An Experimental Investigation of the Congestion Control Used by Skype VoIP*

Luca De Cicco, Saverio Mascolo, Vittorio Palmisano
{ldecicco, mascolo, vpalmisano}@poliba.it

Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari,
Via Orabona 4, Italy

Abstract. The explosive growth of VoIP traffic poses a potential challenge to the stability of the Internet that, up to now, has been guaranteed by the TCP congestion control. In this paper, we investigate how Skype behaves in the presence of time-varying available bandwidth in order to discover if some sort of congestion control mechanism is implemented at the application layer to match the network available bandwidth and cope with congestion. We have found that Skype flows are somewhat elastic, i.e. they employ some sort of congestion control when sharing the bandwidth with unresponsive flows, but are inelastic in the presence of classic TCP responsive flows, which provokes extreme unfair use of the available bandwidth in this case. Finally, we have found that when more Skype calls are established on the same link, they are not able to adapt their sending rate to correctly match the available bandwidth, which would confirm the risk of network congestion collapse.

1 Introduction

Skype is by far the most used VoIP application, with an ever growing user-base which today counts more than 8 million users. This explosive growth poses challenges to telecom operators and ISPs both from the point of view of business model and network stability. Regarding network stability, the issue here is to check if the growth of unresponsive non-TCP flows, i.e. flows without end-to-end congestion control, would impact the stability of the best-effort Internet that everyone knows.

Other important issues that have been addressed in recent literature on Skype are the employed peer-to-peer protocol and the evaluation of the QoS of VoIP calls placed using Skype [1,3].

The goal of this paper is to investigate how Skype reacts to network congestion, that is, how Skype manages to adapt its sending packet rate to match the network available bandwidth when competing with TCP flows or with multiple Skype calls placed over the same link.

* This work was partially supported by the MIUR-PRIN project no. 2005093971 "FAMOUS Fluid Analytical Models Of aUtonomic Systems"

The results we have obtained are twofold: i) Skype flows are somewhat elastic and they employ some sort of congestion control when coexisting with unresponsive flows; ii) Skype flows exhibit unresponsive behaviour when sharing the bandwidth with responsive TCP flows.

The paper is organized as follows: in Section 2 we present a brief analysis of the state of the art related to Skype; in Section 3 we describe the experimental testbed that has been set up in order to carry out our investigations; Section 4 describes the considered scenarios and presents the obtained results. Finally Section 5 concludes the paper.

2 Related work

The efficient transport of multimedia flows is an open issue and it is currently an hot topic as long as multimedia services are rapidly increasing their importance. In this area the voice over IP applications are taking an ever increasing relevance as it is shown by the success of Skype application for end users and by the large deployment of SIP-based networks.

In spite of this explosive growth it is not clear what will be the impact of VoIP traffic on the stability of the Internet when a very large amount of VoIP flows will populate the network. The main driver of Internet stability is the congestion control algorithm developed by V. Jacobson for the TCP [7]. For this reason many researchers have conjectured a congestion collapse in case VoIP flows don't employ a responsive congestion control algorithm [4].

As a consequence, several efforts have been carried out to design multimedia congestion control protocols that are TCP friendly, where friendliness here means that the audio flows will share the network bandwidth with TCP flows fairly.

The TCP Friendly Rate Control (TFRC) protocol is currently being discussed within the IETF as a possible congestion control algorithm for multimedia flows [6]. In particular the Small Packet version of TFRC has been proposed in order to be employed for VoIP applications [5]. In spite of this effort, as a matter of fact, all commercial audio/video applications run over UDP and we conjecture that they implement some congestion control algorithm at the application level. Other proposals for multimedia traffic are RAP [9] and TEAR [10].

A recent investigation [2] concludes that VoIP traffic does not harm network stability because of the user behaviour that would drop a call if an unacceptable quality is perceived due to congestion. In other terms, the user behaviour would provide an intrinsic stability mechanism for congestion avoidance. In particular, by carrying out simulations, authors infer that, by taking into account the user back-off behaviour, VoIP flows consume much less bandwidth than TCP flows and respond to congestion when network is overloaded. Differently from [2], in this paper we will investigate if the Skype VoIP application implements its own congestion control algorithm to match the available bandwidth. As it will be shown in this paper the answer is affirmative.

