A Mathematical Model of the Skype VoIP Congestion Control Algorithm

Luca De Cicco, Saverio Mascolo and Vittorio Palmisano

***Abstract*— Nowadays the Internet is changing from being only an efficient platform for data delivering to become a major platform for many other applications such as Voice over IP. The stability of the traditional data Internet is due to the congestion control algorithm developed by V. Jacobson for the TCP. However, the TCP congestion control is not optimal for VoIP applications because of its retransmission mechanism and additive increase multiplicative decrease sliding window control. As a consequence, VoIP applications often employ proprietary and hidden congestion control algorithms executed over the UDP protocol. In this paper we focus on Skype audio, which is the most known and used VoIP application, in order to derive a mathematical model of its congestion control algorithm. To the purpose, the controller input/output variables are first identified and then their dynamic relation is described in the form of a hybrid automaton. Main findings are: (1) Skype does not implement a delay based control; (2) the loss ratio is the main input that affects the sending rate; (3) the sending rate matches the available bandwidth with a finite error.**

I. Introduction

Skype is by far the most used Voice over IP (VoIP) application, with an ever growing user-base which today counts more than 10 million concurrent users. This explosive growth poses challenges to telecom operators and ISPs both from the point of view of business model and network stability.

Skype is a closed source application, which employs a proprietary protocol that is hidden by using AES encryption [1]. Some efforts have been spent so far to investigate the features of the protocol by using reverse-engineering techniques [1].

In this work we do no not address the Skype signalling protocol (call establishment, call teardown) already addressed in [2]. We instead focus on Skype congestion control, i.e. on how Skype adapts its sending rate in the presence of variable network conditions. The issue is complex for several reasons: (1) the protocol behaviour is hidden by AES encryption; (2) the input variables that drive the controller are unknown; (3) it is very much reasonable to conjecture that the controller implements a complex switching dynamics due to the use of if-the-else decisional blocks.

We will start by considering Skype as it is: a pure black box of which we do not know the controller inputs and we can't inspect the feedback packets because they are encrypted.

In order to carry out the investigation, we have set up a local testbed where different network parameters such as link delays and capacities, loss rates and queue sizes can be varied. All those variables are candidate for being inputs to the controller. In particular, to identify the controller inputs we have designed a proper set of experiments to evaluate the response of Skype to any input. Of course, we expect that some of these inputs will have stronger effects on the output of the controller (like the available bandwidth and the packet loss rate) whereas others will have weaker ones.

In [3] we have shown that Skype reacts to network congestion by reducing the sending rate, thus being able to match the link capacity to some extent. In this work we propose a mathematical model of how Skype tracks the network available bandwidth. In particular, we identify the controller inputs and outputs and we propose a hybrid automaton to describe the controller switching dynamics.

The rest of the paper is organized as follows: in Section II we present an overview of the literature on modeling congestion control in data networks, which mainly concerns the TCP congestion control; in Section III we describe the experimental testbed that has been set up in order to investigate the Skype congestion controller; in Section IV we show the effects of the candidate inputs on the output of the controller; in Section V we present a mathematical model of Skype sending rate when congestion occurs and we derive a hybrid switching dynamic model of a Skype flow when accessing a bottleneck; a stability analysis is also carried out. Finally Section VI concludes the paper and outlines further research work.

II. Related work

Nowadays, the efficient transport of multimedia flows is a hot issue due to the booming of applications based on multimedia content delivery. In this area Voice over IP plays a key role 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 multimedia traffic on the stability of the Internet when a very large amount of this traffic will populate the network. The main driver of Internet stability is the TCP congestion control algorithm developed by V. Jacobson for data delivery [4]. However, TCP is not well suited for audio/video delivery due to retransmissions mechanism and sliding window control. As a consequence, audio and video applications are run over the UDP.

Luca De Cicco is a PhD student of Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via Orabona 4, Italy ldecicco@poliba.it

Saverio Mascolo is a Faculty member of Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via Orabona 4, Italy mascolo@poliba.it

Vittorio Palmisano is a PhD student in Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, Via Orabona 4, Italy vpalmisano@poliba.it

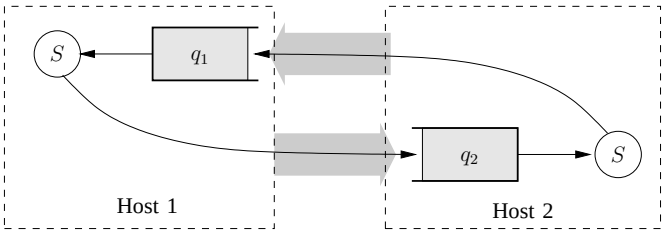


Fig. 1. Experimental testbed employed in the investigation

Mathematical models have played a major role in understanding the fundamental properties of large scale and complex communication systems [5]. Up to now the only congestion control algorithm for data networks that has been modelled is the standard TCP congestion control and its variants [6],[7],[8],[9],[10]. This is due to the fact that the proposed TCP congestion control algorithms are fully disclosed and well described in scientific literature and in standardization bodies such as the IETF.

On the other hand, in the case of audio or video applications over the Internet, hidden proprietary protocols are executed over the UDP protocol. The only exception is represented by the TCP Friendly Rate Control (TFRC) protocol which is currently being discussed within the IETF as a possible congestion control algorithm for multimedia flows [11][12]. In particular the Small Packet version of TFRC has been proposed in order to be employed for VoIP applications [13]. 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.

