2009 11th IEEE International Symposium on Multimedia

# vDelay: A Tool to Measure Capture-to-Display Latency and Frame Rate

Omer Boyaci, Andrea Forte, Salman Abdul Baset, and Henning Schulzrinne
Department of Computer Science
Columbia University, New York, USA
{boyaci,andreaf,salman,hgs}@cs.columbia.edu

## ABSTRACT

We present *vDelay*, a tool for measuring the capture-to-display latency (CDL) and frame-rate of real-time video applications such as video chat and conferencing. vDelay allows measuring CDL and frame-rate without modifying the source code of these applications. Further, it does not require any specialized hardware. We have used vDelay to measure the CDL and frame-rate of popular video chat applications such as Skype, Windows Live Messenger, and GMail video chat. vDelay can also be used to measure the CDL and frame-rate of these applications in the presence of bandwidth variations.

## 1. INTRODUCTION

Real-time video chat applications augment the communication experience of participants by allowing them to see other participants in addition to having an audio conversation. These applications have three key software components: a video encoder that compresses the video captured from the camera, a video decoder that decompresses the video received over the network, and a playout buffer that smooths the playout of received video due to network variations. These software components impact capture-to-display latency (CDL) and frame-rate of the real-time video played at a receiver application. Capture-to-display latency is the total time to encode and decode a video frame, playout buffer time, and latency of the network path. Along with bit-rate, these two metrics provide quick insights into the performance of a real-time video application. Developers and testers can use these metrics to determine whether the experimental performance of a video chat or conferencing application meets the expected performance. Moreover, since numerous video chat applications are available today, users can use the CDL and frame-rate metrics to guide their selection of a video chat application.

We present *vDelay*, a tool to measure the capture-to-display latency (CDL) and frame-rate of a video stream. The video stream is captured at the caller user agent and displayed at a callee user agent. Both caller and callee user agents run the same video application. vDelay has three important properties. First, it does not require any change in the source code or executable of a real-time video application. Thus, it can be used to measure the CDL and frame-rate of closed source video applications. Second, vDelay does not require any specialized hardware. Third, it is written in Java so it is platform independent and can be used to measure CDL and frame-rate of a real-time interactive video application on any operating system.

For the rest of the paper, we assume that a video session is established between two participants. We designate one machine running the video chat application as a caller user agent and the other as a callee user agent. For simplicity, we refer to these machines as caller and callee, respectively.

The paper is organized as follows. Section 2 discusses issues involved in measuring CDL and frame-rate. Section 3 presents the architecture of vDelay. Section 4 describes the experimental setup and Section 5 presents CDL and frame-rate results for video chat applications. Section 6 discusses related work. Section 7 presents conclusions and future work.

## 2. MEASURING CDL AND FRAME-RATE

The key to measuring capture-to-display latency (CDL) and frame-rate lies in embedding a timestamp in the caller's video, and retrieving that timestamp at the callee. The timestamp is the current system time at the machine running the caller user agent. Assuming that the machines running the caller and callee user agents are time synchronized within an acceptable error, the capture-to-display latency is the difference between the timestamp retrieved from the caller's video and current system time at the machine running the callee user agent. This difference can also be used to calculate the inter-frame display time at the callee user agent. Further, since every new frame must have an increasing value of a timestamp, the number of frames within a time period can be used to calculate the frame-rate of the received video.

