A Black-box QoS Measurement Methodology for VoIP End-points

Wenyu Jiang, Henning Schulzrinne
Department of Computer Science, Columbia University
{wenyu,hgs}@cs.columbia.edu

ABSTRACT

We present a black-box based measurement methodology for evaluating the QoS of VoIP end-points (IP phones and soft-phones). We discuss how this general methodology can be coupled with novel input stimuli to measure a number of important end-point QoS metrics, such as mouth-to-ear (M2E) delay under WAN conditions, the playout delay and inference of playout algorithm behaviors. We then illustrate its use and efficacy with real measurement cases and results.

1. INTRODUCTION

Voice over IP (VoIP) provides many benefits, including communication cost savings and faster development of new services [8]. The performance, particularly the Quality of Service (QoS) of VoIP end-points (IP phones or soft-phones) is crucial in the implementation of a VoIP service, because the end-users will assume a new communication service with IP is at least comparable to today's public switched telephone network (PSTN). This assumption will be a big decision factor in whether the users will adopt VoIP.

The QoS metrics for a VoIP end-point include mouth-to-ear (M2E) delay, silence detection behavior, clock skew, packet loss concealment robustness, to name a few. Most research advances in QoS, such as IntServ, DiffServ, MPLS, constraint based routing [9], have focused on measuring and improving the network. In contrast, few papers have addressed end-point related behaviors, such as M2E delay in real applications. Good network QoS is not trivial to achieve, but it can be easily undone by poor end-point QoS.

Previously [4], we evaluated a number of end-points in terms of the above metrics. In this paper, we generalize our measurement methodology. Our major contributions are: first, our methodology naturally spawns new QoS metrics. For example, by measuring M2E delay trends, one can observe clock skew and playout delay adjustment behavior. Second, it enables us to extract important properties of an end-point, such as the playout delay and playout algorithm [5, 7] behavior using novel techniques and stimuli even under a black-box setting. Black-box measurement is indispensable, particularly in a *third-party independent* evaluation and comparison of VoIP products. Finally, using our methodology, we made many interesting observations and explored some unanswered questions in our previous work. For example, we found an anomaly in rat's delay adjustment that leads to ever-increasing M2E delay (Figure 4(a)), and we have traced this to the existence of dual oscillators in an end-point we suspected in [4]. These observations could help substantially in finding problems in existing products and improving upon them.

The remainder of this paper is organized as follows. In Section 2, we present the general methodology for measuring VoIP end-point QoS, and highlight metrics that fall well into our approach. In Section 3, we describe a case study of our methodology, by applying it to real world VoIP products, and emphasizing on interesting discoveries during this application. Finally, we conclude the paper in Section 4.

2. MEASUREMENT METHODOLOGY

M2E delay is probably the most fundamental in VoIP end-point QoS metrics. In [4], we devised a simple and effective solution, by capturing both original and output audio signals, and recording them in stereo mode, as shown in Figure 1. Later, we can derive M2E delay by correlating and finding the delay offset between the two channels. We developed a software program called adelay that automates delay estimation (http://www.cs.columbia.edu/~wenyu/research/adelay/), thus facilitating long duration measurements and making the observation of long term delay trends a simple task. In comparison, Calyam [1] and Hagsand *et al.*[2] both use some signal generator (e.g., metronomes) as an input source and estimate M2E delay manually. We also study far more end-point characteristics.

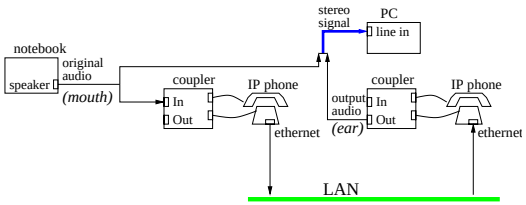


Figure 1: Measuring mouth-to-ear (M2E) delay

M2E delay measurement can spawn other types of measurements. For example, by performing linear regression on M2E delay trends, one can measure clock skew. It may also help track down misbehaviors that otherwise go unnoticed with a short test duration.

2.1 M2E Delay under Jitter (over WAN)

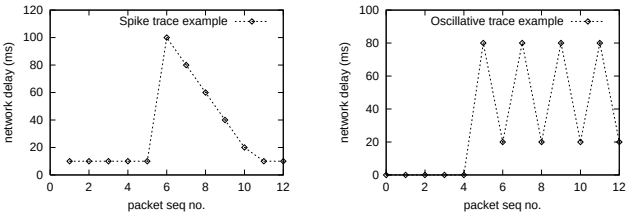
To simulate Internet or WAN behavior, we can add a UDP relay in Figure 1 between two end-points, and let the relay process insert loss, delay and jitter accordingly, for example based on a packet trace. The trace can thus serve as a special stimulus for extracting end-point properties. Previously [4] we used deterministic loss patterns as the stimuli to study packet loss concealment (PLC) [6] robustness. Here we will examine the receiver's playout algorithm adaptiveness to delay and jitter, using special types of delay traces.

