Performance of Video-Chat Applications Under Congestion

Omer Boyaci, Andrea G. Forte, Henning Schulzrinne
Department of Computer Science
Columbia University
{boyaci, andreaf, hgs}@cs.columbia.edu

ABSTRACT

We study the performance of four popular IM clients focusing our attention on video-chat. In particular, we analyze how Skype, Windows Live Messenger, Eyebeam and X-Lite react to changes in available bandwidth, presence of HTTP and bit-torrent traffic and random packet losses.

Keywords

Video delay, bandwidth adaptation, measurements, Skype, Live Messenger, X-Lite, Eyebeam

1. INTRODUCTION

When thinking about high-speed Internet, we have to consider many different technologies such as ADSL, cable and satellite. Usually, for all of them the speed of the downlink and the speed of the uplink differ. For example, for ADSL it is common to have downlink speeds of 3 Mb/s and uplink speeds of 1 Mb/s with the uplink speeds always significantly lower than the downlink speeds. For common web applications such as web browsing and video streaming, this does not cause problems as the amount of information sent on the uplink is considerably lower than the one received on the downlink. Things however are different for applications such as video chat where a client needs not only to receive video but also to send it. In such case, the uplink becomes the bottleneck of the system. To make things worse, the presence of cross traffic creates congestion on both links, with the video stream on the uplink suffering more, given the much lower available bandwidth on the uplink. In other words, when congestion happens the video stream on the uplink will suffer first, thus determining the quality for the whole video chat. Because of this, in the rest of the paper we focus our attention on the uplink only.

Changes in bandwidth can significantly affect video and audio quality, with video suffering the most given its higher bandwidth requirements. Usually, congestion happens because bandwidth has to be shared among multiple competing flows. When this happens, the way an application reacts can bring to different scenarios by either increasing the overall congestion or by trying to maintain a fair share of bandwidth among flows. In order to analyze this aspect in our measurements we consider HTTP and bit-torrent traffic competing with video and audio traffic.

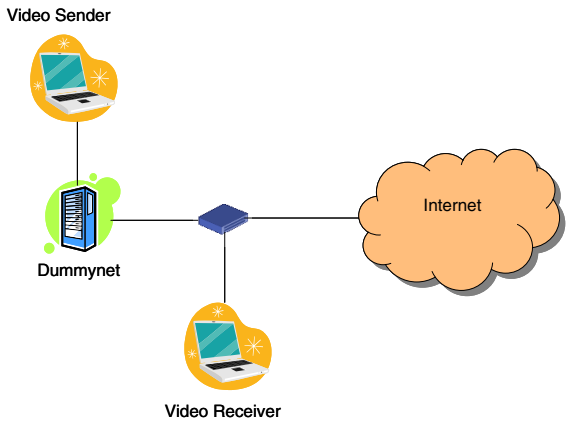

Figure 1: Experimental testbed

The application needs to find a way to detect congestion. Usually, a way to do so is through packet loss. When congestion happens, the queues in the routers fill up quickly and at some point overflow. This causes packets to be dropped and lost. Furthermore, when considering wireless networks, for example, there are random losses due to the medium itself and not to congestion. Because of this, a way to distinguish between wireless losses and congestion losses needs to be defined. More will be discussed in Section 3.

We tested Skype 4.0.0.215 [2], Windows Live Messenger 14.0.8064.206 [11], Eyebeam version 1.5.19.5.52345 [1] and X-Lite 3.0.47546 [3].

Among the clients we tested, Skype behaved the best by adapting its codec parameters based not only on packet loss but also on RTT and jitter. This allowed Skype to closely follow the changes in bandwidth without causing any packet loss. Eyebeam performed the worst with high fluctuations in the transmission speed of its video traffic and with poor adaptation to bandwidth fluctuations.

The rest of the paper is organized as follows. In Section 2 we give an overview on the state of the art, Section 3 describes the difference between random losses and losses due to congestion. In Section 4 we present our experimental results and finally Section 5 concludes the paper.