We use a trick to embed the timestamp in the caller's video that does not require any change to the video chat application. The timestamp, i.e., the current sys-

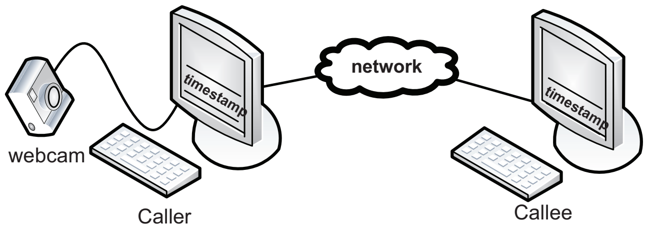

**Figure 1: vDelay setup.**

tem time at the machine running the caller user agent, is displayed at the monitor of the machine running the caller user agent every $t$ time units. In our case, the monitor is a liquid crystal display (LCD) device. A webcam is attached to the machine running the caller user agent and faces the LCD monitor. Thus, it captures the current image on the LCD monitor which includes the timestamp. The caller user agent then encodes this captured frame including the timestamp, and sends it over the network to the callee user agent which decodes the frame and displays it on its attached LCD monitor. An application running on the same machine as the callee user agent grabs the timestamp from the received frame, and calculates the time difference between the timestamp grabbed from the received frame and local system time. The timestamps are processed to calculate CDL and frame-rate. Figure 1 shows the setup for measuring CDL and frame-rate. The novelty of this approach lies in the fact that no additional hardware is needed and no modification to the software of any real-time video application is required.

Next, we discuss how to display and retrieve a timestamp at/from a LCD, and the factors that impact the latency and accuracy of the received timestamp.

## 2.1 How to display and capture a timestamp?

We considered three approaches for displaying the timestamp at the caller user agent. These approaches display the timestamp as (1) an EAN-8 barcode [5], (2) numeric characters, and (3) a progress bar. From experimentation, we found that displaying timestamp as a barcode was the most attractive option. This was so because barcodes such as EAN-8 and EAN-13 have a built in checksum mechanism and because barcode reading is very fast. The checksum is necessary because a barcode image can get distorted due to bandwidth variations and lossy encoding of video codecs. Without a checksum, it is difficult to ascertain whether the timestamp grabbed from the frame is the same as the one displayed at the caller user agent. No built in checksums exist for timestamps displayed as numeric characters and as a progress bar. Although it is conceivable to design checksums for both numeric characters and a progress bar, we did not feel a need since the reading accuracy of the timestamp encoded as a barcode was above 94% for a range of video chat applications (see Section 4).

During our experiments we found that at the callee, barcode image could be read in less than a millisecond, facilitating the online calculation of capture-to-display latency and frame-rate, whereas it took a few seconds to recognize via OCR the timestamp displayed as numeric characters. For these reasons, we have used barcodes to encode and retrieve the timestamp from a video frame.

## 2.2 Factors impacting the embedding and retrieval of timestamp

Below, we discuss the factors that impact the embedding and retrieval of a timestamp. These are LCD refresh rate and response time, camera aperture, shutter speed, and timestamp area in the captured video. The first three impact the capture-to-display latency and frame-rate calculations whereas the last factor impacts the success rate of retrieving barcodes at the machine running the callee user agent. For the rest of the paper, our use of the timestamp means the current system time displayed as a barcode on the LCD monitor of the machine running the caller user agent. In Section 4, we describe how a timestamp is encoded as a barcode.

### 2.2.1 *Refresh rate and response time of a LCD*

The refresh rate determines the speed at which an image is displayed on the LCD monitor and typically starts at 60 Hz on modern LCD monitors. This means that it may take up to 16.6 ms for a timestamp to appear on a LCD monitor. The response time is the amount of time it takes a pixel to refresh itself, i.e., ready itself for displaying the new pixel. The response time on modern LCD monitors is typically 5 ms. Together, refresh rate and response time can delay the displaying of timestamp by a few milliseconds.

To eliminate the impact of refresh rate and response time, a virtual webcam driver can be designed which grabs the frame from the frame buffer and passes it to the caller user agent for encoding. However, the design of such a virtual webcam driver is tightly coupled with the underlying operating system. Thus, we have not explored this approach.