In [3] we have found that Skype implements congestion control functionalities which are able to match the available network bandwidth to some extent, while exhibiting persistent losses.

At our best knowledge this is the first attempt to derive a mathematical model of the congestion control algorithm used in VoIP applications such as Skype.

III. 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. We have routed all packets generated from Skype application to the ingress queues q_1 and q_2 of each host as it is shown in Figure 1. The measurement tool allows delays, available bandwidth and buffer size of each queue be set by the user. It is worth noticing that the described set-up implements an emulated environment similar to that of Dummynet [14], the only difference being that in our case we can use two hosts instead of three.

On each host we have installed Skype (S) 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 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.

Our testbed is able to log jitter, RTT, packet loss ratio as declared by the Skype application in the "Technical Call Infos" tooltip.

Finally, it is worth noticing that Skype flows are generated using always the same audio sequence by hijacking audio I/O[1]. From now on, the RTT of the connection is set at $100\,\text{ms}$ and the queue size is set equal to the bandwidth delay product unless otherwise specified.

IV. Experiments to Investigate the Skype congestion control

As we have already mentioned in the introduction, modeling the adaptation algorithm employed by Skype to match network available bandwidth is made complex by the fact that the source code of the application is not available and the application uses a proprietary and undisclosed communication protocol which is hidden by means of AES encryption.

In this investigation we consider the Skype sending rate as the output of the controller dynamics we are trying to model. At first we will determine what are the inputs and then how do they affect the output. We will consider as possible inputs to the controller the following variables: i) the end-to-end round trip time (RTT) experienced by the connection; ii) the packet loss rate. It is worth to notice that, since those two variables can be measured end-to-end, they are often employed in congestion control algorithms as feedback signals to detect network congestion.

For this reason congestion control algorithms are often classified as follows: i) *delay-based* algorithms, which infer congestion by monitoring either the one way delay, the end-to-end round trip time (RTT) or the queueing delay (examples of congestion control also belonging to this category are Fast and Vegas TCP [15],[16]); ii) *loss-based* algorithms, which infer congestion based on packet loss events such as in the case of TCP NewReno [17], which is the congestion control standardized by IETF and its variants; iii) *mixed loss and delay based* algorithms, which use both delay and packet losses as feedbacks such as in the case of recently proposed TCP Compound [18].

Figure 2 shows a schematic of the system: the receiver monitors the feedback variables like round trip time $\hat{RTT}(t)$, loss ratio $\hat{l}(t)$, (as it is shown in the technical info tooltip of Skype) and periodically sends these information back to the sender by piggybacking them on data packets. The sender receives feedback data and adjusts the sending rate $r_s(t)$ accordingly by throttling both packet size and packet sending rate.

[1]Skype DSP hijacker: http://195.38.3.142:6502/skype/

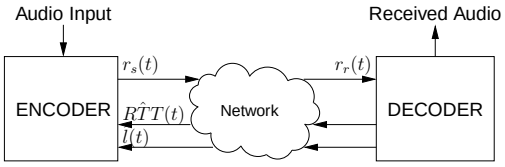


Fig. 2. Schematic of a Skype audio call over the Internet

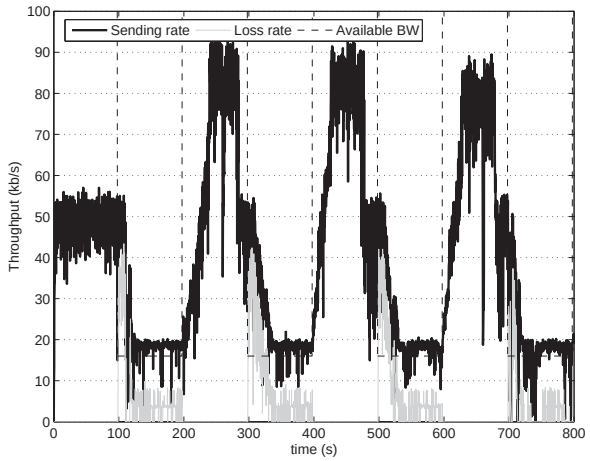


Fig. 3. Sending rate and loss rate in the presence of square wave available bandwidth of 200 s period

In order to derive how the feedback signal affects the Skype sending rate we consider step-like inputs which is a basic practice when trying to investigate the dynamic behaviour of a system. In particular we will start by investigating Skype flows when in the presence of square-wave available bandwidths to figure out the effect of network available bandwidth on the Skype sending rate $r_s(t)$. Then we will study how variable link delay $RTT(t)$ and packet loss ratio $l(t)$ influence the sending rate $r_s(t)$. Finally, we will conjecture and verify a simple model for the Skype sending rate controller.

Before starting to report our results, it is worth noticing that Skype version we have used employs the adaptive codecs iSAC[2] and iLBC[3] both developed by Global IP Sound.

A. Skype dynamics over a square form wave available bandwidth

This scenario aims at showing how the Skype sending rate reacts to sudden changes of available bandwidth. We employ an available bandwidth that varies as a square wave with maximum value $A_M = 160$ kb/s and minimum value $A_m = 16$ kb/s (see Figure 3).

