A Detailed Measurement of Skype Network Traffic

Dario Rossi
ENST ParisTech – INFRES
email: dario.rossi@enst.fr

Marco Mellia, Michela Meo
Politecnico di Torino – DELEN
email: name.surname@polito.it

***Abstract*—The last few years witnessed peer-2-peer (P2P) and VoIP telephony gaining a tremendous popularity: Skype is beyond doubt the most amazing example of this new phenomenon, as its 170 millions users testify. In this paper, we propose a detailed measurement of Skype sources: we adopt a twofold methodology, using active experiments and passive measurement to gain knowledge about the traffic Skype generates, in a *controlled* and *realistic* environment, respectively.**

The analysis considers both signaling traffic as well as voice calls. Furthermore, we address the description of Skype sources at both packet- and flow-levels: we describe fine-grained dynamics of the interaction between Skype and the network, as well as provide a macroscopic characterization of Skype traffic.

I. Introduction

In the last years, the Peer-2-Peer (P2P) paradigm gained popularity to the point that p2p application are responsible for vast portion of Internet traffic, which not only involves residential broadband Internet accesses, but also enterprise networks. Indeed, many different p2p applications exist, such as chat, file-sharing, telephony, video-streaming, backup systems, etc. Furthermore, even within the same target application, there are many different application protocols and flavors available–such as structured vs unstructured overlays– that translate into very different types of traffic from the network viewpoint.

The ability to measure, understand and model the network traffic has been a fundamental activity since the time when Web was *the* Internet application [1]: indeed, this knowledge is instrumental for a wide range of network operations such as performance evaluation, traffic engineering, capacity planning. As a consequence of p2p widespread adoption, the research activities related to p2p traffic identification, classification and measurement acquired importance. File-sharing, being the first example of new application exploiting the p2p paradigm has been studied for a relatively long time [2]: as a result, many details concerning the file popularity [3], user churn [4], and query process [5] are available. At the same time, we point out that while the first applications such as KaZaa were proprietary and unknown, many currently popular applications (such as BitTorrent, Gnutella, aMule) are implemented as open-source software and their protocol internals are well known and documented. Conversely, more recent p2p applications (such as Internet telephony, videoconferencing, video streaming, etc.), have enjoyed an enormous success among the Internet population but remains relatively unknown.

The best example is represented by Skype – which is beyond any doubt *the* VoIP application in the current Internet application spectrum: developed in 2002 by the creators of KaZaa, it recently reached over 170 millions of users, and accounts for more than 4.4% of total VoIP traffic [6]. As such, Skype is attracting the attention of the research community and of the telecom operator as well, and a great deal of valuable work can already be found in the literature [7], [8], [9], [10], [11], [12], [13]. Yet, due to Skype protocols proprietary and obfuscated nature, due to traffic encription and due to the use of reverse-engineering techniques [7], many interesting questions remain, to date, unanswered. Besides, Skype implements a number of techniques to circumvent NAT and firewall limitations [8], which add further complexity to an already blurred picture.

In this paper, we develop a complete characterization of the traffic Skype generates, considering both signaling traffic and voice/video calls, providing several insights at both packet- and flow-levels. The analysis relies on two complementary approaches. On the one hand, we examine Skype activities in a controlled environment, using an active testbed methodology. On the other hand, building over our previous work [10] that allows us to precisely identify Skype clients, we resort to passive measurements in order to reveal those aspects that a purely active methodology alone cannot capture.

II. Skype Premier and Source Model

This section provides a high-level description of the Skype model we adopt in this paper, as well as useful information about Skype that can be gathered by works dealing with its internals [7], [8], with the issue of Skype identification [9], [10], and with the characterization of

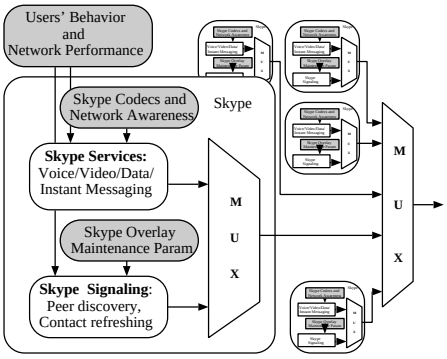

Fig. 1. Synopsis of the Skype source model

its traffic and users [11], [12], [13]. The main difference between Skype and other VoIP clients is that Skype adopts a P2P architecture. Only user authentication is performed under a classical client-server architecture, via a public key mechanisms: except for the authentication, all further signaling is performed on the P2P overlay. Thus, Skype user's informations (e.g. contact list, status, preferences, etc.) are entirely decentralized and distributed among P2P nodes – which allows scalability on the one hand and cost-effectiveness on the other hand.