2.1.1 Benchmark Delay Traces

The first benchmark trace is a delay spike, that is, a sudden increase in delay, followed a sharp decrease. The receiver will see a sudden stop in audio packets, and then many packets almost back to back, as shown in Figure 2(a). According to Moon [5] and Ramjee [7], spikes are relatively common. The spike trace tests how well a receiver adapts to sudden changes in delay.

The second benchmark is a trace with oscillative delays. For example, in Figure 2(b), it oscillates between 20 ms (d_{lo}) and 80 ms (d_{hi}). This stimulus tests the playout algorithm's

intelligence. Exponential Average (Exp-Avg) [7], a common playout algorithm, predicts playout delay D_p (i.e., buffering delay) as $4 \cdot RTP\_jitter$. Denoting D_{all} as the sum of average network delay and playout delay, we have:



(a) delay spike (b) oscillative delay

Figure 2: Illustration of benchmark traces

$$D_{all}(\text{Exp-Avg}) = (d_{hi} + d_{lo})/2 + 4 \cdot (d_{hi} - d_{lo})/2$$
$$= 2.5d_{hi} - 1.5d_{lo}$$
$$D_{all}(\text{optimal}) = d_{hi},$$
$$\text{overhead} = D_{all}(\text{Exp-Avg}) - D_{all}(\text{optimal})$$
$$= 1.5(d_{hi} - d_{lo})$$

So Exp-Avg will use an overall delay of $50 + 4 \times 30 = 170$ ms, even though we really need only 80 ms. By observing M2E delay increase with this stimulus compared to LAN conditions, we can tell whether the receiver uses an Exp-Avg style of playout algorithm.

2.1.2 Time Collation of Trace and M2E Delay Curve

To accurately depict how a packet trace affects the resulting M2E delay curve, we need to time-align (collate) them. Therefore, we need to know for each packet i, which time-point it corresponds to in the stereo wav file recorded in Figure 1. However, because the recording action is done manually, this time-alignment offset is unknown.

We have found a simple yet effective solution to determine this offset. All we need is to have the UDP relay log its received packets. Later, we use a special program to reconstruct the audio waveform by extracting RTP payload from the log file. Then, by comparing the delay offset between the RTP-generated wav file (which has a static timing association with the packet trace) and the left channel of the stereo recording in Figure 1, one can perfectly collate the trace and M2E delay curve.

2.2 Playout Delay

For an IP phone, probably nothing could be more mysterious than its playout delay. Although dynamic, its value should be relatively stable under LAN conditions yet hidden to black-box testers. Naturally, our idea would be to create a situation in which the receiver is forced to reduce its playout delay to nearly zero, even if only temporarily. This led to the estimation algorithm shown in Figure 3.

2.2.1 Using Sudden Large Delay Step Function

If there is a sudden, fixed increase in delay, say by 200 ms as in Figure 3, and suppose the current playout delay is 30 ms, then the receiver will activate PLC. After a few packets PLC will fade out waveforms, and finally when the next packet arrives, assuming the receiver will immediately playback this packet since it was long overdue, there will be a 200-30=170 ms difference (T' in Figure 3) between when PLC is activated and when the delayed waveforms appear. Therefore, by measuring T', one can infer the playout delay.

2.2.2 Using Incremental Small Delay Step Functions

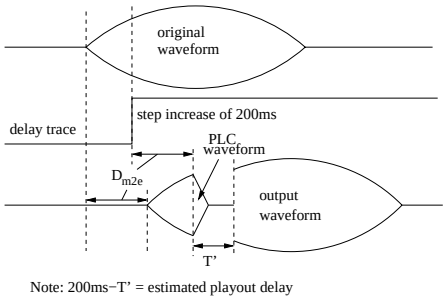


Figure 3: Playout delay extraction using high steps

We assumed the playback of the next long overdue packet is immediate. But, a conservative receiver may choose to buffer even this packet in fear of further delay increase. In that case, we will need a more general solution.

We later came up with a simple yet novel technique, based on the fundamental concept of playout delay. If we introduce a step delay increase slightly smaller than the playout delay, any well-behaving receiver should be able to absorb it without creating any distortion or PLC-induced elongation in the output waveform. But this will no longer be the case if the increase slightly exceeds the playout delay, because PLC will start to activate. So we can use a series of small step functions with incremental heights, and check at which height PLC starts to activate. Then the previous step size would be the closest match for playout delay.