### 2.2.2 *Aperture*

Aperture is a hole in the camera through which light enters the camera. If the hole is narrow, less light enters the camera and the captured image containing the barcode is likely to be dark. Consequently, it may be necessary to keep the camera shutter open for a longer period to capture the barcode being displayed on the LCD monitor. Keeping the shutter open for a long period adds delay to the capture-to-display latency. Therefore, it is necessary to set the camera aperture to its

| Resolution | Size of timestamp |
|---|---|
| 320x240 | 1/4 |
| 640x480 | 1/16 |
| 800x600 | 1/24 |
| 1024x768 | 1/40 |

**Table 1: The size of timestamp encoded as an EAN-8 barcode w.r.t resolution of captured video.**

highest value to minimize the length of the period the shutter is open.

### 2.2.3 *Shutter speed*

Shutter speed is the duration of time for which the shutter of a camera is open. If a shutter remains open when multiple timestamps are being displayed on a LCD monitor, the camera will capture all of these timestamps, and it may be difficult to retrieve them at the receiver. Thus, shutter speed must be greater than the refresh rate of a LCD monitor. Further, a low shutter speed can impact the frame-rate of the received video. Therefore, it should be set to a value that adds the least delay to the capture-to-display latency and maximizes the achievable frame-rate.

### 2.2.4 *Timestamp area in the captured video*

Video chat applications capture the video at different resolutions such as 640x480 and 320x240. The timestamp should occupy a sufficient area in the captured frame to maximize the successful reading of barcodes at the callee machine. Through trial and error, we determined the minimum area that a timestamp encoded as an EAN-8 barcode should occupy in the captured video frame. Table 1 shows these values. These results may be adjustable depending on the quality of a webcam, and the barcode reader. Nevertheless, they provide a useful guideline for conducting similar experiments.

## 3. VDELAY ARCHITECTURE

The vDelay tool consists of a vDelay-S and vDelay-R Java application that run on the caller user agent machine and callee user agent machine, respectively. The vDelay-S application displays the system time as an EAN-8 barcode on the LCD monitor. There are three related issues. First, the system time must be displayed to the resolution of a millisecond to accurately measure capture-to-display latency. Since an EAN-8 barcode can only represent a maximum of eight digits, an EAN-8 barcode can at most capture the eight least significant digits of the system time measured in milliseconds. Second, through experimentation, we found that generating barcodes every millisecond was not computationally efficient, so we generated the barcode images in advance. Based on the least significant digits of the system time, the vDelay-S application selects the appropriate barcode image and displays it on the screen. Lastly, an EAN-8 barcode image can represent a numeric range between zero and 10 million, so it might be necessary to generate these many barcode images. However, we only generated 10,000 EAN-8 barcode images using barcode4j [3], an open source barcode generator, that represent the numeric range [0, 9999]. Depending on the last four digits of the system time measured in milliseconds, the vDelay-S application displays the barcode image that represents those digits and displays them on the LCD monitor. This numeric range implies that after 10 seconds, the same barcode image is displayed on the screen. As discussed in Section 2.2.1, the displaying of barcodes on the LCD monitor is delayed by few milliseconds depending on the refresh rate and response time of a LCD monitor.

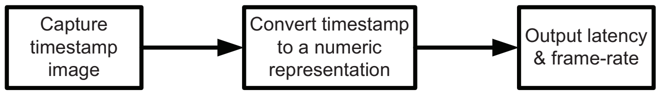
Capture timestamp image
Convert timestamp to a numeric representation
Output latency & frame-rate