Skype offers end users several (free) services: i) voice communication, ii) video communication, iii) file transfer and iv) chat services. The communication between users is established using a traditional end-to-end IP paradigm, but Skype can also route calls through other p2p nodes (called supernodes) to ease the traversal of symmetric NATs and firewalls. Voice calls can also be directed toward the PSTN using Skypein/Skypeout service, in which case a fee is applied. In the following, we denote by *End-to-End (E2E)* call any voice/video communication occurring between two Skype clients, and by *End-to-Out (E2O)* any call involving a Skype peer and a PSTN terminal.

A single random port is selected during application installation, and it is never changed (unless forced by the user), thus we can identify:

- a Skype *client*, by its socket, i.e., the (host IP address, Skype UDP/TCP port) pair.
- a Skype *flow*, by using the traditional tuple (IP source and destination addresses, UDP/TCP source and destination ports, 1IP protocol type), in which at least one endpoint is known to be a Skype client.

A flow starts when a packet with a new flow tuple is first observed, while it is ended by either an *inactivity timeout*[1] or, in case of TCP, by observing the tear-down sequence if present. Skype clients are identified as in [10], so that it is possible to identify all flows that are originated by or directed to a Skype client.

For what concerns the transport layer, Skype relies on both TCP and UDP. Both signaling and communication are preferentially carried over UDP – although when UDP communication is impossible, Skype falls back to TCP, listening to the same random port whenever possible (or eventually port 80 or 443, which are normally left open by network administrators to allow Web browsing). However, irrespectively of the transport layer protocol, the Skype traffic can be divided, as Fig. 1 depicts, in two components: namely *signaling* and *services*, which can be influenced by different external factors (such as user's behavior and network status) as well as internal Skype settings (such as Codec preferences).

Signaling traffic, addressed in Sec. III, can be modeled as a superposition of different "threads," corresponding to different activities – such as peer discovery, overlay maintenance and contact refreshment. In the following, we will show that different threads are differently affected by external factors: for instance, we anticipate that peer discovery is *latency* driven, whereas the contacts-refreshing activity appears to be *social-network* driven. *Service* traffic is addressed in Sec. IV, where we consider voice calls but explicitly neglect video calls, data transfer and instant messaging. We investigate Skype dynamics at a packet-level granularity through a testbed study, unveiling if and how Skype reacts to the inferred *network performance* – such as packet loss and available bandwidth.

III. Skype Signaling Traffic

In this section, we analyze Skype signaling traffic by means of passive measurements, applying the classification framework presented in [10]. Results refer to one week long trace collected during may 2007 at the access link of Politecnico's campus LAN, where about 7000 different hosts are used by both students and staff members. In the following we focus on signaling flows carried over UDP only, which are the largest and most interesting part of Skype signaling traffic.

A. Signaling activity

Since all signaling traffic is encrypted, we are forced to adopt a black-box approach. First of all, it is necessary to

[1]Since the largest inter-packet gap we ever observed is 180 s (likely used by Skype as keep-alive message to force the refresh of possible NAT entries), in this paper we set the inactivity timer to 200 s.