978-0-7695-3890-7/09 $26.00 © 2009 IEEE
DOI 10.1109/ISM.2009.45

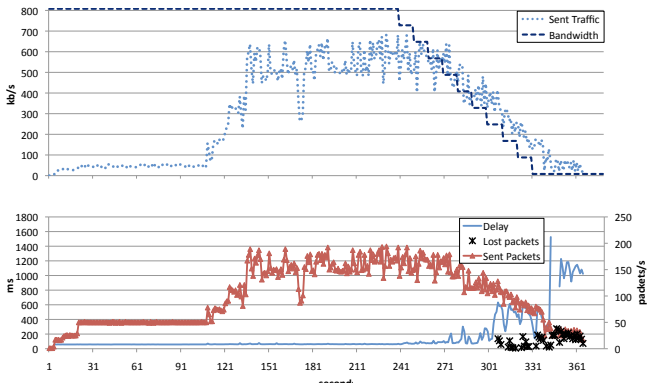

Figure 2: Skype with 10-10 step function

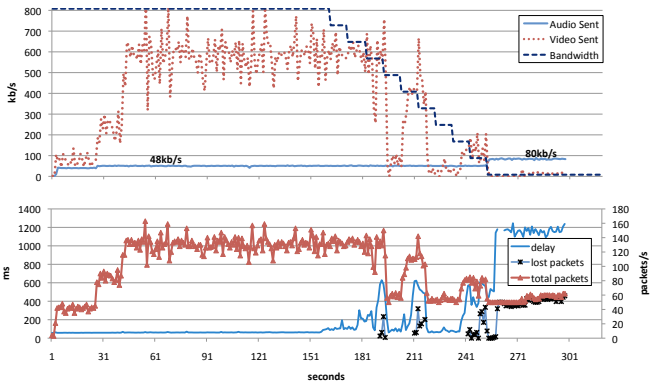

Figure 3: Windows Live Messenger with 10-10 step function

2. RELATED WORK

Real-time video chat has stricter requirements than streaming video. Popular streaming video websites like Youtube, Hulu, Netflix and Joost use TCP, which has flow and congestion control mechanisms. They buffer the content in the client before playing. The content is stored at more than one bitrate and the most appropriate one is used. Netflix determines the available bandwidth itself whereas Hulu and Youtube allow users to switch to high quality. Several studies propose video streaming over heterogeneous environments [8, 12]. Real-time video chat, however, has very strict delay requirements and the retransmission mechanism of TCP does not fit into this model. Because of this, all the measured video chat clients stream over UDP.

In order to avoid congestion or under-utilization of the link the sender needs to adjust its transmission rate. Under-utilization may cause low quality because uplink of the residential area networks are limited. Over-utilization causes unfairness to other traffic as well as packet loss, hence degrading video quality.

The performance of audio chat applications has been studied extensively compared to video chat. Baset and Schulzrinne compared Skype, MSN, Yahoo and Gtalk in terms of audio quality and mouth-to-ear latency [4]. Hofeld and Binzenhfer measured Skype quality and bandwidth adaptation in UMTS [9]. Although their work studies performance of Skype under congestion, it only covers audio calls whereas we focus on video calls.

The piece of work closest to ours is [5]. Here, however, the authors only measure Skype's video adaptation to bandwidth variations. On the other hand, we cover Skype, Windows Live Messenger, Eyebeam, a commercial SIP-based client and X-Lite, a free SIP-based client. We use Skype 4.0 for Windows which supports high-quality video chat whereas they used Skype 2.0 for Linux.

3. RANDOM LOSSES VS. CONGESTION LOSSES

Generally, applications consider packet loss a sign of congestion. This is usually true since during congestion queues in the routers fill up and packets get dropped. There are situations, however, in which packet loss is not a sign of congestion. This is true, for example, in a wireless environment. The wireless medium introduces by its own nature losses due to many factors such as signal fading, obstacles and co-channel interference [7]. Because of this, an application needs to distinguish between the two kinds of losses. If an application is not capable to distinguish between the two kinds of losses, when a wireless loss occurs, the application will think that the medium is congested and therefore will try to back-off by lowering its sending rate. This, however, will only be counterproductive since there is no congestion and yet the application will experience lower quality due to its lower sending rate.

An algorithm that helps in distinguishing between the two types of losses is Spike [14]. Spike is an end-to-end loss differentiation algorithm (LDA) which is based on relative one-way trip time (ROTT). Spike classifies a loss as congestion related if the loss happens when the ROTT presents a spike.

4. MEASUREMENTS

In this section we introduce our experimental results.