3. CASE STUDY

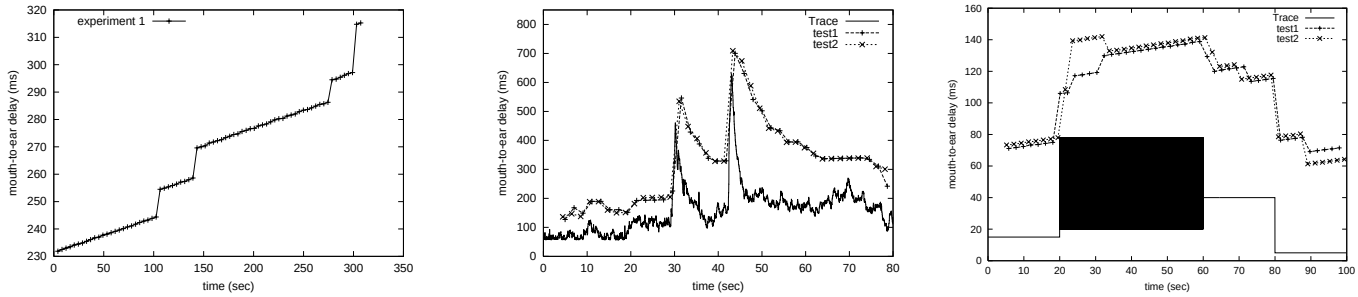
We now illustrate the application of our methodology to real products, and reveal many interesting observations. For brevity, please refer to [4] for the list of end-points.

3.1 M2E Delay Comparisons

In Figure 4(a), the rat client (running on a AMD K7 PC running Windows XP, dubbed K7-XP) apparently adjusts the delay upward even when clock skew causes M2E delay to linearly increase. These delay jumps are certainly mysterious, if not unexplainable. Here we did not use silence suppression, so it could not have been caused by any misinterpretation of RTP timestamps and/or sequence numbers. Using tcpdump, we also verified that there was no duplicate packet either. Playout buffer underflow is extremely unlikely since the buffer was accumulating rather than drying up, let alone the test PC was lightly loaded.

Because this anomaly is doing exactly the opposite of what clock skew compensation is supposed to, we suspect it is caused by incorrect skew detection. It is also more than coincidental that this PC appears to have two crystal oscillators after repeated tests [4], with recording skew ≈165 ppm (faster than UTC) and playback skew ≈-82 ppm. The 3Com phone's skew is ≈46 ppm.

Now, the mystery of anomaly in Figure 4(a) starts to unveil itself. If a skew detection algorithm assumes only one audio clock per sound card, and if it uses the audio-send (i.e., recording) clock to compare that of its peering end-point, it will draw the wrong conclusion. We confirmed the existence of this assumption from [3] and by inspecting the source code of rat. Here the skew of 3Com to rat is 46-(-82)=128 ppm, which matches well with the slope in Figure 4(a). But rat thinks the relative skew is 46-165=-119 ppm, which would surely cause its skew detection algorithm to go haywire and adjust in the wrong direction. Hence these algorithms must



(a) Delay trend anomaly: 3Com→rat(K7-XP) (b) rat → Cisco, trace with 2 delay spikes (c) rat → Cisco, trace with oscillative delay

Figure 4: M2E delay trends and WAN behaviors

assume worst case of two oscillators per circuit, to detect and adjust for skew correctly.

3.2 WAN Behavior

For WAN simulations, we first used some real traces we collected earlier. Figure 4(b) shows that the Cisco phone follows the trace trend well, with a small lag in responding to spikes. After a spike, it is conservative by slowly lowering the delay. This is desirable behavior, since another spike or increase in network delay could potentially come again.

Next, we tested how the Cisco phone reacts to oscillative delay. Figure 4(c) shows the result. The big, dark region corresponds to a delay oscillating between 20 ms and 78 ms, and initial delay was 15 ms. Using equations from Section 2.1, we know Exp-Avg will use an overall delay of $2.5 \times 78 - 1.5 \times 20 = 165$ ms, although in reality it need only be 78 ms, and Exp-Avg would have inserted 165-78 = 87 ms of unnecessary delay.

Fortunately, in Figure 4(c), the Cisco phone does not overshoot. Its M2E delay increased approximately from 80 ms to 140 ms (60 ms increase), which matches well with the network delay increase 78-15 = 63 ms. Therefore, its playout algorithm is not Exp-Avg. We also repeated the same experiment on the 3Com phone and rat. They all have behaviors similar to the Cisco phone.

3.3 Playout Delay

In Section 2.2 we presented two methods of inferring playout delay. Table 1 summarizes a test using the first approach, which uses high delay step functions. Here we tested three IP phones. For the Polycom phone, the results are fairly consistent, between 30 ms and 40 ms. The results for the 3Com phone however, vary more significantly, from 9 ms to 28 ms. Finally, the Cisco phone's results are almost unbelievable, with three measurements indicating 0 ms playout delay. As explained in Section 2.2, a 0 ms result indicates the Cisco phone was using the same (non-zero) playout delay before and after the delay step-increased.