We have run the 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.

Figure 3 shows that Skype decreases the sending rate when the link capacity drops from the value A_M to the value A_m. The Skype flow takes approximately 40 s to track the

[2]http://www.gipscorp.com/files/english/datasheets/iSAC.pdf
[3]http://www.gipscorp.com/files/english/datasheets/iLBC.pdf

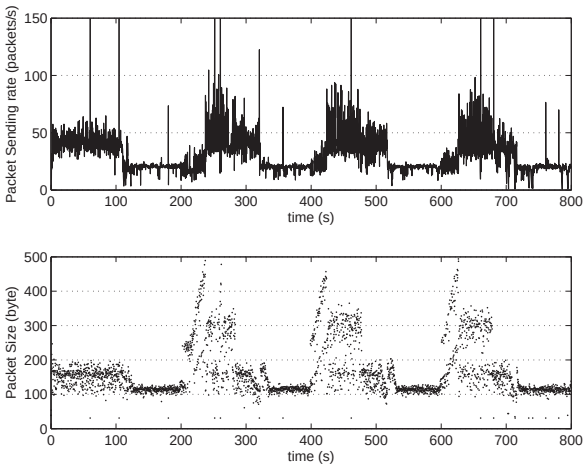


Fig. 4. Packet sending rate and packet size in the presence of a square wave available bandwidth of 200 s period

available bandwidth during which it experiences a significant loss rate. It can be viewed that, when the available bandwidth drops, the loss rate increases to a peak value of 35 kb/s whereas the sending rate reduces to less than 20 kb/s in around 40 s. By observing this behaviour it is very reasonable to conjecture that Skype implements a form of congestion control algorithm that reduces the sending rate when a high packet loss rate is measured. We will return soon on this feature by investigating how Skype behaves in the presence of packet loss rates.

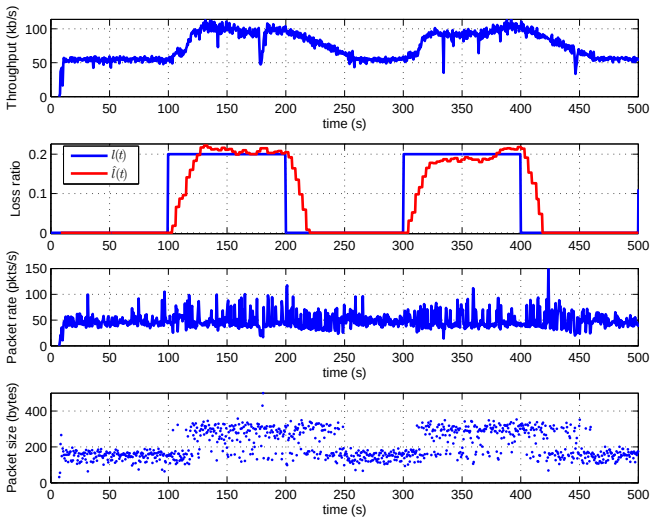
When the link capacity increases (i.e. at $t = 400$ s) the input rate ramps up to 90 kb/s again in around 40 s.

In order to find out the Skype controller dynamics, we have captured both packet sizes and packet sending rate over time (see Figure 4). By comparing the two figures it can be noticed that the packet sending rate drops in a step-like function when Skype detects congestion, whereas the packet size is decreased slowly, resulting in the slow adaptation to the available bandwidth we have dealt above. It seems that most part of Skype congestion control is performed by throttling the packet sending rate, whereas the packet size is varied for fine tuning.

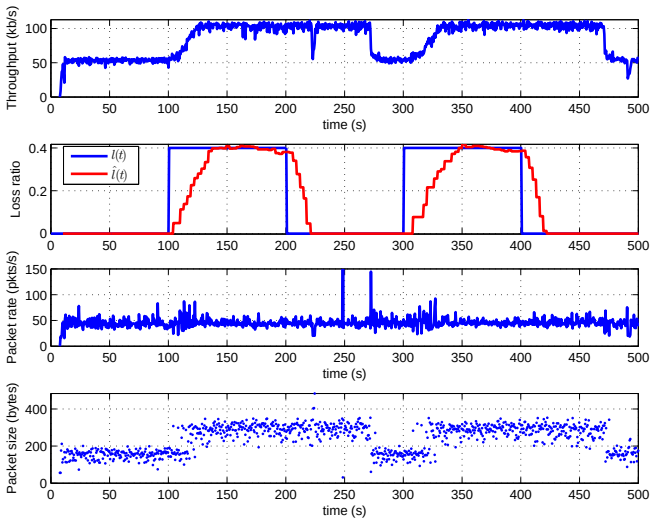
B. Skype over lossy link: throttling the loss ratio $l(t)$

In the previous section we have shown that Skype reacts to bandwidths shrinking by decreasing the packet sending rate and the packet size to adapt to the available bandwidth. However, the experiment we have just described does not reveal the inputs of the congestion controller either the congestion control category to which Skype belongs, i.e. loss based, delay based or both. For pursuing these goals, we now investigate the influence of packet losses on the Skype sending rate.