4.1 Experimental Setup

We deployed a small testbed consisting of a desktop PC running FreeBSD 7.1 and two Lenovo Thinkpad X63 laptops running Windows Vista. In order to adjust the available bandwidth we used the desktop PC as a gateway by installing two ethernet cards and by running the dummynet application [13]. By using the dummynet application we were able to adjust many different parameters such as queue sizes, RTT, maximum bandwidth and random packet loss. Figure 1 shows the setup for the experiments. All PCs were connected to the Internet and all traffic between sender and receiver was going through the PC running dummynet. The two laptops were used as IM clients, that is were running a video chat. One desktop PC running FreeBSD and Dummynet was used as gateway. All machines used as IM clients were also running Wireshark [16] in order to collect and later

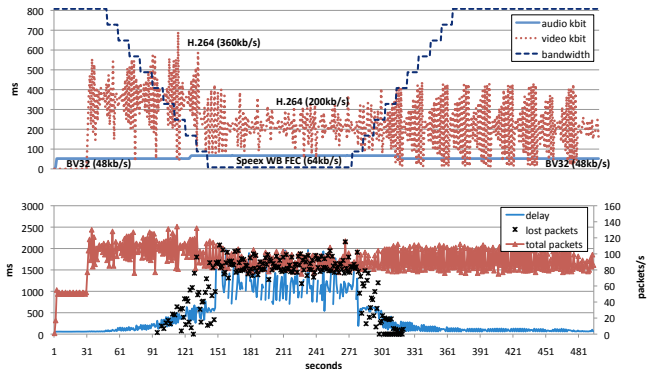

Figure 4: Eyebeam with 10-10 step function

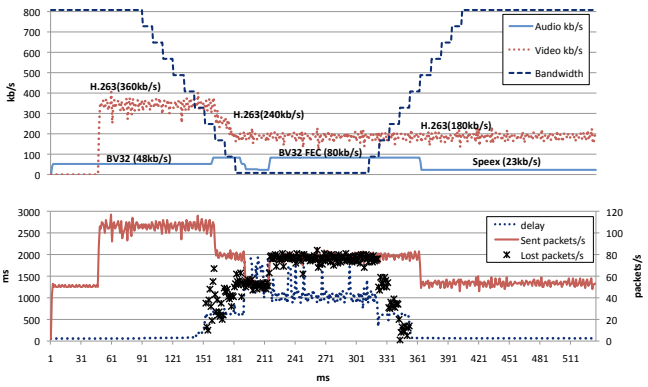

Figure 5: X-Lite with 10-10 step function

analyze packet flows.

4.2 Results

We wanted to emulate a video-chat session between two ADSL users located on either coast of the United States. For all experiments we set the total RTT value to 114 ms to emulate a cross-country link, queue size to 60 kB [6] and the maximum available bandwidth to 3 Mb/s for the downlink and 1 Mb/s for the uplink.

We performed three sets of experiments. In the first set, we analyzed how the IM clients adapt to changes in bandwidth. In the second set, we measured the impact of cross traffic, either HTTP or BitTorrent as well as the impact of random losses.

4.2.1 Changes in bandwidth

We modify the available bandwidth by following a step function. We consider two step functions. The first step function decreases and increases the available bandwidth of 80 kb every 10 seconds, while the second one has decreases and increases of 400 kb every 10 seconds.

Figures 2, 3, 4 and 5 show the measurement results for Skype, Live Messenger, Eyebeam and X-Lite, respectively, when the first step function is used. We also show what different video and audio codecs were used and how the applications changed codecs depending on the congestion level.

In the following we describe in more detail the behavior of each IM client.

X-Lite does not support H.264 for video, but it rather uses H.263 which has poorer quality compared to H.264. From our measurements, we have seen that it has a minimum bitrate of 180 kb/s as it does not go below such value even when it experiences 100% loss. When congestion happens, even though it experiences 100% packet loss, it does not stop the video. It tries to recover from a congestion situation by using Forward Error Correction (FEC) for audio. This however, does not help much as it contributes to increasing the level of congestion by increasing the packet size. Finally, X-Lite does not drop the call even though the audio quality is very poor.

Generally speaking, a good video-chat application, when in a congested state, should drop the video stream in order to preserve the audio stream as much as possible. Furthermore, if congestion is so high that even the audio stream is severely affected, then it should drop the call. This would help in lowering the overall level of congestion and it would not represent a big penalty for the user since the quality of the call was extremely poor.