3 Experimental testbed

In order to investigate how Skype adapts to variations in the available bandwidth we have set up a local testbed using a measurement tool we have developed. In each host in Figure 1 we have routed all packets generated from Skype application to the ingress queues q_1 and q_2. The measurement tool allows delays, available bandwidth and buffer size of each queue be set by the user. It is worth noticing that the connection implemented is strictly equivalent to a connection made by using a Dummynet-like router [11], the only difference being that in this way we can use two hosts instead of three.

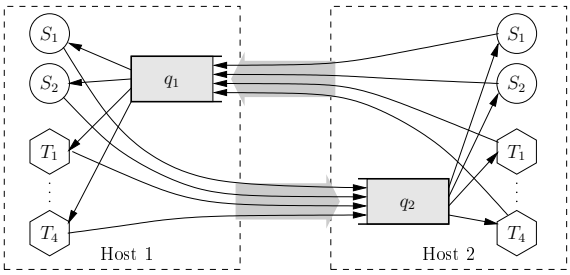


Fig. 1. Experimental testbed

On each host we have installed Skype (S_1 and S_2) and `iperf` ($T_1, \ldots, T_4$) [14] in order to generate TCP flows and we have collected logfiles by tracing the per-flow data arriving to and departing from the queue. By comparing data at the input of the queue and at its output, we have been able to compute packet drop rates and goodputs for Skype and TCP flows. Goodput, throughput and loss rate are defined as follows:

$$\text{goodput} = \frac{\Delta sent - \Delta loss}{\Delta T}; \text{ throughput} = \frac{\Delta sent}{\Delta T}; \text{ loss rate} = \frac{\Delta loss}{\Delta T}$$

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 $\Delta T = 0.4\,\text{s}$ in our measurements.

Finally, it is worth noticing that Skype flows are generated using always the same audio sequence by hijacking audio I/O by using [15]. From now on, the RTT of the connection is set at 100 ms and the queue size is set equal to the bandwidth delay product unless otherwise specified.

4 Investigating the Skype congestion control

In order to understand how Skype behaves in the presence of congestion we start by considering a step-like time-varying available bandwidth. Considering a step-like input is common practice in control theory when testing the dynamic

behaviour of a system [8]. In particular we start by considering square-wave available bandwidths characterized by different periods in order to test not only the Skype capability to match the available bandwidth but also the transient time required for the matching.

Before starting to report our results, it is worth noticing that Skype employs the adaptive codecs iSAC and iLBC both developed by Global IP Sound [12,13] to provide sending rate adaptation capability.

4.1 Case 1: One Skype flow over a square wave form available bandwidth

This scenario aims at investigating how Skype sending rate reacts to sudden changes of available bandwidth in order to infer if it employs some sort of congestion control. In order do this, we have used a technique that is often employed in system identification, i.e. we have used an available bandwidth that varies as a square wave with maximum value $A_M = 160\,\mathrm{kb/s}$ and minimum value $A_m = 16\,\mathrm{kb/s}$ (see Figure 2).

We have considered two periods for the square wave form in order to identify how fast is the Skype response to bandwidth changes.

We have run the first experiment by setting the period of the square wave equal to 200 s, which happened to be large enough to show all the transient dynamics.

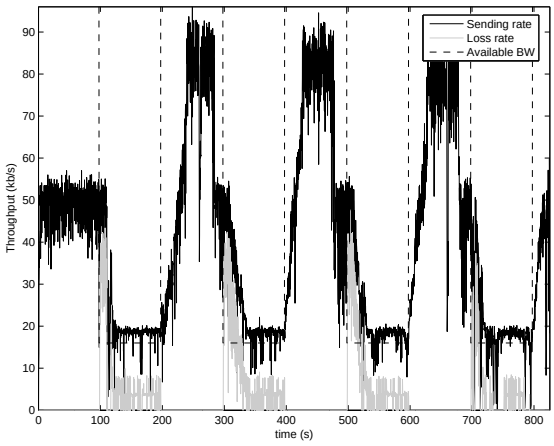


Fig. 2. Sending rate and Loss rate in the presence of square wave available bandwidth of 200 s period

Figure 2 shows that Skype decreases the sending rate when the link capacity drops from the high value A_M to the low value A_m. It is worth noticing that the Skype flow takes approximately 40 s to track the available bandwidth during

which it experiences a significant loss rate. From Figure 2 we also argue that the Skype reaction is triggered by the high loss rate.