In order perform the investigation, the emulator injects packet losses which vary as a square wave having a maximum value of $l_M \in [0, 1]$ and a minimum value of 0. The RTT is set to $100\,ms$. We have logged the loss ratios as it is reported by the Skype application "Technical Call



(a)



(b)

Fig. 5. Sending rate, loss ratio, packet rate and packet size in the case of a square wave packet loss ratio with (a) $l_M = 0.2$ and (b) $l_M = 0.4$

Info" and we have reported them in the next figures. We have considered a packet loss ratio which varies as a square-wave with period of $200\,s$ and maximum values in the set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. Due to space constraints, we only report the most significant figures obtained for $l_M = 0.2$ and $l_M = 0.4$.

Figures 5(a) and (b) show two different behaviours of Skype in the presence of persistent losses on the link. In both cases Skype reacts to a persistent loss by increasing the transmission rate, i.e. the throughput. By looking at the packet size over time, in both cases we can see that when a persistent loss is detected the packet size is increased. This fact suggests that Skype employs a Forward Error Correction (FEC) scheme in order to cope with persistent losses.

Figure 5 reveals other interesting facts: i) the Skype

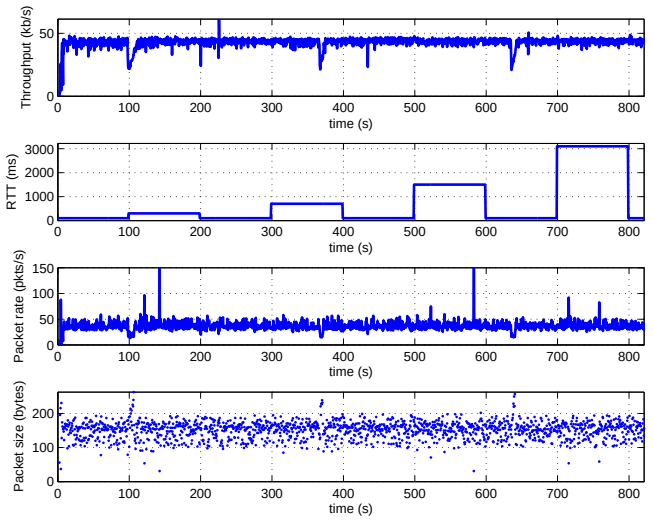


Fig. 6. Sending rate, loss ratio, packet rate and packet size in the case of a variable link delay

estimate of the loss ratio $\hat{l}(t)$ looks like a filtered version of the actual loss ratio $l(t)$ which has been set by using our emulation tool; ii) more interestingly, it seems that the signal $\hat{l}(t)$ drives the sending rate $r_s(t)$ in a roughly proportional way following as it can be argued by looking at $r_s(t)$ and $\hat{l}(t)$ dynamics; iii) in the case of $l_M = 0.4$ (Figure 5 (b)) the sending rate is kept at its maximum value of around 100 kb/s even when $\hat{l}(t)$ reaches 0. We argue that in this case Skype conservatively keeps on the FEC action for roughly $40\,s$ because it measures a high value of losses. Other experiments have shown that such a behaviour is followed when $l_M \geq 0.3$, so that we can argue that Skype uses a threshold of l_M between 0.2 and 0.3 to switch from one behaviour to the other.

Finally, it can be said that, due to the fact that the sending rate is not decreased when packet losses are detected, Skype does not employ a loss-based congestion control scheme.

C. Skype over variable delay link: throttling the round trip time $RTT(t)$

In this section we investigate the effect of the end-to-end delay on the Skype sending rate. To the purpose we vary the link delay as shown in Figure 6 and we set the packet loss ratio to 0. The available bandwidth is made large enough in order not to generate congestion on the link. The figure clearly shows that even very large variations in the RTT (the last variation is $3.0\,s$) do not produce any effect on the Skype sending rate. Thus we can also exclude that the application implements a delay based congestion control algorithm.

We have performed experiments in which the loss ratio $l(t)$ and $RTT(t)$ vary as in-phase square waves with the same period ($T = 200$), but we do not report the figures due to space constraints. Interestingly, we have found that the FEC action is inhibited when the RTT increases to its maximum value. Therefore, it can be concluded that the RTT influences the behaviour of the FEC action when in the presence of lossy links.

V. The Skype Congestion Control Model

A. Modelling the Skype congestion control

In Section IV-A we have already shown the behaviour of Skype in the presence of a variable link capacity and we have found that Skype is able to match the available bandwidth to some extent after a significant transient time. In this subsection we provide a mathematical model to describe the dynamic behaviour of Skype.

We make the hypothesis (confirmed by the large number of experiments we have run) that the audio codec employed by Skype is multi-rate so that the encoder can select among N levels $L_k = \{L_1, L_2, \ldots, L_N\}$ with $L_1 < L_2 < \ldots < L_N$. Moreover, we assume that the Skype adaptive codec is able to select the most appropriate mode according to some metric which should be the analogous of Carrier to Interference ratio (C/I) in the case of Adaptive Multi-Rate Wide Band (AMR-WB) encoder [19]. Let $i(t)$ denote the switching law and let $L_{i(t)}$ be the encoder level at time t. It is very reasonable to assume that the switching law $i(t)$ is implemented by using if-then-else clauses, thus being very difficult to be identified[4].

