Skype Video Responsiveness to Bandwidth Variations

Luca De Cicco
Politecnico di Bari
Bari, Italy
ldecicco@poliba.it

Saverio Mascolo
Politecnico di Bari
Bari, Italy
mascolo@poliba.it

Vittorio Palmisano
Politecnico di Bari
Bari, Italy
vpalmisano@poliba.it

ABSTRACT

The TCP/IP stack has been extremely successful for reliable delivery of best-effort, time insensitive elastic type data traffic. Nowadays, the Internet is rapidly evolving to become an equally efficient platform for multimedia content delivery. Key examples of this evolution are, to name few, YouTube, Skype Audio/Video, IPTV, P2P video distribution such as Coolstreaming or Joost. While YouTube streams videos using the Transmission Control Protocol (TCP), applications that are time-sensitive such as Skype VoIP or Video Conferencing employ the UDP because they can tolerate small loss percentages but not delays due to TCP recovery of losses via retransmissions. Since the UDP does not implement congestion control, these applications must implement those functionalities at the application layer in order to avoid congestion and preserve network stability. In this paper we investigate Skype Video in order to discover at what extent this application is able to throttle its sending rate to match the unpredictable Internet bandwidth while preserving resource for co-existing best-effort TCP traffic.

Categories and Subject Descriptors

C.2.5 [**Local and Wide-Area Networks**]: Internet; C.4 [**Performance of Systems**]: Measurements
; H.4.3 [**Information Systems Applications**]: Communications Applications

General Terms

Measurements, Performance, Experimentation

Keywords

Skype, Video over IP, Congestion Control

1. INTRODUCTION

The TCP/IP stack has been extremely successful for reliable delivery of best-effort, time insensitive elastic type data traffic. Nowadays, the Internet is rapidly evolving to become an equally efficient platform for multimedia content delivery. The part of Internet traffic due to multimedia applications such as Voice over IP, Video over IP, YouTube, IPTV, peer-to-peer video distribution softwares such as, to name few, Coolstreaming and Joost is ever increasing. A key difference between time-insensitive data traffic and time-sensitive traffic generated by applications such as VoIP or real-time video is that, while a data sending rate can be modulated to match the network available bandwidth, a real-time audio/video sending rate must follow the source rate. For these reasons data traffic is elastic and is carried on the TCP, which implements congestion control, whereas real-time traffic is inelastic and is carried on UDP.

Although in principle time-sensitive audio/video applications generate inelastic traffic because, due to time-constraints, flows cannot reduce their bandwidth requirements in the presence of congestion as TCP does, in practice well-designed time-sensitive applications must adapt to network available bandwidth at least at some extent. The way this goal can be obtained is by using a congestion control algorithm along with a scalable video codecs that adapts video quality, frame rate and picture size to match both the QoS requirements and network available bandwidth [18].

Differently from TCP flows that continuously probe for network capacity via the Additive Increase Multiplicative Decrease paradigm, the throughput of the flows originated by means of a scalable video codec is always bounded by the maximum and minimum bitrate achievable by the specific codec.

YouTube is an example of video distribution system that employs the TCP to generate elastic traffic. In particular, the video stream is buffered at the receiver for a while before the playing is started. In this way, short-term mismatch between the source video rate and the network available bandwidth are averaged out and masked by the playout buffer. On the other hand, Skype is one of the most prominent example of applications providing unicast Audio/Video calls over UDP. Skype is a closed source application which supports VoIP calls and one-to-one video conferencing over unicast UDP flows. Skype Audio employs several audio codecs such as G729, SVOPC, iSAC, iLBC, whereas Skype Video employs the VP7 codec provided by On2[1].

In literature there are several papers that propose to design new transport protocols tailored to transport multimedia content. A review of these protocols along with a proposed one can be found in [10]. Among these protocols,

[1]On2 Truemotion VP7 codec, http://www.on2.com/

NOSSDAV '08 Braunschweig, Germany

the only congestion control for multimedia flows that has been proposed for IETF standardization is the TCP Friendly Rate Control (TFRC) [12] [14]. Implementation of TFRC is complex since it requires ad-hoc tuning of many parameters. For example, to enable a VoIP application, it has been necessary to propose a variant [9]. For these considerations, the state of art of today running real-time applications such as Skype Audio/Video employs the UDP. Since the UDP does not implement congestion control functionalities, it is mandatory for a well-designed multimedia application to implement an efficient congestion control algorithm, otherwise the Internet would experience a congestion collapse as the one happened in the eighties before the introduction of TCP congestion control [17].