**Figure 2: vDelay-R architecture.**

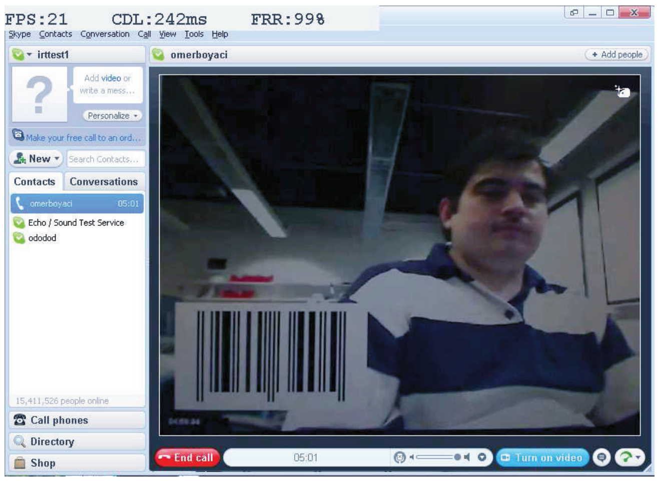

**Figure 3: Screen shot of vDelay-R application. FPS, CDL, and FRR statistics are shown at the top of the image. The barcode received from the caller user agent is also visible.**

Figure 2 shows the block diagram of vDelay-R application. To facilitate grabbing the barcode from the received frame, the vDelay-R application lets user select the (top, left) and (bottom, right) screen coordinates of the barcode image with mouse clicks. The vDelay-R application then grabs the barcode image from the framebuffer every five milliseconds, and passes it to a barcode reader. The time to grab the barcode image from the

| Chat application | Version | Video codec | Resolution | Bit-rate (kb/s) | Fps | CDL (ms) | Std. dev (ms) | Encoding CPU (%) |
|---|---|---|---|---|---|---|---|---|
| Live Messenger | 14.0.8064.206 | H.264 | G 640x480 | 600 | 23 | 69 | 16 | 28 |
| Gtalk | v1.0.8.0 | H.264 | G 512x300 | 1000 | 27 | 99 | 16 | 16 |
| X-Lite | 3.0.47546 | H.263+ | 320x240 | 400 | 27 | 102 | 15 | 20 |
| Yahoo | 9.0.0.2152 | Unknown | 320x240 | 72 | 3 | 113 | 23 | 1 |
| eyeBeam | 1.5.19.5.52345 | H.264 | 640x480 | 400 | 27 | 129 | 16 | 25 |
| AIM | 6.8.14.6 | Unknown | 240x180 | 120 | 9 | 147 | 57 | 20 |
| Tokbox (LL) | 2.01 2351d05 | Unknown | 270x200 | 320 | 24 | 148 | 72 | 25 |
| Skype (HQ) | 4.0.0.215 | VP7 | 640x480 | 560 | 20 | 238 | 22 | 44 |
| Tokbox (HL) | 2.01 2351d05 | Unknown | 270x200 | 320 | 23 | 342 | 69 | 25 |

**Table 2: Comparison of video chat applications. The results are sorted by capture-to-display latency (CDL). The 'G' in the Resolution column is our best guess of the video resolution. LL, HL and HQ are abbreviations for low latency, high latency, and high quality.**

frame is less than a millisecond. We have used zing [16], an open-source barcode reader. The vDelay-R application reads the barcode, retrieves the timestamp, and computes the difference between the local system time and the timestamp retrieved from the frame. It then computes the capture-to-display latency and frame-rate and outputs them to the LCD monitor and writes them to a file. It also computes the first read rate (FRR) of barcodes and writes it to the display and a file. Figure 3 shows a screen shot of vDelay-R application. Kato *et al.* [17] define FRR as:

$$\text{FRR} = \frac{\text{Number of successful first reads}}{\text{Number of attempted first reads}}$$

A timestamp from a single frame can be grabbed multiple times depending on the the instant at which the frame is grabbed from the screen. However, the vDelay-R application reports the difference between the earliest grabbed timestamp from a frame and the current system time.

## 3.1 Clock synchronization

vDelay tool assumes that clocks are synchronized between the machines running the video chat applications. With a minor adjustment, vDelay tool can be used to calculate the capture-to-display latency when clock synchronization may not be possible. The idea is that the callee user agent reflects the video containing the timestamp back to the caller user agent. To reflect the video without requiring any change to the video chat application, the webcam attached to the machine running the callee user agent points towards the LCD monitor of the callee machine which displays the video received from the caller that also contains the timestamp. The vDelay-R application is run at the caller user agent which retrieves the timestamp from the frame received from callee. vDelay-R then compares the timestamp with its current time to calculate the time elapsed since the frame was sent from caller to callee user agent. This elapsed time includes the round-trip network delay, and video encoding, decoding, and playout time at the caller and callee user agent. Assuming the round-trip network delay is negligible as it would typically be in a laboratory LAN, one half of this elapsed time approximately gives the capture-to-display latency.