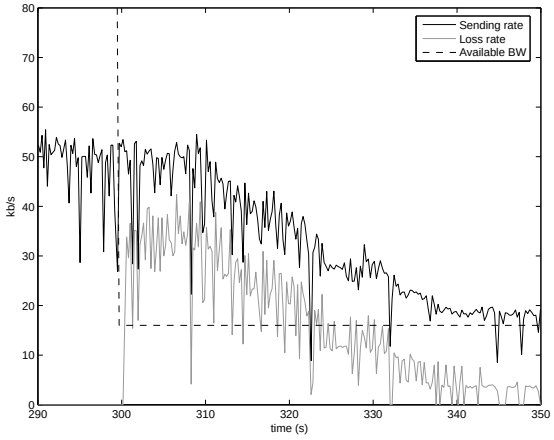


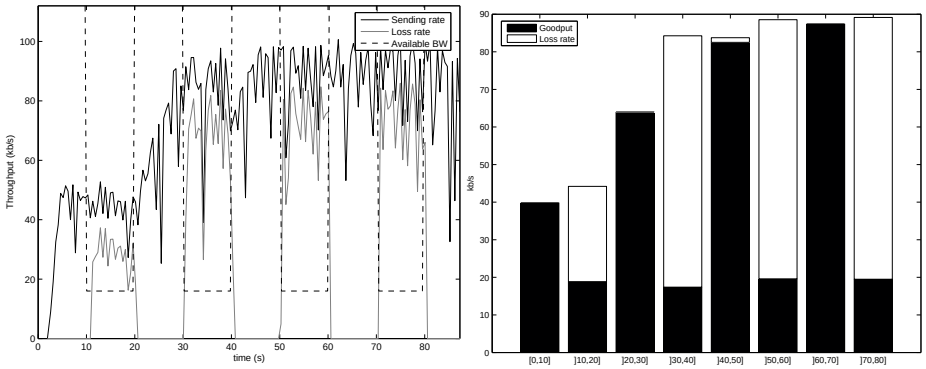
Fig. 3. Zoom of the Figure 2 around the bandwidth drop at $t = 300\,\mathrm{s}$

To provide a further insight, Figure 3 shows a zoom of Figure 2 in the time interval $[290, 350]$ in order to look at what happens when the link bandwidth drops from $120\,\mathrm{kb/s}$ to $16\,\mathrm{kb/s}$ at $t = 300\,\mathrm{s}$. It can be viewed that when the available bandwidth drops, the loss rate increases to a peak value of $35\,\mathrm{kb/s}$ whereas the sending rate reduces to less than $20\,\mathrm{kb/s}$ in $40\,\mathrm{s}$.

By observing this behaviour it may be conjectured that Skype implements a form of congestion control algorithm that reduces the sending rate when a high packet loss rate is measured.

When the link capacity increases (i.e. at $t = 400\,\mathrm{s}$) the input rate goes up to $90\,\mathrm{kb/s}$ again in $40\,\mathrm{s}$. Therefore it seems that Skype reacts to bandwidth variations with a transient dynamics that lasts $40\,\mathrm{s}$. In order to validate this finding we consider the same square wave with a period of $20\,\mathrm{s}$. Figure 4 (a) shows that in this case Skype is not able to match the available bandwidth thus provoking congestion loss rates up to $80\,\mathrm{kb/s}$.

From this experiment it results that Skype is able to match the available bandwidth within a transient time of $40\,\mathrm{s}$. We conjecture that this somewhat slow response to bandwidth variation is due to the fact that a sudden variation in the encoding bitrate would have negative effects on user perceived quality. However, such a slow response to congestion episodes is likely to cause unfriendliness when TCP flows share the available bandwidth with Skype flows (see Section 4.3 for details). Moreover, the high packet loss rate experienced in this scenario and shown in Figure 4 (b) may not guarantee perceived quality either.