In terms of bandwidth, X-Lite decreases its transmission speed gradually. Unfortunately, once congestions stops and more bandwidth becomes available, X-Lite does not increase its transmission speed.

Eyebeam uses the H.264 video codec. Similarly to X-Lite, it tries to use FEC when it detects high congestion while still trying to keep both video and audio streams. In other words, it does not try to disable video in order to keep the audio quality to an acceptable level. Furthermore, it seems to support only two different bitrates for video. This is insufficient to support the different levels of congestion. Eyebeam presents much higher fluctuations in transmission speed than X-Lite, due perhaps to the implementation of the H.264 codec. Such fluctuations translate in higher losses when the available bandwidth starts decreasing since the peaks of such fluctuations exceed the maximum available bandwidth. Furthermore, once the available bandwidth starts increasing again, similarly to X-Lite, the transmission speed does not increase, staying steady at the lower speed, thus never reaching the original level.

Skype behaves differently than the other IM clients. In particular, it promptly adapts its transmission rate to changes in bandwidth, thus preventing packet loss until the minimum bit-rate is reached at which point it drops the call. Skype has this behavior because it uses other metrics on top of packet loss in order to detect congestion. Parameters such as RTT and jitter are taken into account. In particular, we can see from Figure 2 that as the available bandwidth goes down, the transmission speed follows it closely, avoiding packet loss. On the other hand, for the other IM clients packet loss starts much earlier since in order to detect congestion they need to "see" some packet loss. Also, the way other IM clients lower their transmission rate is much more aggressive.

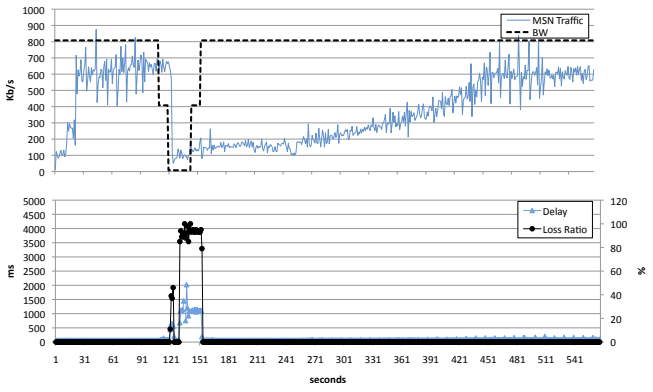

Figure 6: Windows Live Messenger behavior when decreasing/increasing the available bandwidth according to 10-50 step function

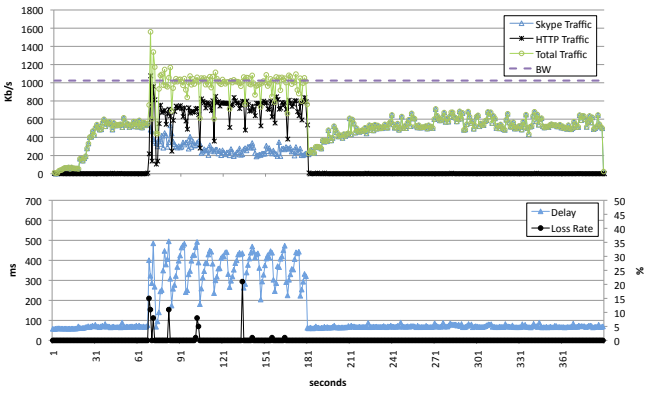

Figure 7: Skype bandwidth, delay and loss in the presence of concurrent HTTP traffic

While Skype reacts to congestion by trying to closely match the available bandwidth, Windows Live Messenger drastically drops its transmission speed when it detects congestion. In particular, it drops the video transmission rate and then slowly tries to increase it again. No action is taken on the audio flow. When it reaches very low bit-rates, it completely disables the video and it adds FEC to the audio trying to preserve the audio stream as much as possible. A disadvantage of Live Messenger compared to Skype is that its minimum audio bit-rate is 50 kb/s which prevents it from operating at very low bit-rates. Skype audio codec, on the other hand, can operate at bit-rates as low as 16 kb/s. In terms of bandwidth, Live Messenger decreases its transmission rate with the available bandwidth. As the available bandwidth increases, the transmission speed for Live Messenger increases very slow taking up to 9 minutes to reach the original value.