This work investigates how Skype Video behaves when sharing the Internet with other TCP and Skype Video flows. The goal is to determine the responsiveness of Skype Video to the unpredictable time-varying Internet bandwidth in terms of transient times needed to match the available bandwidth and fairness with respect to coexisting TCP and Skype flows. At the best of authors' knowledge, this is the first investigation of Skype Video. To the purpose, we have set up a local area network testbed in which it is possible to emulate wide area networks delays and set different traffic conditions, bottleneck capacities and queue size.

The rest of the paper is organized as follows: in Section 2 we summarize the related work; in Section 3 we summarize the knowledge made available to the public on the adaptive video codec used by Skype; in Section 4 we briefly describe the experimental testbed and the tools we have developed in order to carry out the experiments; in Section 5 we present and discuss the experimental results. Finally, Section 6 concludes the paper.

2. RELATED WORK

It is well-known that the best-effort Internet cannot provide guaranteed resources for real-time multimedia applications. The first attempt to address the problem is presented in [3] where authors show the benefit of implementing a very basic congestion control scheme in conjunction with the adaptive video codec H.261 in a video conferencing system. In the past years, the idea of applying congestion control to multimedia systems [8] has consolidated itself and it has led to several design efforts [10],[12],[14],[16].

One of the most prominent applications which implements real-time audio/video transmission over the Internet is Skype. An experimental investigation has revealed that Skype VoIP implement some sort of congestion control by varying the sending rate to match the network available bandwidth at some extent [7].

Other relevant papers on Skype can be grouped in the following categories: i) P2P network characterization; ii) perceived quality of the Skype VoIP flows; iii) identification of Skype flows.

First papers on Skype mainly focused on the characterization of the P2P network built by Skype in order to enlight, at least partially, interesting details on its architecture and on the NAT traversal techniques [2],[11].

Moreover, several studies have been carried out on the quality provided by the Skype VoIP calls in different scenarios by using metrics such as mean opinion score (MOS) and Perceptual Evaluation of Speech Quality (PESQ) [1], [6], [13] or by defining metrics based on packet level measurements such as round trip time, input rate and duration of the calls [5].

Another relevant aspect addressed in the literature is the detection of Skype flows, which is very important from the Internet Service Providers' point of view. Recently, it has been proposed a method to identify Skype traffic which uses two classifiers: one, which is based on applying the Chi-Square test to the payload of passively sniffed traffic, and the other one that is based on packet size and inter packet gap [4].

This work enriches the state of the art by characterizing the behaviour of Skype Video flows when accessing network paths with time-varying characteristics.

3. VIDEO CODEC EMPLOYED BY SKYPE

In this Section we summarize all the information available to the public concerning the video codec used by Skype Video that are reported in [15]. Since 2005 Skype employs the proprietary Video Codec TrueMotion VP7 provided by On2 in order to manage one-to-one videoconferencing. The codec supports real-time video encoding and decoding using a "datarate control" which adjusts frame quality, video resolution and number of frame per seconds to adapt to bandwidth variations. Moreover, the white paper [15] states that a model of the client buffer level is employed in order to control those variables, but no further details are provided. Regarding the bitrates produced by VP7, On2 claims to provide video transport starting from bitrates as low as 20 kbps; they do not provide any information on the maximum bitrate.

4. EXPERIMENTING WITH SKYPE VIDEO: THE SKYPE MEASUREMENT LAB

In order to investigate how Skype Audio/Video connections behave when network bandwidth changes over time, we have developed a measurement tool that allows real network experiments be deployed over one or more hosts. Figure 1 shows the testbed set up which is made of two real hosts: on each host one or more Skype applications are started with or without concurrent Iperf generated TCP traffic [2]. All packets generated by Skype and Iperf are routed to the user space ingress queues (using IPtables `QUEUE` target[3]), which allows several traffic measurements, such as incoming/dropping rates and packets size, to be performed. Moreover, the ingress queues have been used to set delays in order to emulate LAN or WAN scenarios and bottleneck bandwidth variations over time.