(a) (b)

Fig. 4. (a) Sending rate and Loss rate with a square wave available bandwidth of 20 s period; (b) Goodput and Loss rate during time intervals at constant available bandwidth

4.2 Case 2: One Skype flow in the presence of variable bandwidth

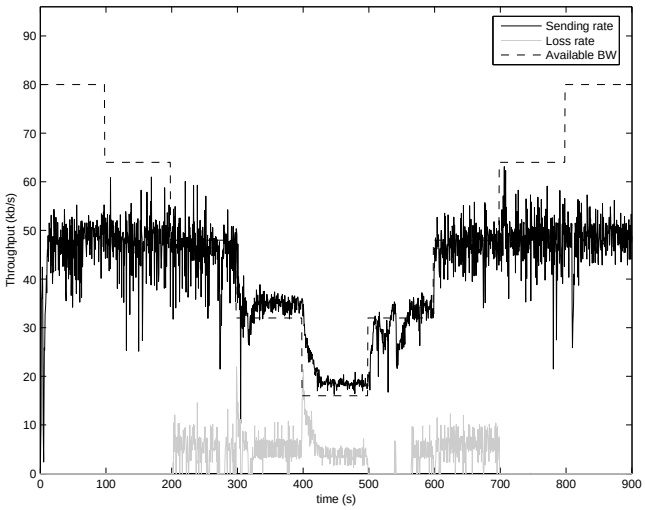
This scenario is aimed at investigating how a Skype's sending rate reacts to small step-like increases and decreases of available bandwidth. To the purpose we allow the available bandwidth to vary in the range $[16, 80]$ kb/s, which are the minimum and maximum Skype encoder bitrates that we have measured in our experiments.

By using the knowledge about transient times that we have gathered in Section 4.1, we set bandwidth variations to occur every 100 s in order to let sending rates extinguish transients. In particular, the available bandwidth is set as follows: in the first half of the experiment, the flow experiences a 16 kb/s bandwidth drop every 100 s, whereas, in the second half, a bandwidth increase of 16 kb/s occurs every 100 s. This bandwidth variation pattern is particularly suited to test how the Skype sending rate adapts to a sequence of drops and increases.

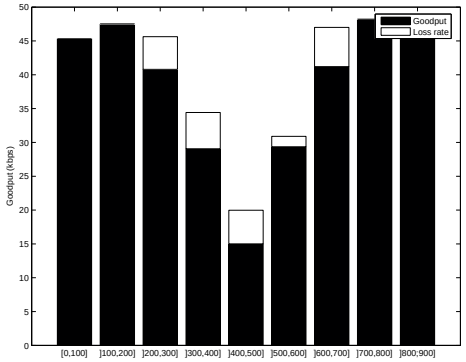
Figure 5 (a) clearly shows how Skype reacts to sudden drops in available bandwidth: when the loss rate is under a given threshold (such as in the time interval $[200, 300]$) the sending rate is kept unchanged until a burst of losses is detected, which triggers a sending rate reduction.

During the bandwidth increasing phase (for $t > 500$ s), the sending rate is able to match the available bandwidth without congesting the link. It is worthy to focus on the time interval $[600, 700]$: we conjecture that, similarly to what happens in the time interval $[200, 300]$, the sending rate is kept constant since measured loss rate is not considered harmful for the user perceived quality by the codecs employed by Skype [12,13].

Finally Figure 5 (b) shows goodputs and loss rates measured in each interval during which the bandwidth is constant.



(a)



(b)

Fig. 5. (a) Sending rate and loss rate in kb/s of a Skype connection over a time-varying available bandwidth; (b) Goodput and loss rate measured during time intervals at constant available bandwidth

4.3 Case 3: One Skype flow with one concurrent TCP connection

The Transmission Control Protocol is by far the most used transmission protocol in the Internet, so it is very important to evaluate how Skype behaves when it shares the network with TCP flows.

One Skype flow and one TCP flow.
In this test we consider a link with a capacity of 56 kb/s. We first start a TCP connection at $t = 0$ and then a Skype call at $t = 70$ s that lasts 200 s.