## 4. EXPERIMENTAL SETUP

Figure 1 shows the experimental setup. It consists of two machines, each with an Intel Core Duo processor [8] and a Dell 1909W flat panel display [4]. The brightness on the LCD monitors is set to its maximum value. Both machines run the Windows Vista operating system and are connected to the same subnet (RTT $< 1\,\text{ms}$). The time on both machines is synchronized through NTP and the NTP server query interval was 10 seconds. Each machine runs a video chat application. A Logitech Quickcam Pro 9000 webcam [9] is attached to the machine running the caller user agent and point towards the LCD monitor displaying the timestamp. A video session is established between two user agents. The caller user agent sends the captured video including the timestamp encoded as barcode over the network to the callee user agent.

The webcam attached to the caller user agent captures the images on the LCD monitor. These images include icons and desktop applications along with the timestamp. It can be argued that video chat applications are optimized for human images and not the computer displays, and thus the statistics obtained for CDL and frame-rate may not reflect the common use case for these applications. To address this, we prerecorded a video of a human user sitting in front of a webcam and run it on the machine running the caller user agent. The vDelay-S application displays the timestamp at a corner of this prerecorded video of a human user. The webcam connected to the machine running the caller

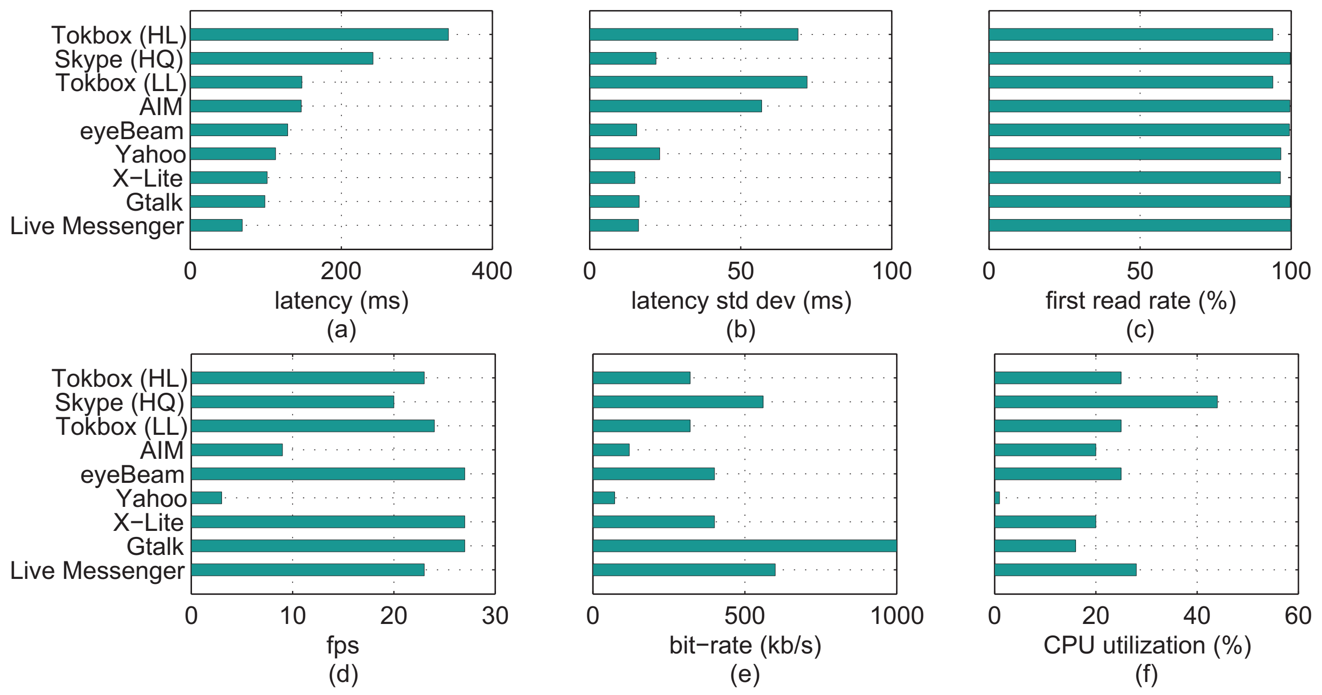

**Figure 4: (a) Capture-to-delay latency (CDL) (b) Standard deviation of capture-to-display latency (c) First read rate (FRR) (d) Frames per second (fps) (e) Bit-rate (f) CPU utilization for video encoding.**

user agent sender points to the monitor and captures the prerecorded video of a human user and the timestamp, thereby mimicking a realistic video chat session.

At the machine running the callee user agent, we run the vDelay-R Java application after the video call has been established and the caller's video along with the barcode is visible at the LCD monitor. We run the video session for 10 minutes and report the CDL and frame-rate results for this duration.