We make the hypothesis that the Skype control law of the sending rate in the case of congestion is ruled by the following equation:

$$r_s(t) = (1 - \hat{l}(t)) \cdot (1 + f(t)) L_{i(t)} \qquad (1)$$

where $f(t) \in [0, 1]$ models the FEC action, meaning that when $f(t) = 0$ the FEC action is off and when $f(t) = 1$ the FEC action is at maximum. Basically, we conjecture that the switching function $i(t)$ selects the layer $L_{i(t)}$ based on the network conditions (RTT, jitter, loss ratio) and the rate is shrinked as much as the filtered loss ratio $\hat{l}(t)$ suggests. It is worth noting that (1) is able to explain the normal behaviour of Skype when no congestion or losses occur i.e. when $f(t) = 0$ and $\hat{l}(t) = 0$ when the sending rate results $r_s(t) = L_{i(t)}$ (see Figure 6).

We have run a set of experiments using a square wave available bandwidth with maximum value of 160 kb/s and minimum value of 16 kb/s with a period of 400 s. We have shown the obtained results in Figure 7. In order to verify Eq. (1), we have conjectured the FEC action $f(t)$ shown in Figure 7 and we have supposed that $L_{i(t)}$ is set to 54 kb/s during the experiment. Figure 7 shows both the Skype measured sending rate and predicted sending rate using Eq. (1). It can be seen that the model nicely predicts the Skype sending rate.

Now let us focus on the conjectured behaviour of $f(t)$ shown in Figure 7: the FEC is kept off until the first bandwidth increase happens at $t = 400$ s. We conjecture that Skype infers a bandwidth increase when the RTT of the connection suddenly decreases and it triggers a "probing phase". Moreover, we think that Skype increases the FEC value when probing for bandwidth since losses can occur during this phase. Figure 7 shows that the FEC action is reduced to 0.2 when $\hat{l}(t)$ reaches 0, and is then turned off later when $\hat{l}(t)$ is detected not to grow.

[4]The boolean expression evaluated in the if clause can be complex.

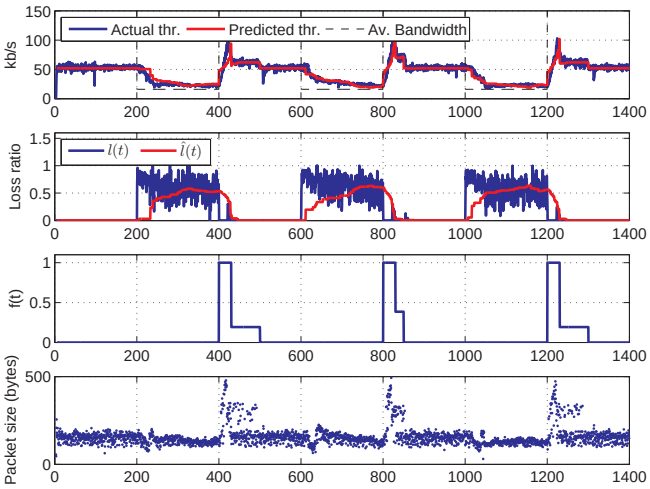


Fig. 7. Comparison between the actual and predicted rate using Eq. (1)

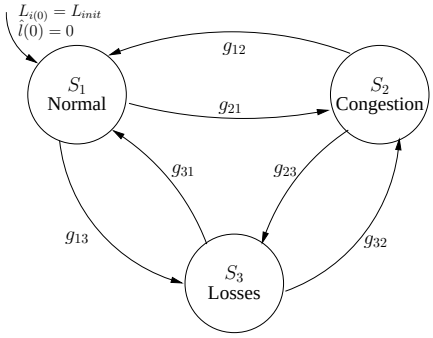


Fig. 8. Skype VoIP hybrid system model

It is worth to notice that the Equation (1) allows us to explain the long transient time that we have reported in Section IV-A. The main driver of the Skype sending rate is in fact $\hat{l}(t)$, which is a low-pass filtered version of the actual loss ratio $l(t)$. The long transient exhibited by $\hat{l}(t)$ (see Figure 5) affects the sending rate response to congestion which is somewhat slow.

B. The Skype Hybrid Automaton

Building upon the results obtained by the investigations presented in Section IV, we model the Skype VoIP sending rate using the *hybrid automaton* shown in Figure 8. In particular, the sending rate can exhibit three different dynamics depending on the state S_i $(i = 1, 2, 3)$ of the automaton: in the state S_1, which is characterized by normal network conditions, i.e. no congestion occurs and no losses are present, the sending rate is kept unchanged; in the state S_2, which is triggered when Skype realizes that the network capacity has changed and congestion occurs, the sending rate is throttled using (1); in the state S_3, which is triggered when losses are present but are not due to network congestion (i.e. due to a lossy link, see Section IV-B) the FEC action $f(t)$ is used in order to counteract the losses which Skype attributes to a lossy link.

C. Hybrid modelling of a Skype Flow Accessing a Bottleneck

In this Section we propose a hybrid automaton to model a Skype flow accessing a single bottleneck link characterized by an available bandwidth $b(t)$, a drop tail queue whose maximum size is q_M and a round trip time T which is the sum of the delay of the forward path T_1 and the delay of the backward path T_2 . In the following we will denote with x_T the signal x delayed T seconds and we will omit the time dependence of the signals for brevity. The evolution of the queue can be modelled by the following differential equation [9]:

$$\dot{q} = \begin{cases} 0 & q = 0, r \leq b \text{ or } q = q_M, r \geq b \\ r - b & \text{otherwise} \end{cases} \quad (2)$$

where r is the queue input rate. The queue overflow rate o can be modelled as follows:

$$o = \begin{cases} r - b & q = q_M, r > b \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

which means that when the queue is full the exceeding input rate $r - b$ is dropped [20].