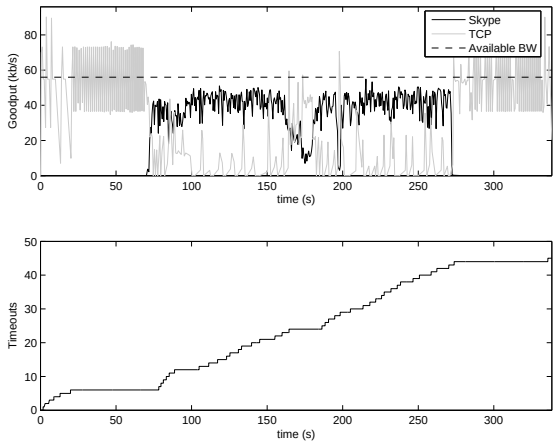


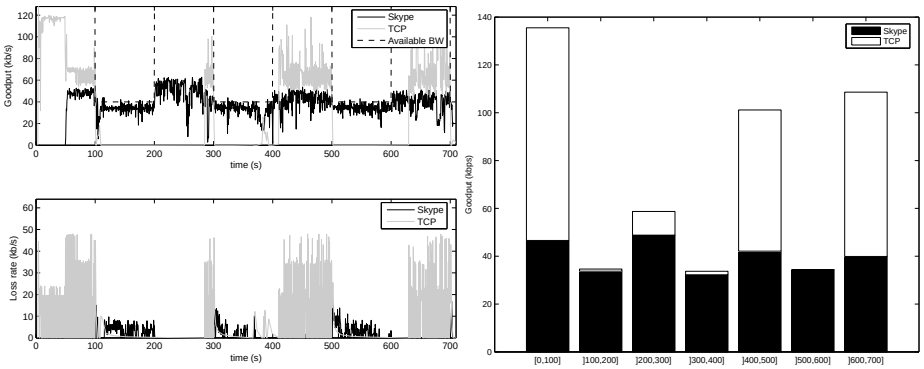
Fig. 6. Goodput of Skype and TCP flows; Number of timeouts of the TCP flow

Figure 6 shows that when the Skype flow enters the bottleneck, the TCP connection is not able to get a bandwidth share anymore. In particular, Figure 6 shows that when the Skype flow is ON, the TCP flow experiences around 40 timeouts! This result seems to contradict results obtained before when we have shown that Skype matches the available bandwidth.

The reason is that the TCP congestion control reacts to loss events by halving its congestion window whereas, on the other hand, Skype flows adapt to the available bandwidth slowly. Therefore, even though the TCP congestion control continuously probes for the link bandwidth using its additive increase phase, it is not able to get any significant bandwidth share.

One Skype flow and one TCP flow over a square wave available bandwidth.
In this scenario we consider a square wave available bandwidth with 50% duty cycle, 200 s of period, the maximum value $A_M = 160$ kb/s and minimum value $A_m = 40$ kb/s. Moreover, the TCP flow starts at $t = 0$, whereas the Skype call is placed after 50 s.

Figure 7 (a) shows that when the available bandwidth is high, both TCP and Skype are able to use the link because the TCP is allowed to take the left over capacity. On the other hand, when the available bandwidth is low, the TCP flow is not able to get any share, and its goodput is close to zero (see Figure 7 (b)).



(a) (b)

Fig. 7. (a) Goodput and loss rate for the Skype and the TCP flows in the presence of a square wave available bandwidth; (b) Goodput of Skype and TCP flows in time intervals where the available bandwidth is kept constant

4.4 Skype with 4 concurrent TCP flows

This scenario aims at investigating how Skype behaves when multiple TCP flows share the link with one Skype flow. In this experiment, the available bandwidth is set at 120 kb/s and the four TCP flows will join the link following the timing pattern depicted in Figure 9 (a).

The Figure 8 depicts goodput as a function of time for the Skype flow and for the four TCP flows. It can be noticed that Skype doesn't adapt its sending rate when a new TCP flow joins the bottleneck; on the other hand, TCP flows adapt their rate in order to avoid congestion on the link as it is expected. Figure 9 (b) clearly shows that Skype's goodput is kept unchanged during all the time, while TCP flows share the left available bandwidth.