It is possible that the only webcam available is the one attached to the display of a machine (such as LCD of a laptop) and cannot be detached. Thus, it cannot be pointed towards LCD monitor displaying the timestamp. To resolve this, a mirror can be placed in front of the LCD monitor of the machine running the caller user agent. The camera attached to the top of the LCD monitor can capture the timestamp being displayed in the mirror.

## 5. RESULTS

We used vDelay tool to measure capture-to-display latency and frame-rate of Skype [11], Windows Live Messenger [13], Yahoo Messenger [15], GMail video chat [7], AOL Instant Messenger (AIM) [2], X-Lite [14], eyeBeam [6], and Tokbox [12] applications. We used the setup described in Section 4. For all the video chat applications, we ran the experiment for ten minutes and repeated it twice. The Tokbox application completely runs in browser and only depends on the availability of a Adobe Flash player. In Tokbox, the caller user agent sends packet over TCP to a Flash server maintained by Tokbox which forwards these packets to the callee user agent over TCP and vice versa. With the exception of Tokbox, the caller user agent sends packets directly to the callee user agent. Besides Tokbox and Yahoo Messenger, all the video applications send packets over UDP. For Skype, the video session was of high quality (HQ) as indicated by an icon in the received video.

Table 2 shows the results of these video applications. The reported results include capture-to-display latency (CDL), standard deviation of CDL, frame-rate, and first read rate (FRR) measured using vDelay, and bit-rate and CPU utilization of the caller user agent that encodes the video. The results are sorted by capture-to-display latency. For ease of comparison, we also graphically show these results in Figure 4. As mentioned before, Tokbox forwards packets from a caller user agent to a callee user agent through servers which are based in different geographical locations. The use of a server in different location impacts the CDL. Therefore, we report the minimum and maximum observed CDL for Tokbox which are abbreviated as LL (low latency) and HL (high latency) in Table 2.

Our results indicate that amongst all video chat applications, Windows Live Messenger has the best CDL value. For Tokbox (LL), Tokbox (HL), and AIM, the