Let us now consider the model of the sending rate of a Skype VoIP flow: equation (1) means that the Skype sending rate r_s is mainly driven by the signal $\hat{l}$ which is a filtered version of the actual packet loss ratio $l = o/r$ measured at the sender after the delay T_2. Based upon experiments we assume that Skype filters l_{T_2} using a first order low pass filter with a time constant τ:

$$\dot{\hat{l}} = -\frac{1}{\tau}\hat{l} + \frac{1}{\tau} l_{T_2}$$

which by considering that $l = o/r$ turns out:

$$\dot{\hat{l}} = -\frac{1}{\tau}\hat{l} + \frac{1}{\tau}\frac{o_{T_2}}{r_{T_2}} \quad (4)$$

By substituting (1) and (3) in (4), after straightforward computations we obtain:

$$\dot{\hat{l}} = \begin{cases} f_1 = \frac{1}{\tau} - \frac{\hat{l}}{\tau} - \frac{b_{T_2}}{\tau(1-\hat{l}_{T_2})(1+f_{T_2})L_{T_2}} & q = q_M,\ r > b \\ f_2 = -\frac{1}{\tau}\hat{l} & \text{otherwise} \end{cases} \quad (5)$$

Let $x = [\hat{l} \quad q]^T$ denote the state of the system. It is simple to show that the state dynamics of the considered system can be described by means of the three state hybrid automaton $\mathcal{H}$ which is shown in Figure 9. In particular, the state Σ_1 holds when the queue is empty and the input rate is below the link available bandwidth, the dynamics of state is described by Σ_2 when the queue is neither full nor empty; the state Σ_3 describes the evolution of the system when the queue is full and the input rate is larger than the available bandwidth.

Lemma 1: The system Σ_3 has a unique equilibrium point:

$$\hat{l}^* = 1 - \sqrt{\frac{b^*}{L^*(1+f^*)}}\,;\, q^* = q_M \quad (6)$$

when $b^* < (1+f^*)L^*$. Considered the system with no delay the equilibrium is asymptotically stable for perturbations $\Delta l \in [0, 1]$.

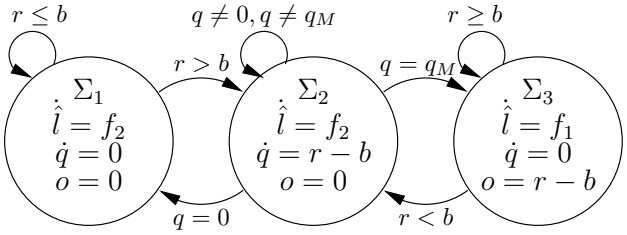


Fig. 9. The hybrid automaton model $\mathcal{H}$ of a Skype flow accessing a drop tail queue

Proof: Let us focus on the system Σ_3 which holds when the queue is full, i.e. when congestion occurs. By imposing $\dot{\hat{l}} = 0$ we obtain the equilibrium $(\hat{l}^*, q^*)$ which corresponds to the autonomous system obtained when the steady state inputs are b^*, L^* and f^*:

$$\frac{1}{\tau} - \frac{1}{\tau}\hat{l}^* - \frac{1}{\tau}\frac{b^*}{L^*}\frac{1}{1-\hat{l}^*}\frac{1}{1+f^*} = 0 \Rightarrow$$

$$\hat{l}^* = 1 - \sqrt{\frac{b^*}{L^*(1+f^*)}}$$

It is worth to notice that $\hat{l}^* \in [0, 1]$ since the inequality $b^* < L^*(1 + f^*))$ holds.

In order to prove the stability of the equilibrium we employ the direct Lyapunov method. By using the Lyapunov function $V(\hat{l}) = \frac{1}{2}(\hat{l} - \hat{l}^*)^2$ we obtain after straightforward computations:

$$\dot{V}(\hat{l}) = -\frac{1}{\tau}\frac{(\hat{l} - \hat{l}^*)^2(2 - \hat{l} - \hat{l}^*)}{1 - \hat{l}}$$

which is definite negative since $\hat{l} \in [0, 1]$. ■

Lemma 2: The system Σ_1 has a unique equilibrium point:

$$\hat{l}^* = 0\,;\, q^* = 0 \quad (7)$$

when $b^* > (1 + f^*)L^*$ which is globally asymptotically stable.

Proof: The lemma is proved by observing that Σ_1 is a linear system with one eigenvalue that is always strictly negative. ■

Lemma 3: The hybrid automaton $\mathcal{H}$ has a sink state Σ_3 if $b^* < (1 + f^*)L^*$ and a sink state Σ_1 if $b^* > (1 + f^*)L^*$.

Proof: Let us consider the first part of the proposition which assumes $b^* < (1+f^*)L^*$. The proof starts by showing that for any initial condition $(\hat{l}_0, q_0)$, the state dynamics of the hybrid automaton enters the state Σ_3 and remains indefinitely in this state, i.e. Σ_3 is a *sink* state. To this purpose we find the reachability set of $\mathcal{H}$.