The throughput is defined as $\Delta sent/\Delta T$, the loss rate as $\Delta loss/\Delta T$ and the goodput as $(\Delta sent - \Delta loss)/\Delta T$, where $\Delta sent$ is the number of bits sent in the period ΔT, $\Delta loss$ is the number of bits lost in the same period. We have considered $\Delta T = 0.4$ s in our measurements.

It is of fundamental importance to perform experiments in a controlled environment in order to allow tests be reproducible. We provide reproducibility by employing a controlled LAN as a testbed and using the same video as input.

[2]http://dast.nlanr.net/Projects/Iperf/
[3]NetFilter: http://www.netfilter.org/

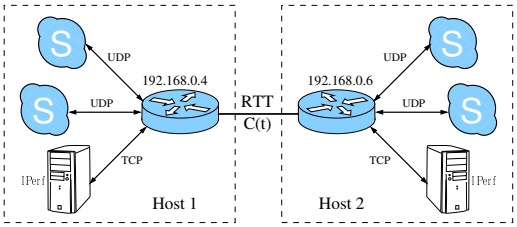

Figure 1: Experimental testbed

Using the input obtained by a webcam would always generate different bitrates, thus not allowing experiments to be reproduced.

Towards this end we have developed a software called *Skype Measurement Lab (SML)*, which allows a desired video source to be injected as input to Skype. In particular, we have modified `the` GStreamer plug-in `gst-fakevideo`[4], which generates a fake `/dev/video` device that simulates a video source (like a webcam) using a technique similar to the one employed by Skype Audio Dsp Hijacker[5]. Another important feature of the SML is the automatic logging of all the informations contained in the Skype technical call information tooltip, which is displayed when the "Technical Call Infos" option is enabled in the preferences. To the purpose we have modified the QT 4.3 user interface library[6] that is used by this client (freely available as source code) in order to periodically log all information contained in the call tool-tip, which includes among others: RTT, jitter, video resolution, video frame rate, estimated sent and received loss percentages.

The experiments have been run using the Linux Skype client version 2.0.0.27 and the standard *Foreman* YUV test sequence[7]. The audio input has been muted in order to analyze only the network traffic generated by video flows. From now on, the RTT of the connection is set at 50 ms and the queue size at the two hosts is set equal to the bandwidth delay product unless otherwise specified.

5. EXPERIMENTAL RESULTS

The main goal of this investigation is to show how Skype Video flows throttle their sending rates when step-like changes in available bandwidth occur and how Skype flows behave when concurrent TCP flows share the bottleneck. It is worth noticing that we consider step-like bandwidths because this is a simple and efficient practice in control theory when testing the dynamic behaviour of a system. Indeed, the step response of a system reveals interesting features of the system dynamics such as transient time and degree of stability.

In particular we are interested in revealing the transient dynamics of the Skype flows in response to bandwidth increase/decrease or to joining/leaving of TCP flows. It is reasonable to assume that the Skype video encoder [15] throttles the sending rate $r_s(t)$ by changing frame quality $q(t)$, video resolution $s(t)$ and number of frame per seconds (frame rate) $f(t)$ based on feedback reports sent by the receiver as it is depicted in Figure 2. Also, it is reasonable to

[4]http://code.google.com/p/gstfakevideo/
[5]Skype DSP hijacker: http://195.38.3.142:6502/skype/
[6]QT 4.3: http://trolltech.com/products/qt
[7]http://www.cipr.rpi.edu/resource/sequences/sif.html