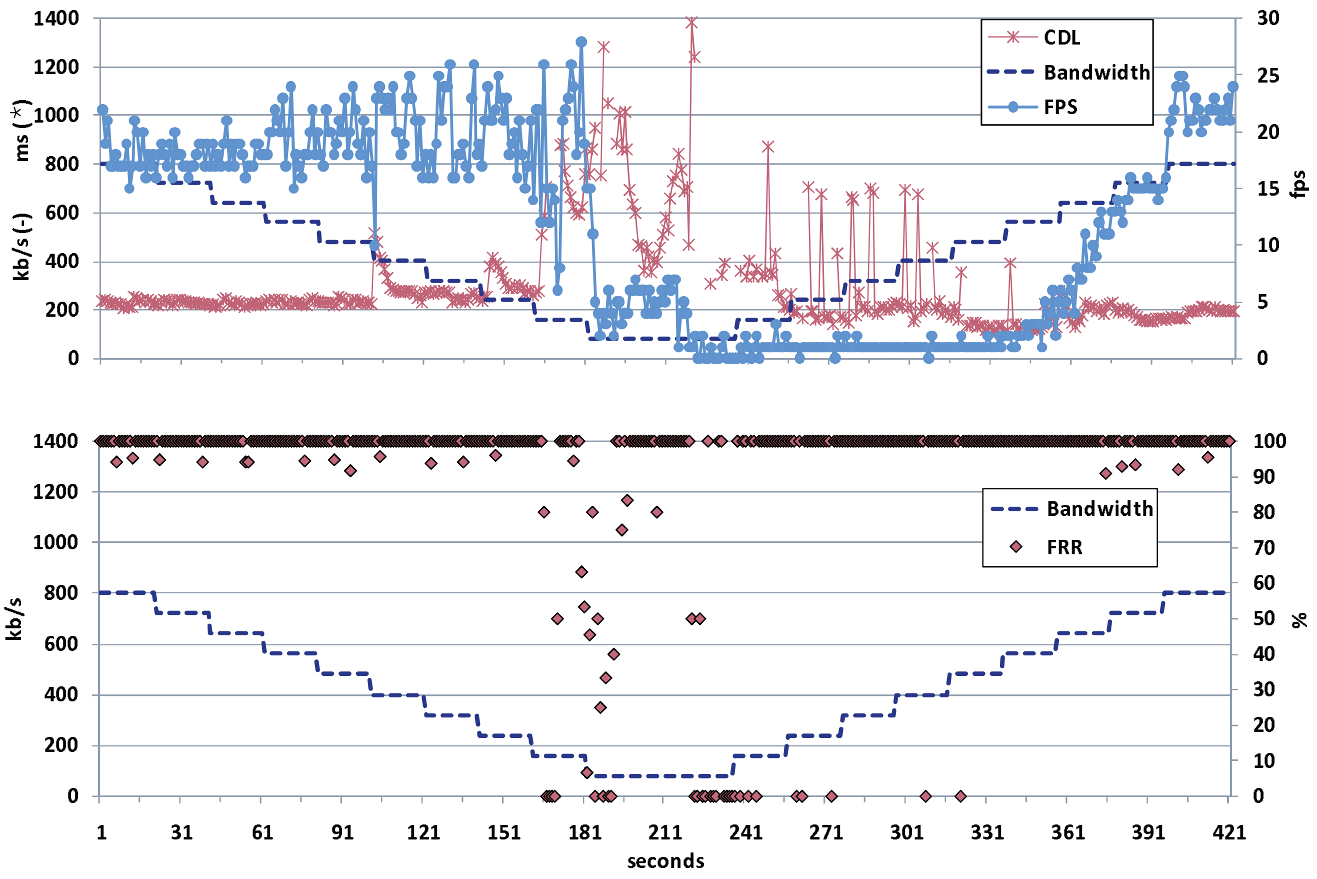

**Figure 5: CDL, fps, and FRR for Skype as a function of time when the available bandwidth is adjusted as a step function.**

standard deviation of CDL is more than 50 ms. We conjecture that Tokbox has a high standard deviation for CDL due to the packet scheduling at the server relaying media packets. For AIM, we attribute the high standard deviation to the video encoding function.

