band Control Policy of Playout Scheduling for Voice over IP," in *Proc. EUSIPCO'08*, August 2008.

[3] L. Pang and L. Böszörmenyi, "E-Model based Adaptive Jitter Buffer with Time-Scaling Embedded in AMR decoder," in *Proc. ICDT'11*, pp. 80-85, April 2011.

[4] P. Gournay and K.D. Anderson, "Performance analysis of a decoder-based time scaling algorithm for variable jitter buffering of speech over packet networks," in *Proc. ICASSP'06*, pp. I-17-I-20, March 2006.

[5] ITU-T G.191, ITU-T Software Library Tools 2009 User's Manual, November 2009.

[6] 3GPP TR 26.976, Performance characterization of the Adaptive Multi-Rate Wideband (AMR-WB) speech codec, April 2011.

[7] 3GPP TS 26.114, IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction, June 2011.