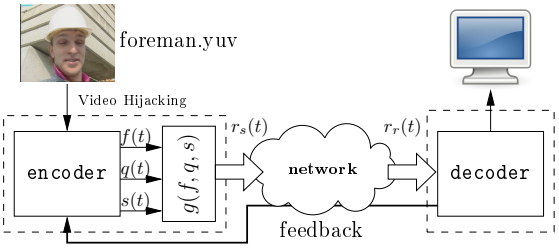

Figure 2: Model of the Skype Video rate adaptation scheme

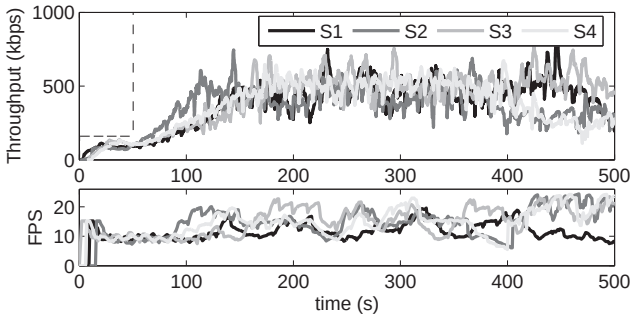

Figure 3: Skype Video response to a step change in available bandwidth at $t = 50\,s$

conjecture that the feedback variables used to throttle $q(t)$, $s(t)$ and $f(t)$ are available bandwidth, loss rate $l(t)$ and jitter $j(t)$. Throughout the discussion of the experimental results we will illustrate the effect of variable network conditions on the three control variables throttled by Skype.

5.1 Skype Video response to a step variation of available bandwidth

We start by investigating the behaviour of one Skype flow accessing a bottleneck link whose bandwidth capacity changes following a step function with minimum value $A_m = 160$ kbps and maximum value $A_M = 2000$ kbps. No concurrent traffic is injected. Figure 3 shows throughput, frame rate and jitter dynamics obtained by repeating four experiment runs. The video flow starts sending at a very low rate and achieves a steady state sending rate of roughly 80 kbps, well below the available bandwidth of 160 kbps. When the available bandwidth increases at $t = 50$ s , the sending rate reaches an average bitrate which is slightly below 450 kbps , after a quite long transient time of roughly 100 s.

Now, let us focus our attention on the three variables $f(t)$, $q(t)$ and $s(t)$ that are throttled by the video codec [15] to match the network available bandwidth. First of all, in the four experiments the resolution $s(t)$ of the videos produced by Skype was set at 320×240 pixels and kept unchanged throughout all the experiments. For what concerns the time behaviour of the frame rate, the initial value of $f(t)$ is always found to be 15.2 fps, then $f(t)$ decreases to around 10 fps in less than 10 s. After the step increment of the available bandwidth, $f(t)$ starts to increase at around $t = 85$ s and then it oscillates around the value of 15 fps. Moreover, the sending rate $r_s(t)$ starts to increase at $t = 50$ s whereas the value of $f(t)$ remains roughly constant in the time inter-

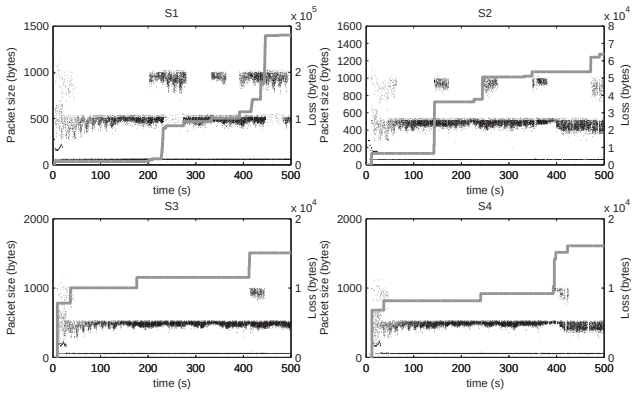

Figure 4: Packet size (black points) and cumulative losses (grey lines) of 4 Skype flows in response to a step change in available bandwidth at $t = 50$s