According to our subjective observations, Skype and MSN when in congested state decrease video frame rate and quality, showing an almost-still image with few artifacts as there is no or little packet loss. On the other hand X-Lite and Eyebeam try to keep their frame-rate and quality high, showing a smoother video but with lots of artifacts due to higher packet loss. We believe that in terms of end-user experience,

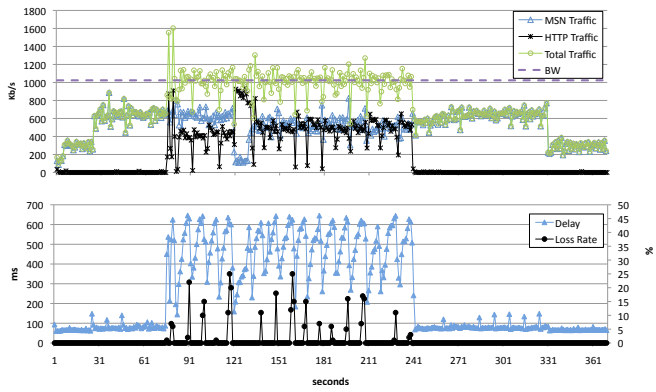

Figure 8: Windows Live Messenger behavior in the presence of concurrent HTTP traffic

the first approach is better. Low-quality and low frame-rate video without artifacts gives a better user experience than high frame-rate video with lots of artifacts.

Lastly, video chat applications should be able to lower their video bit-rate to very low levels in order to keep the audio at an acceptable level. Video codecs should be able to adapt to changes in bandwidth by supporting any requested bit-rate. Such behavior we have seen it only in Skype while the other video-chat applications support only a few fixed bit-rate levels.

When the second step function is used, all IM clients behave similarly to the case of the first step function. In this case, however, Live Messenger seems to be performing best by quickly adapting to the sudden change in bandwidth (see Figure 6). In particular MSN monitors the packet loss ratio and if it sees a very high packet loss then it drastically drops its transmission rate and when more bandwidth becomes available, it increases its transmission rate very slowly. However, if Live Messenger sees low packet loss, it still drops its transmission rate to a very low level but this time it tries to increase it back to its original value in a very short time.

Eyebeam performs worst as it lowers its transmission speed only after the bandwidth has increased back to its original value. Still, both Eyebeam and X-Lite do not increase back their transmission speeds once the available bandwidth is back to its original value.

4.2.2 HTTP as Cross-traffic

We show our experimental results when a video chat encounters HTTP cross traffic. That is, the cross traffic competes with the audio and video traffic for available bandwidth. In this case, we use Dummynet just to restrict the uplink bandwidth to 1 Mb/s.

For generating HTTP traffic on the uplink, we uploaded a 9 MB file to a web-hosting service called Media Fire [10].

Eyebeam and X-Lite show a similar behavior; this is not surprising given that they are both products of the same company. In particular, both do not adjust their transmission rate at all, keeping it steady at the same value it had before the