4.5 Two Skype connections over a square wave available bandwidth

In this scenario we consider a square wave available bandwidth with 50% duty cycle, 200 s of period, the maximum value $A_M = 144$ kb/s and minimum value $A_m = 64$ kb/s. We start two Skype calls over the same link in order to investigate how Skype clients share the available bandwidth. The first Skype call starts at $t = 0$, whereas the second one at $t = 25$ s. We have run several experiments and we report two of them showing different behaviours.

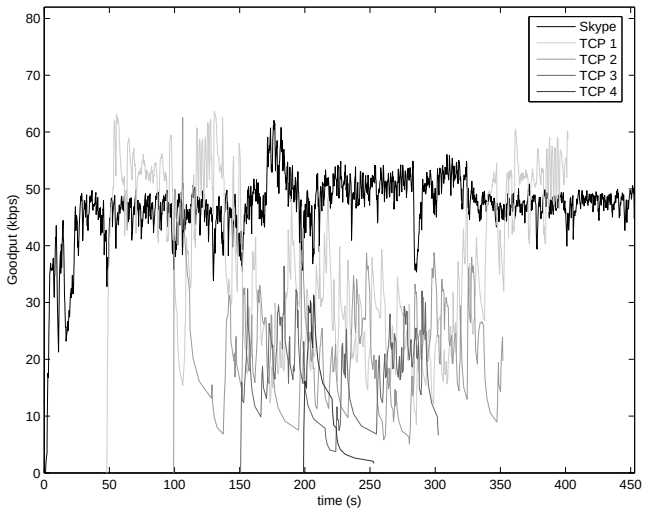
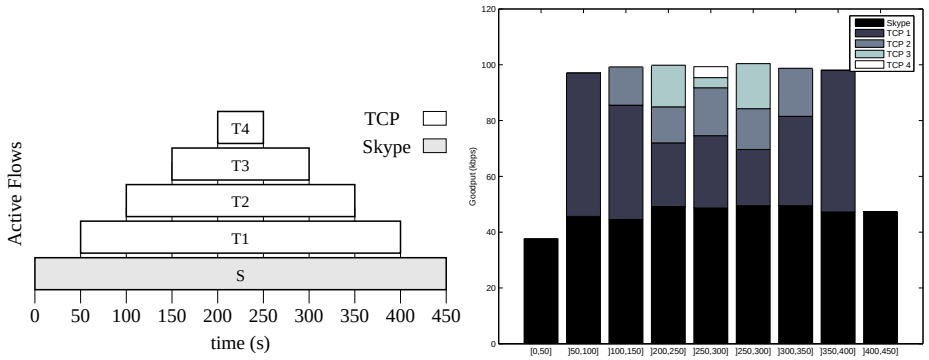


Fig. 8. Goodput of the flows



(a) (b)

Fig. 9. (a) Time intervals during which Skype(s) and TCP flows (T_i) are active; (b) Goodput of the flows during each time interval.

Experiment 1. Figure 10 (a) depicts sending and loss rates for both Skype flows. It can be noticed that except for the first period, i.e. for $t > 200\,\mathrm{s}$, the first flow increases its sending rate when the available bandwidth is high, whereas the second flow maintains its sending rate at 30 kb/s throughout all the experiment. When the bandwidth drop is experienced the first flow decreases its sending rate in about 40 s, as we have seen before.

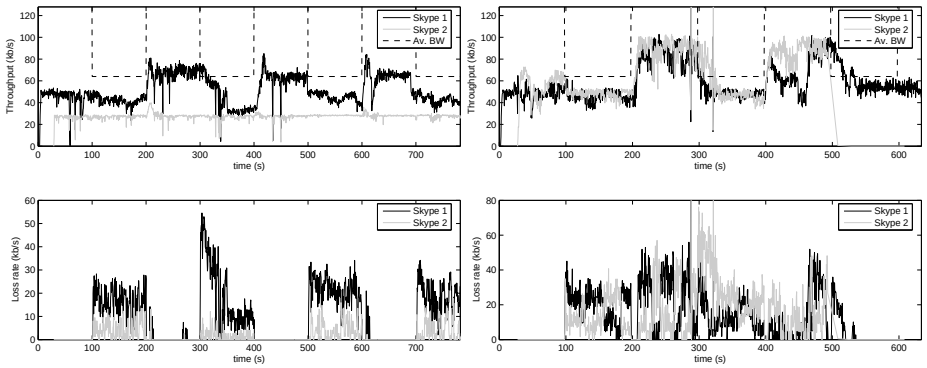


Fig. 10. Sending and loss rates for two Skype flows sharing the bottleneck. (a) Experiment 1; (b) Experiment 2.