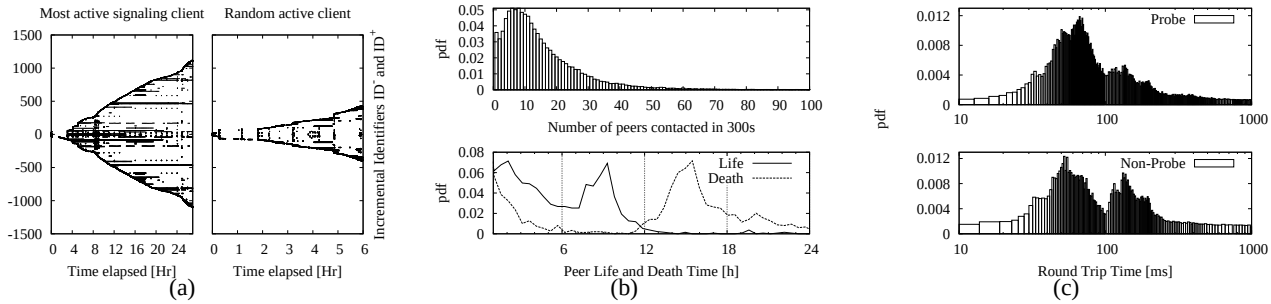

Fig. 2. Passive network monitoring: (a) Pictorial representation of Skype activity pattern for two different peers. (b) PDF of the number of peers contacted 300 s (top), and inferred peers' life-time and death-time (bottom); (c) PDF of the round trip time of probe and dialog traffic.

point out that Skype traffic source signaling is very limited in bitrate: the 95-th percentile tops to about 100 bps, while very few nodes (possibly supernodes) generate, on average, more than 1 kbps. Yet, an intense signaling task is carried on during all the peer lifetime, as shown in Fig. 2-(a), which depicts the typical Skype activity pattern. We consider two specific peers, namely the most active peer that do not perform any call (left plot) and a randomly picked peer exhibiting both signaling and voice traffic (right plot). Let p be the observed peer. Each dot in the picture corresponds to a packet in the trace: the x-axis represents the packet arrival time (since the first packet observed for p). A positive value on the y-axis reports an identifier, ID, of a peer that received a message from p; similarly, negative values represent peers that sent messages to p.

Several remarks can be gathered from Fig. 2-(a): indeed, though the *semantic* of the signaling activity cannot be inferred from purely passive measurements, the *form* of signaling activity can be further differentiated. Interestingly, the number of contacted peers exhibits an almost linear growth with time, hinting to P2P network discovery and maintenance being carried on during most of the peer lifetime. Moreover, the range of the y-values corresponds to the number of different Skype peers whom the selected peer p is exchanging message with: the plot shows that the most active peer has contacted (was contacted by) about 1100 other peers during 28h, whereas the random peer by about 450 during 6h. Then, notice that signaling is mainly built by single message probes, to which (most of the times) some kind of acknowledgment follows. The fact that p knows the address and ports of valid (but previously uncontacted) Skype peers means that the above information is carried in the encrypted portion of previous signaling messages. Some peers are instead contacted on a regular basis: in the activity pattern plot, horizontal segments state that the same peer is periodically contacted during p lifetime. On the contrary, vertical patterns hint to the presence of timers that trigger an information refreshment, which involves both old peers, and probe discoveries toward new peers (this behavior is clearly visible in the right-hand side of Fig. 2-(a) every hour). The above observations suggest the existence of two very different kinds of the Skype signaling, namely:

- **Probe** traffic, which aims at network discovery: probe flows are made of a single packet sent toward a peer, to which a single reply packet possibly follows, but *no further message* is exchanged between the pair of peers.
- **Non-probe** traffic, which aims at the network maintenance, including overlay and contact information management: non-probe flows are either flows longer than one packet or sequence of single-packet probe flows, separated by a time gap larger than the inactivity timeout.

Finally, in order to gauge a relationship among the amount of signaling flows and the number of contacts in the buddy list, we performed a simple active experiment. We used three Skype accounts and measured 8 hours of their idle activity: one account had a single online contact, the second account had 5 online contacts, and the third had 10 online contacts. The experimental results suggest the existence of a *super-linear* dependency between the number of contacts and the number of signaling packets exchanged, although a larger scale experiment would be needed to more precisely quantify the relationship.