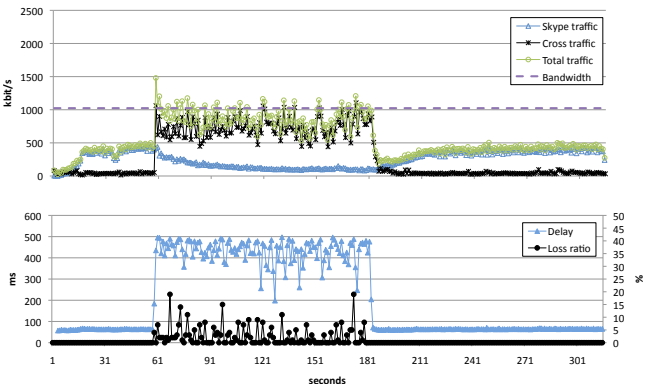

Figure 9: Skype behavior in the presence of concurrent bit-torrent traffic

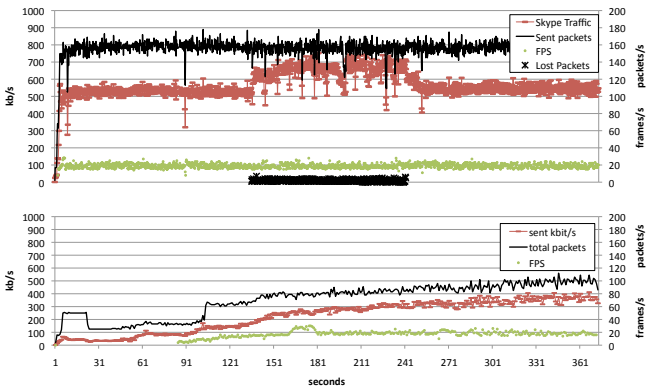

Figure 10: Skype behavior in the presence of random losses

competing traffic was introduced. As a consequence, bandwidth is shared in a more or less fair way between audio-video traffic and HTTP traffic. Packet loss is higher for Eyebeam than for X-Lite because Eyebeam's transmission rate fluctuates. As mentioned earlier, such fluctuations are due to the video codec Eyebeam uses and its implementation. Such heavy fluctuations cause spikes in transmission rate which translate to spikes in used bandwidth, that is spikes in packet loss. It is curious to notice how the free version of Eyebeam, that is X-Lite, does not present such spikes. This is due to the fact that X-Lite is using a different video codec, H.263.

Skype adapts to the presence of other traffic by lowering its transmission rate. As we can see from Figure 7, the very good adaptability of Skype allows it to generate very low packet loss. Unfortunately, in doing so, Skype will always be penalized as the bandwidth that is not used by Skype is consumed by the HTTP traffic.

Live Messenger, similar to Eyebeam and X-Lite, does not lower its transmission rate, keeping it steady. On one hand this prevents HTTP from using most of the available bandwidth, on the other hand it causes higher packet loss.

4.2.3 Bit-torrent as Cross-traffic

In this section we discuss our experimental results when introducing bit-torrent traffic during a video chat. For the bit-torrent traffic we used the Vuze [15] bit-torrent client and did not limit its maximum upload speed.

Eyebeam and X-Lite react in the same way as in the HTTP case. They both keep the same transmission rate with and without bit-torrent traffic. The fluctuations in transmission rate in Eyebeam, cause a higher loss rate than in X-Lite. It is interesting to note that since X-Lite and Eyebeam do not lower their transmission rate, they will prevent bit-torrent traffic from consuming more bandwidth.

On the other hand, Skype lowers its transmission rate as soon as bit-torrent traffic is introduced (see Figure 9). This limits the losses of video and audio traffic, however bit-torrent traffic will take most of the available bandwidth. Once the bit-torrent traffic stops, Skype goes back to its initial transmission speed fairly quick. The reason why Skype lowers its transmission rate considerably more than with HTTP traffic, is because the bit-torrent client opens several concurrent TCP connections taking in our experiments about 85% of the available bandwidth. We observed more than 20 concurrent TCP connections.

Live Messenger behaves differently than in the HTTP case. When bit-torrent traffic is introduced, Live Messenger lowers its transmission rate significantly, leaving almost all the available bandwidth to the bit-torrent traffic. This is because in this case the amount of cross traffic is much higher than in the HTTP case, therefore the amount of packet loss is also higher. This causes Live Messenger to lower its transmission rate considerably. In the HTTP case, the amount of cross traffic and therefore packet loss was considerably smaller, thus leaving its transmission rate the same. As mentioned before, once the cross traffic is removed, Live Messenger takes a long time to go back to its original transmission rate.

Ideally, the desired behavior would be an equal share of bandwidth with small packet loss. This makes Eyebeam, X-Lite and Windows Live Messenger all better than Skype from this point of view. By lowering its transmission speed, Skype just frees bandwidth which is then taken by other flows, leaving the level of congestion unchanged.

4.2.4 Random Losses

Packet losses degrade the quality of a video chat significantly. This is especially true with modern codecs like H.264 as there is a high correlation between frames. Therefore, a bandwidth adaption algorithm should try to eliminate packet losses by decreasing its transmission rate in case of congestion. However, not all losses are due to congestion, wireless networks introduce random losses due to signal fading, interference and channel quality. Decreasing transmission rate will not help in case of non-congestion related losses. Some other techniques like FEC and retransmissions can be utilized. However, in order to respond to losses an application should differentiate congestion losses from random ones.

In these measurements we wanted to see how the various IM clients behaved to random losses and in particular, to see if they could differentiate random losses from congestion

losses.

By using Dummynet we introduce 1% random packet loss and consider two scenarios. In one we introduce packet loss throughout the video-chat session, in the other one we introduce packet loss in the middle of the video-chat session. Figure 10 shows our results for Skype in these two scenarios. The top graph refers to random losses introduced in the middle of the video-chat session while the graph on the bottom refers to random losses introduced throughout the video-chat session.