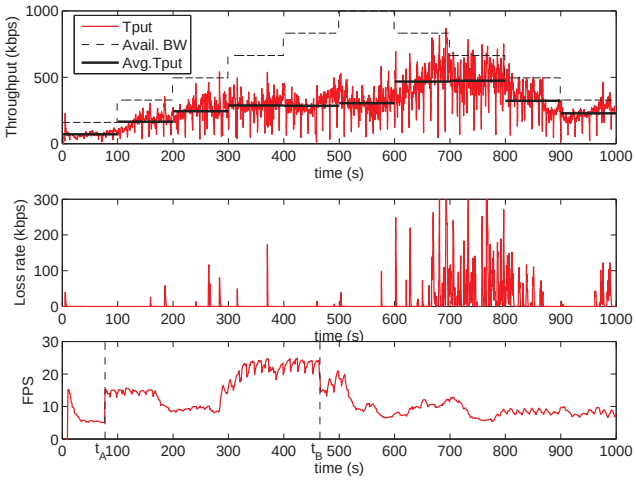

Figure 5: Skype Video flow in response to a time-varying available bandwidth

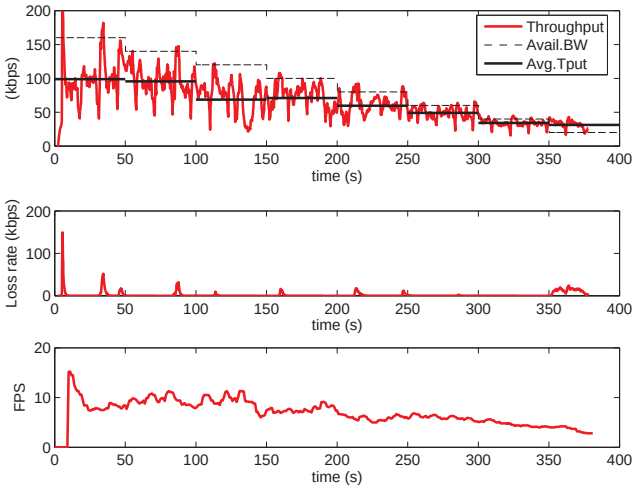

Figure 6: Skype Video response to an available bandwidth starting at 160 kbps and decreasing to 20 kbps

val [50, 85] s. This can be explained by looking at Figure 4 that shows packet sizes and cumulative losses of the four experiment runs: the packet size increases in the time interval [50, 85] s whereas $f(t)$ is left almost unchanged, which means that the increment of the sending rate is due to an improved quality $q(t)$.

Moreover, Figure 4 reveals an interesting correlation between packet size and packet losses: every time a large loss event occurs (marked by a large step in the cumulative line shown in the Figure 4) the packet size doubles, thus meaning that Skype employs a FEC scheme to counteract packet losses. On the other hand, Skype does not trigger the increase in packet size when the entity of the loss is considered negligible as it can be inferred by looking at the S3 plot at time $t = 177\,s$, which shows that a small step increase in the cumulative loss curve does not doubles the packet size.

To summarize, the main result of this first investigation is that a Skype Video flow produces a sending rate that achieves the maximum value of around 450 kbps and employs FEC mechanism to counteract large packet losses.

5.2 Skype response to a staircase variation of available bandwidth

In this scenario we aim at investigating how a Skype Video flow adapts to small step-like increment/decrement decreases in available bandwidth. To the purpose we start by allowing the available bandwidth to vary in the range [160, 1000] kbps. By using the knowledge on transient times that we have gathered in the previous scenario, we set bandwidth variations to occur every 100 s in order to let sending rates to extinguish their transients. In particular, in the first half of the experiment, the available bandwidth increases every 100 s of 168 kbps, whereas, in the second half, it decreases of the same amount every 100 s.

Figure 5 shows that Skype Video flow is somewhat slow in reaching the steady state since the maximum sending rate is achieved only at time $t = 700$ s when the second half of the experiment is already started. Moreover, in the first half of the experiment, losses are negligible and the average throughput is around 300 kbps, a value that is well below the available bandwidth that goes up to 1000 kbps.

Regarding the frame rate, after an initial value of $f(t)$= 15.2 fps, it decreases until $t = t_A$ when it suddenly increases its value again to 15.2 fps. The sudden increase in the frame rate occurs in correspondence to a change in the video resolution $s(t)$ from 320×240 to 160×120. The frame rate is kept unchanged to this value until time $t = t_B$ when the resolution switches back to 320×240 and the frame rate is set again to 15.2 fps.