Receiving end-point	estimated playout delay D_p			
	test 1	test 2	test 3	test 4
Polycom	40 ms	30 ms	40 ms	39 ms
3Com	12 ms	28 ms	9 ms	9 ms
Cisco	0 ms	0 ms	0 ms	10 ms

Table 1: Measured playout delay using delay step functions, step size varies from 120 ms to 300 ms

Then using the second and more general method, we got the results of Table 2. The Polycom phone can sustain only 30 ms of delay increase without introducing distortions (PLC artifacts) into the output waveform. Although less than the 40 ms result previously attained, they are not contradictory because their corresponding M2E delays before the step increase were different. For example, in test 1 of Table 1, the Polycom phone's previous M2E delay was 100 ms, whereas in Table 2 in all cases (the table only provides a summary) the Polycom phone's previous M2E delay was around 90 ms.

Receiving end-point	estimated playout delay D_p	$D_{bare} = D_{m2e} - D_p$
Polycom	30 ms	60 ms
3Com	20 ms	35 ms
Cisco	15 ms	48 ms
rat (Ultra-10)	40 ms	200 ms

Table 2: Measured playout delay D_p and bare minimum sustainable delay D_{bare} using small steps

Hence, by subtracting the M2E delay (D_{m2e}) with D_p, we obtain the "bare" minimum M2E delay that an end-point can sustain (at least temporarily) without introducing waveform distortions. We denote this as D_{bare}. The 3Com phone has the lowest D_{bare} in Table 2, at 35 ms, so it only has 15 ms left after subtracting D_{bare} with the 20 ms packet interval. This is a very small and desirable value. In contrast, D_{bare} of rat is 200 ms. The main cause for such a high delay has to be rat itself, and most likely the sound API it uses, and the sound card output buffer size. Hagsand *et al.*[2] provide some suggestions on achieving low output delay for Windows-based sound API.

4. CONCLUSIONS AND FUTURE WORK

We presented a general methodology for measuring a wide variety of VoIP end-point QoS metrics that naturally spawn from mouth-to-ear (M2E) delay measurements. Long-term observation helps detect less-than-ideal trends or sometimes anomalous behaviors (like the skew compensation anomaly of rat in presence of two crystal oscillators per audio circuit). Using a UDP relay program and the right trace stimuli, we can infer many hidden receiver properties, such as PLC robustness, playout algorithm behavior (e.g., whether it is Exp-Avg like), and the often mysterious playout delay.

We have applied our methodology to the testing of many VoIP end-points. We believe our results reveal interesting observations and provide insights to the betterment of VoIP products. For example, the extraction of playout delay, along with the corresponding M2E delay, gives clue to how much delay overhead is in playout buffering, and how much else in other aspects (e.g., sound API), and thus reveals the true bottleneck in a VoIP end-point system.

We plan to implement an integrated and automated VoIP end-point QoS measurement system using the general methodology and various techniques we developed in this paper.

5. REFERENCES

[1] Calyam, A. P. Performance measurement and analysis of H.323 videoconference traffic. Master's thesis, Ohio State, 2002.

[2] Hagsand, O., Marsh, I., and Hanson, K. Sicsophone: A low-delay Internet telephony tool. Tech. Rep. T2002:26, Swedish Institute of Computer Science, Dec. 2002.

[3] Hudson, O., Perkins, C., and Hardman, V. Skew detection and compensation for internet audio applications. In *IEEE International Conference on Multimedia and Expo* (July 2000).

[4] Jiang, W., Koguchi, K., and Schulzrinne, H. QoS evaluation of VoIP end-points. In *Conference Record of the International Conference on Communications (ICC)* (May 2003).

[5] Moon, S., Kurose, J. F., and Towsley, D. F. Packet audio playout delay adjustment algorithms: performance bounds and algorithms. Research report, Department of Computer Science, University of Massachusetts at Amherst, Amherst, Massachusetts, Aug. 1995.

[6] Perkins, C. E., Hodson, O., and Hardman, V. J. A survey of packet loss recovery techniques for streaming audio. *IEEE Network 12*, 5 (Sept. 1998), 40–48.

[7] Ramjee, R., Kurose, J. F., Towsley, D. F., and Schulzrinne, H. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)* (Toronto, Canada, June 1994), IEEE Computer Society Press, Los Alamitos, California, pp. 680–688.

[8] Rosenberg, J., and Schulzrinne, H. The IETF Internet telephony architecture and protocols. *IEEE Network 13*, 3 (May/June 1999), 18–23.

[9] Xiao, X., and Ni, L. M. Internet qos: A big picture. *IEEE Network 13*, 2 (1999), 8–18.