In both scenarios Eyebeam and X-Lite do not change their rate.

Skype on the other hand behaves differently. When all the losses are introduced in the middle of the chat session (see Figure 10), Skype reacts by adding FEC, therefore *increasing* its transmission bit-rate by about 20%. This is different from the case when losses are due to congestion as in that case Skype *decreases* its transmission rate. Skype can distinguish between congestion losses and random losses by monitoring packets delay. In case of random losses packet delay does not change while in case of congestion losses packet delay spikes [14]. Such increase in transmission rate is not due to retransmissions triggered by the losses as the increase in transmission rate would then be on the same order of the random losses, that is 1%.

When random losses are introduced throughout the video chat session (see Figure 10), Skype increases its transmission speed gradually. This is different from its usual behavior of reaching full transmission speed almost immediately.

5. CONCLUSIONS

We built a testbed in order to analyze the behavior of four popular IM clients, focusing on the video-chat feature and on how such clients react to changes in bandwidth due to congestion. We analyzed the behavior of Skype, Live Messenger, X-Lite and Eyebeam. As competing traffic we considered both HTTP traffic and bit-torrent traffic.

We found that Skype adapts gradually to changes in bandwidth, reacting to both increases and decreases in bandwidth. Because Skype monitors also RTT and jitter on top of packet loss, usually it can adapt its transmission speed before packet loss occurs. Live Messenger drops its transmission rate drastically when packet loss is detected and increases its transmission rate very slowly when there is available bandwidth. Because of this, Live Messenger performs best when drastic drops in available bandwidth happen. On the other hand, however, it does take an extremely long time to raise back its transmission rate.

X-Lite and Eyebeam do not change their transmission speed when cross traffic is present which makes them less sensitive to the presence of bit-torrent traffic. When the available bandwidth decreases, they decrease their transmission speed. Unfortunately, once more bandwidth becomes available, both X-Lite and Eyebeam do not increase their transmission rate. Finally, Eyebeam presents strong fluctuations in transmission rate due to the codec used and its implementation. These fluctuations are not present in X-Lite and cause higher packet loss when spikes in transmission speed occur.

Due to limited upstream bandwidth, video clients must have bandwidth adaptation mechanisms and must be able to differentiate between wireless losses and congestion losses.

6. REFERENCES

[1] Eyebeam. http://www.counterpath.com, 2009.

[2] Skype. http://www.skype.com, 2009.

[3] X-Lite. http://www.counterpath.com, 2009.

[4] S. A. Baset and H. G. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. April 2006.

[5] L. De Cicco, S. Mascolo, and V. Palmisano. Skype video responsiveness to bandwidth variations. In *NOSSDAV '08: Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 81–86, New York, NY, USA, 2008. ACM.

[6] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing Residential Broadband Networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, 2007. ACM Press.

[7] A. G. Forte, S. Shin, and H. Schulzrinne. IEEE 802.11 in the Large: Observations at an IETF Meeting. Technical report, Columbia University, November 2006.

[8] K.-M. Ho, W.-F. Poon, and K.-T. Lo. Performance Study of Large-Scale Video Streaming Services in Highly Heterogeneous Environment. *Broadcasting, IEEE Transactions on*, 53(4):763–773, Dec. 2007.

[9] T. Hofeld and A. Binzenhfer. Analysis of Skype VoIP traffic in UMTS: End-to-end QoS and QoE measurements. *Computer Networks*, 52(3), 2008.

[10] MediaFire. MediaFire. http://www.mediafire.com, 2009.

[11] Microsoft. Windows Live Messenger. http://messenger.live.com, 2009.

[12] G.-M. Muntean, P. Perry, and L. Murphy. A Comparison-Based Study of Quality-Oriented Video on Demand. *Broadcasting, IEEE Transactions on*, 53(1):92–102, March 2007.

[13] L. Rizzo. dummynet. http://info.iet.unipi.it/ luigi/ip_dummynet, 2008.

[14] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda. Achieving moderate fairness for UDP flows by path-status classification. In *LCN '00: Proceedings of the 25th Annual IEEE Conference on Local Computer Networks*, page 252, Washington, DC, USA, 2000. IEEE Computer Society.

[15] Vuze Inc. Vuze. http://www.vuze.com, 2009.

[16] Wireshark Foundation. Wireshark. http://www.wireshark.org, 2006.