We have run a similar experiment in which the available bandwidth varies from 160 kbps down to 20kbps in order to investigate how Skype flows are able to match a thin link capacity. Figure 6 shows that the sending rates are able to follow bandwidth reductions until the capacity drops to 40 kbps. In this condition a minimum frame rate around 5 fps is measured.When the available bandwidth shrinks at 20 kbps, which is the minimum declared bitrate of the Skype video codec [15], the video call is dropped at $t = 375$ s probably because Skype detects a very large packet loss percentage.

The overall conclusion of this test is that Skype Video

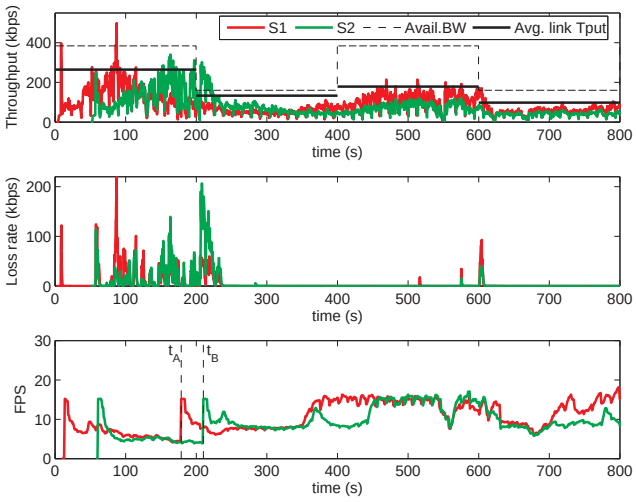

Figure 7: Response of two concurrent Skype Video flows to a square wave available bandwidth

response to bandwidth increment is somewhat slow, which means that Skype is not effective to take all the available bandwidth thus losing the possibility of delivering videos at the highest possible quality. In this test the available bandwidth reaches the value of 500 kbps in 200 s and then outpace this value but the Skype video sending rate maintains an average sending rate of only 300 kbps. On the other hand, the test has shown that Skype Video is able to shrink the sending rate to match a thin available bandwidth as low as 40 kbps.

5.3 Two Skype Video flows over a square wave available bandwidth

In the previous test we have shown how a single Skype Video flow reacts to variable network conditions such as a sudden drop/increase of the available bandwidth. In this test we aim at investigating the effect of multiple video flows on the stability of the network. To the purpose, we set up a scenario in which one Skype Video flow S_1 is started at $t = 0$ and a second flow S_2 is started at $t = 50$ s . The available bandwidth varies as a square wave of period $T = 400$ s with a maximum value $A_M = 384$ kbps and a minimum value $A_m = 160$ kbps . We have selected $A_M = 384$ kbps since this is the downlink capacity of an UMTS link and is smaller than the maximum average sending rate of Skype Video, which we have measured to be around 450 kbps. In this way we are sure that the two Skype Video flows can create congestion on the bottleneck. Again, we have set $A_m = 160$ kbps, since using a lower value calls were dropped. Figure 7 shows that, at the beginning, the first flow increases its sending rate as we have already shown in previous experiments. Moreover, the rate is kept increasing also when the second Skype flow joins the bottleneck at $t = 50$ s. However, for $t > 90$ s the first flow S_1 starts to leave bandwidth to S_2 that in turn increases its sending rate until the first bandwidth drop occurs at $t = 200$ s. It can be seen that S_2 generates a very high and persistent loss rate which lasts around 30 s with an average of 80 kbps.

Figure 7 also shows the average throughput of the link during each time interval where the bandwidth is kept constant. In particular, the channel link utilization is 68% for $t \in [0, 200]$ s, 83% for $t \in]200, 400]$ s, 46% for $t \in]400, 600]$ s and 61% for $t \in]600, 800]$ s.

It is important to note that when the available bandwidth increases again up to 384 kbps at $t = 400$ s, the two Skype flows do not increase their sending rate thus not taking the opportunity to send video at the best possible quality. For what concerns fairness issues, the two flows share the bottleneck in a fair way (the Jain fairness index is 0.97). Regarding the video resolution, the first Skype flow S_1 decreases $s(t)$ from 320×240 to 160×120 at $t = t_A$, which is before the bandwidth drop, whereas S_2 decreases $s(t)$ after the bandwidth drop at $t = t_B$.