X-Lite and eyeBeam have the highest achieved frame-rate per second (fps). Except for Yahoo Messenger and AIM, the frame-rate of all video chat applications is above 20 frames per second. As for the CPU utilization of the machine running the caller user agent, we measured that Skype uses 44% CPU, the maximum amongst all applications. Gtalk tops the bit-rate comparison at 1,000 kb/s.

vDelay can be used to measure CDL and frame-rate of a video chat application under controlled network conditions. Such use provides a powerful testing mechanism for application developers. One instance is shown in Figure 5, which shows the performance of Skype when the available bandwidth of a video session is adjusted as a step function. The figure shows that Skype suffers from a high jitter in frame-rate as the available bandwidth is gradually decreased. With the decrease in available bandwidth, CDL starts to increase indicating the impact of network queuing and playout buffer adjustments. The CDL graph shows large spikes when available bandwidth is below 400 kb/s. However, the CDL of Skype is close to its operating mean when the available bandwidth is above 400 kb/s.

## 6. RELATED WORK

Existing video latency measurement tools involve the use of a specialized hardware. OmniView [10] is a tool that uses a specialized PCI card. Our goal is to measure video latency without the use of any specialized hardware.

Yoshimura *et al.* [18] designed a module for a video streaming application that for each frame measures the deviation from the playout time. Their approach does not calculate capture-to-display latency or frame-rate, and requires changing the video application. Further, their approach is targeted towards video streaming applications.

adelay [1] is a tool that can be used to measure mouth-to-ear latency.

## 7. CONCLUSIONS AND FUTURE WORK

We have designed *vDelay*, a tool to measure capture-to-display latency (CDL) and frame-rate of a video application in real-time. vDelay does not require any change in the source code or executables of video chat applications. Thus, it can be used to measure CDL and frame-rate of closed source applications. Further, it is platform independent and does not require use of any specialized hardware. We have used vDelay to measure CDL and frame-rate of popular video chat applications. Our results indicate that Windows Live Messenger has the best capture-to-display latency. Together with bit-rate, the application designer can use CDL and frame-rate statistics gathered using vDelay to determine if the measured performance of the video chat application meets the expected performance under various network conditions.

In addition to CDL and frame-rate, it is useful to determine if the video chat application synchronizes the video and audio playout, also known as lip-sync. Incorporating lip-sync in vDelay is the subject of future work.

## 8. REFERENCES

[1] adelay. A tool to measure mouth-to-ear latency. http://www.cs.columbia.edu/irt/software/adelay/.

[2] AOL Instant Messenger. http://www.aim.com/.

[3] Barcode4J. http://barcode4j.sourceforge.net/.

[4] Dell 1909W flat panel monitor. http://tinyurl.com/dmpnyn.

[5] EAN barcode. http://en.wikipedia.org/wiki/European_Article_Number.

[6] eyeBeam. http://www.counterpath.com/eyebeam.html.

[7] GMail video chat. http://mail.google.com/videochat/.

[8] Intel Core Duo Processor. http://tinyurl.com/y3kl3x.

[9] Logitech Quickcam Pro 9000. http://www.logitech.com/index.cfm/38/3056.

[10] OmniView. http://www.omnitek.tv/admin/old_support/AVdelay1[1].pdf.

[11] Skype video calls. http://www.skype.com/allfeatures/videocall/.

[12] Tokbox. http://www.tokbox.com/.

[13] Windows Live Messenger. http://messenger.live.com/.

[14] X-Lite. http://www.counterpath.net/x-lite.html.

[15] Yahoo Messenger. http://messenger.yahoo.com/.

[16] zing barcode reader. http://code.google.com/p/zxing/.

[17] H. Kato and K. Tan. First read rate analysis of 2d-barcodes for camera phone applications as a ubiquitous computing tool. In *Proc. of TENCON – IEEE Region 10 Conference*, November 2007.

[18] Y. Yoshimura and M. Masugi. A QoS monitoring method for video streaming service based on presentation-timeline detection at user clients. In *Proc. of Joint 10th Asia-Pacific Conference on Communications and 5th International Symposium on Multi-Dimensional Mobile Communications*, September 2004.