Let us fook at Figure 10 (a) that shows the phase plane $X = \{\hat{l}, q : 0 \leq \hat{l} \leq 1, 0 \leq q \leq q_M\}$ partitioned in two zones Z_1 and Z_2 depending on the sign of $\dot{q}$. It results that if $r > b$ then the queue builds up (zone $Z_1 = \{(\hat{l}, q) \in X : \hat{l} < 1 - b^*/(1 + f^*)L^*, q \neq q_M\}$) otherwise the queue is drained (zone $Z_2 = \{(\hat{l}, q) \in X : \hat{l} > 1 - b^*/(1 + f^*)L^*, q \neq 0\}$).

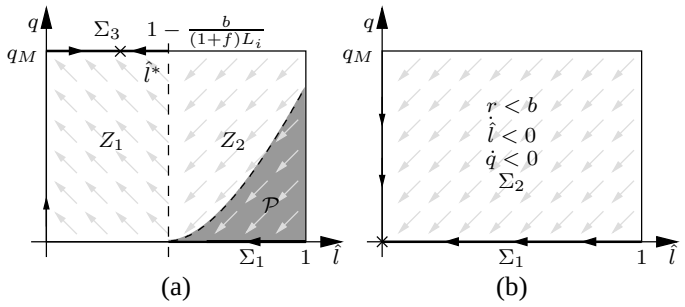


Fig. 10. Qualitative phase portrait of the Skype state space model when (a) $b < (1+f)L_i$ or (b) $b > (1+f)L_i$

It is easy to show that if we initialize $\mathcal{H}$ in the state Σ_1 (the queue is empty and $r < b$) the state evolution is given by:

$$\hat{l} = \hat{l}_0 e^{-\frac{t}{\tau}} \quad (8)$$

$$q = 0 \quad (9)$$

where the initial condition is $(\hat{l}, 0)$. The state of $\mathcal{H}$ remains in Σ_1 provided that $r < b$ i.e. when $\hat{l} > 1 - b^*/(1+f^*)L^*$. The state switches to Σ_2 at the time:

$$t_{1\to 2} = -\tau \log\left(1 - \frac{b^*}{(1+f^*)L^*}\right)$$

It is worth to notice that the switching time $t_{1\to 2}$ is always positive, so that we can conclude that if we initialize $\mathcal{H}$ in the Σ_1 state, the only reachable state is Σ_2. Moreover, since the state is now in the region Z_1 the queue must build up, so that the only reachable state from Σ_2 is now Σ_3. Thus, we can conclude that if $\mathcal{H}$ is initialized in Σ_1 the only possible evolution of the state is $\Sigma_1 \to \Sigma_2 \to \Sigma_3$.

Let us now focus on the states that can be reached starting from Σ_2. We will show that if we initialize $\mathcal{H}$ with Σ_2, that is $(\hat{l}_0, q) \in Z_1 \cup Z_2$ the state of the hybrid automaton $\mathcal{H}$ can go either to Σ_1 or to Σ_3. The evolution of the state is given by:

$$\hat{l} = \hat{l}_0 e^{-\frac{t}{\tau}} \quad (10)$$

$$q = q_0 - k\tau\hat{l}_0(1 - e^{-\frac{t}{\tau}}) + t(k - b^*) \quad (11)$$

where we have defined $k = L^*(1+f^*)$ for sake of brevity. It is very easy to show that if the initial condition belongs to Z_1 for sure there is no way for the state to finish in Σ_1 (the queue must build up) so that the only state that can be reached is Σ_3 (full queue). Let us now start from the initial condition in Z_2: now two different evolutions of the hybrid automaton are possible: the state can be either go in Σ_1 then back to Σ_2 and then to Σ_3 or either go directly in Σ_3. It is simple but lengthy to show that the state follows the path $\Sigma_2 \to \Sigma_1 \to \Sigma_2 \to \Sigma_3$ if and only if the initial condition belongs to the set:

$$\mathcal{P} = \{(\hat{l}, q) \in Z_2 : q < -\tau(k - b^*)(1 + \log\hat{l}) + k\tau\hat{l} + \tau(k - b^*)\log(1 - \frac{b^*}{k})\}$$

otherwise the state of the hybrid automaton will follow the path $\Sigma_2 \to \Sigma_3$. In either the cases if the hybrid automaton starts from Σ_2 the state will end in Σ_3 that is, the queue fills up and packet losses occur.

In order to conclude the first part of the proof we need to show that if we initialize the system in Σ_3 the state of $\mathcal{H}$ will remain in Σ_3. In Lemma 1 we have shown that Σ_3 has one equilibrium point that is asymptotically stable so that the trajectories are attracted by the equilibrium. We can conclude that if we initialize the automaton in the Σ_3 state the evolution of the system will always be in Σ_3 and will not be able to switch to another state.

The second part of the proof which states that Σ_1 is a sink state if $b^* > (1+f^*)L^*$ follows the same arguments we have developed in the first part and it is omitted due to space limitation. ■

Proposition 1: By considering the equilibrium inputs b^*, L^* and f^* the hybrid automaton shown in Figure 9 has the following equilibrium state:

$$\hat{l}^* = 1 - \sqrt{\frac{b^*}{L^*(1+f^*)}}\,;\; q^* = q_M \quad (12)$$

if $b^* < (1+f^*)L^*$ and:

$$\hat{l}^* = 0\,;\; q^* = 0 \quad (13)$$

otherwise. Considered the system delay free system ($T_2 = 0$) both the equilibria are asymptotically stable.