Again, the overall conclusion of this test is that Skype Video is not efficient in getting full bandwidth utilization thus losing the possibility of delivering a video with a higher quality.

5.4 One Skype Video flow with concurrent TCP flows

In the previous subsection we have investigated the behaviour of Skype Video flows in the presence of time-varying available bandwidth. In this subsection we focus on the Skype Video behaviour when the network is shared with TCP traffic. We consider a link with a constant capacity of 384 kbps. A Skype Video call starts at $t = 0$, the first TCP flow starts at $t = 200$ s and a second one starts at $t = 400$ s. Figure 8 shows throughput and cumulative losses of Skype and TCP flows along with packet size and frame rate of the Skype flow.

When TCP1 enters the bottleneck, the Skype Video flow releases bandwidth by decreasing its sending rate. The two flows share the bandwidth fairly until $t = 250$ s when the Skype flow starts decreasing its sending rate leaving bandwidth to TCP1. However, Figure 8 shows that the steady state is not reached when TCP2 flow starts.

After the TCP2 flow joins the bottleneck link, we can observe that in the time interval $[400, 1000]$ s the bandwidth is shared in a somewhat fair way among the flows, except during the interval $[550, 700]$ s in which the Skype flow increases its bandwidth obtaining a significantly larger bandwidth share.

In order to understand the reason that triggers the increasing of the sending rate of the Skype flow, let us look at the packet size evolution shown in Figure 8. We can observe three different packet sizes produced by the Skype flow: the packets having size smaller than 70 *byte*, that we conjecture are control packets sent to the other peer to provide feedback; the packets with size between 350 and 530 bytes, which correspond to video packets with no redundancy; finally, packets with size between 700 and 1000 bytes, which are due to video packets with redundant information.

The Figure 8 shows that the Skype Video packet size increases in the time intervals $[120, 180]$s, $[375, 494]$ s and $[590, 681]$ s which indicates that Skype has increased the FEC. This is confirmed by the frame rate evolution that does not follow the sending rate increase in those intervals. It is worth noticing that the step change in the frame rate evolution that occurs at $t = 436$ s corresponds to a decrease in the video resolution from 320×240 to 160×120. The cumulative losses graph shown in Figure 8 clearly suggests that the increments in the FEC are triggered by the increasing of lost bytes (see also Section 5.1). In particular the Skype

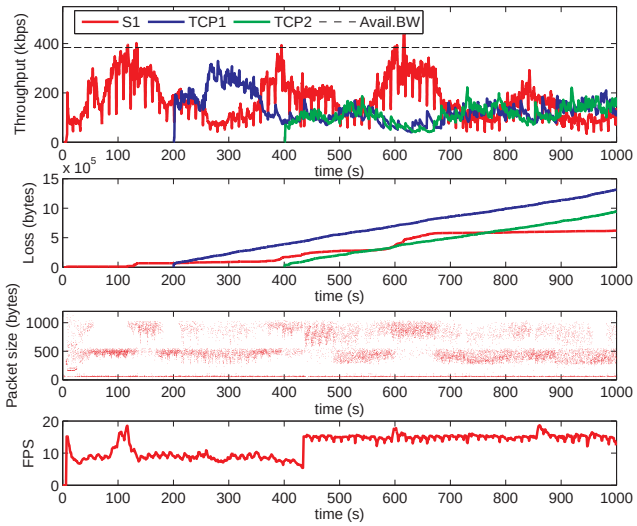

Figure 8: One Skype Video flow over a link with 384 *kbps* capacity with a two concurrent TCP flows started at $t = 200\,s$ and $t = 400\,s$

Table 1: Throughput, loss rate, loss ratio and channel utilization for the Skype and the two TCP flows

	Tput (kbps)	Loss rate (kbps)	Loss ratio	Channel util.
S1	162.5	6.0	3.7%	42.3%
TCP1	101.6	12.3	12%	26.4%
TCP2	102.3	12.6	12%	26.6%

flow loses 258000 bytes in the interval $[590, 681]$ s where the Skype flow exhibits an unfair behaviour with respect to the TCP flows. In order to evaluate how the Skype flow behaves when sharing the link with other TCP flows, Table 1 reports average values of throughput, loss rates, loss percentages and channel utilizations of all the flows for $t > 400\,s$. Reported results show that Skype takes a larger share of channel capacity, whereas the two TCP flows share the left out bandwidth equally.