This experiments seems to indicate that the two Skype flows behave differently even if they share the same link. Moreover it can be seen that, when the bandwidth is high, the goodput achieved by the first flow is much higher than the one achieved by the second flow.

Experiment 2. Figure 10 (b) shows sending and loss rate for the second experiment we have run. In this case, differently from what has happened in the first experiment, the two flows experience similar sending rates, but congesting the link as it is shown by the high loss rates during the interval $t \in [100, 500]$. In particular, by focusing on the bandwidth drop that occurs at $t = 300\,\mathrm{s}$ it can be shown that both the Skype flows reduce their sending rate and after $\sim 30\,\mathrm{s}$ they maintain a sending rate of 50 kb/s, which provokes congestion on the link, i.e. high value of the loss rate.

5 Conclusions

We have carried out an experimental investigation of Skype VoIP in a controlled environment in order to find out if and how Skype implements congestion control to match network available bandwidth.

By examining results of our experiments we have found that Skype implements some sort of congestion control algorithm. However, the reaction speed of this algorithm revealed to be very slow. For this reason Skype has shown two main drawbacks: i) Large packet drops rates during the transients following a bandwidth change; ii) unresponsive behaviour when coexisting with responsive flows such as TCP that provoke extreme unfair use of the limited available bandwidth.

Finally, we have also found that when more Skype calls are established on the same link, they are not able to adapt their sending rate to match correctly the available bandwidth, which would confirm the risk of network congestion collapse.

References

1. S. A. Baset and H. Schulzrinne. "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol". In *Proceedings of IEEE Infocom* '06, Barcelona, Spain, Apr. 2006.
2. T. Bu, Y. Liu, D. Towsley, "On the TCP-Friendliness of VoIP Traffic", In *Proceedings of IEEE INFOCOM '06*, Barcelona, Spain, Apr. 2006.
3. K. Chen, C. Huang, P. Huang, C. Lei, "Quantifying Skype User Satisfaction", in *Proceedings of SIGCOMM '06*, Pisa, Italy, Sep. 2006.
4. S. Floyd, K. Fall, "Promoting the use of end-to-end congestion control in the Internet", *IEEE/ACM Transaction on Networking*, vol.7, no. 4, pp. 458-472, 1999.
5. S. Floyd, E. Kohler. "TCP Friendly Rate Control (TFRC): the Small-Packet (SP) Variant". IETF draft, 20 Nov. 2006.
6. M. Handley, S. Floyd, J. Pahdye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification". RFC 3448, Proposed Standard, January 2003.
7. V. Jacobson, "Congestion avoidance and control", ACM SIGCOMM Computer Communication Review, 1995.
8. S. Mascolo, "Congestion control in high-speed communication networks using the Smith principle", Automatica, vol. 35, no. 12, Dec. 1999, pp. 1921-1935. Special Issue on "Control methods for communication networks".
9. R Rejaie, M Handley, D Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet", In *Proceedings of IEEE INFOCOM '99,* vol. 3, pp. 1337-1345, 1999.
10. I. Rhee, V. Ozdemir, and Y. Yi, "TEAR: TCP Emulation at Receivers - Flow Control for Multimedia Streaming", Dept. of Comp. Sci., NCSU, Tech. rep., Apr. 2000.
11. L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols", ACM SIGCOMM Computer Communication Review, 1997
12. Global IP Sound. "iSAC codec datasheet". [Online] Avilable: http://www.globalipsound.com/datasheets/iSAC.pdf
13. Global IP Sound. "iLBC codec datasheet". [Online] Avilable: http://www.globalipsound.com/datasheets/iLBC.pdf
14. `iperf` [Online]. Available: http://dast.nlanr.net/Projects/Iperf/
15. Skype DSP hijacker [Online]. Available: http://195.38.3.142:6502/skype/