Proof: From Lemma 3 we know that if $b^* < (1+f^*)L^*$ then Σ_3 is a sink, so that for any initial condition $(\hat{l}_0, q_0) \in X$ the state must end in Σ_3. Moreover, in Lemma 1 we proved that Σ_3 is an asymptotic stable equilibrium (12), so that we can conclude that for any $(\hat{l}_0, q_0) \in X$ the state will asymptotically converge to (12).

If we assume $b^* > (1+f^*)L^*$ following similar arguments we can conclude that for any any $(\hat{l}_0, q_0) \in X$ the state will asymptotically converge to (13). ■

Proposition 2: The controller employed by Skype is not able to counteract congestion episodes unless $L^*(1+f^*) < b^*$.

Proof: Since we are under the hypotesis of Proposition 1, we know that (12) is an asymptotically stable equilibrium for $\mathcal{H}$. Therfore by considering (3) the steady state value of the overflow rate is given by:

$$o^* = (1 - \hat{l}^*)L^*(1+f^*) - b^*$$

Now by substituting (12) in the equation written above, we obtain:

$$o^* = \sqrt{b^* L^*(1+f^*)} - b^*$$

that is greater than zero if $b^* < L^*(1+f^*)$. In other terms, under congestion, the evolution of the system will be described by Σ_3 (Lemma 3) and thus the queue will be full ($q^* = q_M$) and the overflow rate will be persistent ($o^* > 0$). ■

Remark 1: The steady state value of the loss rate o^* does not depend on the time constant τ of the filter.

VI. Conclusions and further work

This paper proposes a dynamic switching model of the congestion control algorithm implemented by the Skype VoIP application. At our best knowledge, this is the first attempt aiming at developing such a model because the Skype protocol behaviour is hidden by AES encryption.

By setting up an experimental testbed we have conjectured and verified what are the inputs and how they affect the Skype sending behaviour. Main findings are: (1) Skype does not implement a delay based control; (2) the sending rate matches the available bandwidth with a finite error; (3) the loss ratio is the main driver of the input rate; (4) the encoder selects the encoding rate over a finite set of levels by taking into account a metric that depends on the loss rate.

As a further work we plan to relax the assumption $T_2 = 0$ in the proof of Proposition 1.

References

[1] P. Biondi and F. Desclaux, "Silver Needle in the Skype," *BlackHat Europe '06*, Mar. 2006.

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

[3] L. De Cicco, S. Mascolo, and V. Palmisano, "An Experimental Investigation of the Congestion Control Used by Skype VoIP," *WWIC '07*, vol. 4517, pp. 153–164, May 2007.

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

[5] R. Srikant, *The Mathematics of Internet Congestion Control*. Birkhäuser, 2004.

[6] E. Altman, T. Basar, and R. Srikant, "Congestion control as a stochastic control problem with action delays," *Automatica*, vol. 35, no. 12, pp. 1937–1950, 1999.

[7] S. Low, F. Paganini, and J. Doyle, "Internet congestion control," *Control Systems Magazine, IEEE*, vol. 22, no. 1, pp. 28–43, 2002.

[8] J. Lee, S. Bohacek, a. P. H. Jo and K. Obraczka, "Modeling communication networks with hybrid systems," *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 630–643, 2007.

[9] S. Mascolo, "Congestion control in high-speed communication networks using the Smith principle," *Special Issue on "Control methods for communication networks" Automatica*, vol. 35, no. 12, pp. 1921–1935, 1999.

[10] C. Hollot, V. Misra, and D. Towsley, "A control theoretic analysis of RED," *IEEE INFOCOM '01*, vol. 3, Apr. 2001.

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

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

[13] S. Floyd and E. Kohler, "TCP Friendly Rate Control (TFRC): the Small-Packet (SP) Variant," *RFC 4828, Experimental*, Apr. 2007.

[14] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 1, pp. 31–41, 1997.

[15] D. Wei, C. Jin, S. Low, and S. Hegde, "FAST TCP: motivation, architecture, algorithms, performance," *IEEE/ACM Trans. on Networking*, vol. 14, no. 6, pp. 1246–1259, 2006.

[16] L. Brakmo and L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 8, pp. 1465–1480, 1995.

[17] S. Floyd, T. Henderson, and A. Gurtov, "NewReno modification to TCP's fast recovery," *RFC 3782, Standard*, Apr. 2004.

[18] K. Song, Q. Zhang, and Sridharan, "Compound TCP: A Scalable and TCP-Friendly Congestion Control for High-speed Networks," *PFLDNet '06*, Feb. 2006.

[19] B. Bessette et al, "The adaptive multirate wideband speech codec (AMR-WB)," *Speech and Audio Processing, IEEE Transactions on*, vol. 10, no. 8, pp. 620–636, 2002.

[20] S. Mascolo, "Modeling the Internet congestion control using a Smith controller with input shaping," *Control Engineering Practice*, vol. 14, no. 4, pp. 425–435, Apr. 2006.