The overall conclusion here is that Skype Video seems more aggressive than the TCP, because of the FEC action that seems to unresponsively increase the sending rate even when losses are experienced.

6. CONCLUSIONS AND FUTURE WORK

We have carried out an experimental investigation of Skype Video flows behaviour in the presence of time varying network conditions and TCP traffic. We have found that a Skype Video call uses the frame rate, the packet size and the video resolution in order to throttle its sending rate to match the network available bandwidth. The obtained results have shown that a Skype Video call roughly requires a minimum of 40 kbps available bandwidth to start and it is able to fill in a bandwidth up to 450 kbps. Thus it can be said that a video flow is made elastic through congestion control and adaptive codec within that bandwidth interval.

We have also measured that a Skype Video sending rate exhibits a large transient time when it keeps increasing to match an increment of the available bandwidth. Moreover, we found that in many scenarios a Skype video call refrains from fully utilizing all available bandwidth, which means that a video call is not performed at the best quality that a network would permit. Regarding coexistence with TCP flows, Skype Video seems more aggressive than the TCP because of the FEC action that seems to unresponsively increase the bandwidth even when losses are experienced.

7. REFERENCES

[1] R. Barbosa, C. Kamienski, D. Mariz, A. Callado, S. Fernandes, and D. Sadok. Performance evaluation of P2P VoIP application. *ACM NOSSDAV '07*, 2007.

[2] S. Baset and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. *IEEE INFOCOM '06*, Apr. 2006.

[3] J.-C. Bolot and T. Turletti. A rate control mechanism for packet video in the internet. In *IEEE INFOCOM '94*, pages 1216–1223, 1994.

[4] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi, and P. Tofanelli. Revealing skype traffic: when randomness plays with you. *ACM SIGCOMM '07*, pages 37–48, Aug. 2007.

[5] K. Chen, C. Huang, P. Huang, and C. Lei. Quantifying Skype user satisfaction. *ACM SIGCOMM '06*, Sept. 2006.

[6] W. Chiang, W. Xiao, and C. Chou. A Performance Study of VoIP Applications: MSN vs. Skype. *MULTICOM '06*, 2006.

[7] L. De Cicco, S. Mascolo, and V. Palmisano. An Experimental Investigation of the Congestion Control Used by Skype VoIP. *WWIC '07*, 4517:153, May 2007.

[8] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Trans. on Networking (TON)*, 7(4):458–472, 1999.

[9] S. Floyd and E. Kohler. TCP Friendly Rate Control (TFRC): The small-packet (sp) variant. *RFC 4828, Experimental*, 2007.

[10] L. A. Grieco and S. Mascolo. Adaptive rate control for streaming flows over the internet. *ACM Multimedia Systems Journal*, 9(6):517–532, Jun 2004.

[11] S. Guha, N. Daswani, and R. Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. *Proc. IPTPS '06*, Feb. 2006.

[12] M. Handley, S. Floyd, and J. Pahdye. TCP Friendly Rate Control (TFRC): Protocol Specification. *RFC 3448, Proposed Standard*, Jan. 2003.

[13] T. Hoßfeld and A. Binzenhöfer. Analysis of Skype VoIP traffic in UMTS: End-to-end QoS and QoE measurements. *Computer Networks*, 2007.

[14] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: congestion control without reliability. *ACM SIGCOMM '06*, Sept. 2006.

[15] On2 Technologies. TrueMotion VP7 Video Codec White Paper. 10 Jan. 2005.

[16] H. Schulzrinne, S. Casner, S. Frederick, and V. Jacobson. Rtp: A transport protocol for real-time applications. *RFC 3550, Standard*, 2003.

[17] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review*, 1988.

[18] S. Wenger. H. 264/AVC over IP. *IEEE Trans. Circuits and Syst. Video Technol.*, 13(7):645–656, 2003.