B. Signaling flows

As Fig. 2-(a) already showed, many signaling flows are single-packet probes that create new temporary soft-state entries, rarely used later on. At the same time, it is

possible to observe persistent signaling activity transferring a few MBytes of information over several thousands of packets: indeed, probes account for less than 5% of the exchanged signaling *bytes*. A possible reason behind this empirical evidence could be the presence of *super-nodes* among our internal clients, that generate intense and long-lasting signaling activity – though this statement requires further investigation.

Further insight about Skype signaling is given in Fig. 2-(b), which quantifies the peer contact rate, as well as the peer life-time and death-time. More precisely, let us focus on the number N of *different* peers contacted by peer p considering a time interval of 300s. N is closely related to the number of signaling flows that p generates in the time unit. Considering all internal peers and the whole analysis week, the distribution of N is shown in the top portion of Fig. 2-(b). In 90% of the cases, N is smaller than 30, with an average of 16; in 1% of the cases, N exceeds 75. Note that this metric is of particular interest since it is related to the burden a Skype client poses in any device operating at flow level, e.g., a entry in a NAT table, a look-up in a firewall ACL table, etc.

Bottom portion of Fig. 2-(b) reports the PDF of the peer life-time and death-time, inferred from the signaling activity during the week long observation period: in particular, a peer is considered dead if no packet is sent for a period of time longer than an idle time τ. Since we verified that values $\tau > 200\,\mathrm{s}$ have a minimal impact on the result, we conservatively set $\tau = 500\,\mathrm{s}$. Interestingly, except for very short sessions, most of the peers are alive for about one third of the day (i.e., during working hours) and remain dead for the rest of time (i.e., during the night). Thus, it seems as though peers' lifetime follows the Personal Computers' lifetime, which suggests that most of the people runs Skype by default. An important consequence is that Skype churning rate is very low, which explains the modest bitrate requirement for P2P overlay maintenance.

C. Signaling locality

We now investigate whether further, important, differences exist between probe and non-probe traffic. Indeed, it would be reasonable for Skype to implement some mechanism that, by discovering network hosts that are geographically close, privileges nearby hosts, so as to minimize network latency. Therefore, we focus on the locality properties of probe and non-probe peers, and depict in Fig. 2-(c) the PDF of the Round Trip Time (RTT) between any two peers. More precisely, in the case of probe traffic, we define RTT as the time elapsed between the packet probe going out from the campus LAN and the probe response packet. For non-probe traffic, we consider only the first sent-received packet pair as a measure of the RTT. Clearly, this measurement takes into account both the network and the application layer latencies.

The information in the picture confirms our previous intuition: the latency of probing traffic is lower than that of non-probing traffic. Given our measurement location, RTT smaller than 100ms are typical of nodes within the European Union, while RTT larger than 100ms are typical of nodes outside it. Measurement results allow us to conjecture that the probing mechanism is *latency driven*: Skype client probes for peers based on the information received by other peers so that low latency peers are more likely selected than high latency ones. Conversely, non-probe traffic RTT tends to be larger: considering that users resort to Skype to lower communication fees and to keep contacts with other faraway users, this is rather unsurprising. The contact selection mechanism is reflected into the peer selection mechanisms so that it seems to be *preference driven*, i.e., the Skype overlay is modeled over the user social network. On the contrary, the peer discovery mechanisms implemented by means of single-packet probes is driven by the physical properties of the underlying network.

IV. Skype Service Traffic

The characteristics of voice/video service traffic generated by the Skype source are driven by the voice encoder and the network performance, to which Skype adapts during a call in order to provide users with high QoS.

For what concerns voice/video Codecs, Skype chooses via an unknown algorithm among ISAC, ILBC, G.729, iPCM-WB, E-G.711A, PCM A/U, VP7 that are all standard except ISAC and VP7, proprietary solutions of GlobalIPSound [14] and On2 [15], respectively. ISAC is the preferred Codec for End-to-end calls, while the G.729 Codec is preferred for End-to-Out calls and TrueMotion VP7 is used for streaming webcam video. As a side node, due to the different characteristics of each Codec, a Skype voice call can consume up to 230 kbps and as few as 11 kbps. Also, variable bit rate Codecs (VBR), such as ISAC and iPCM-wb, exhibit larger message size variance with respect to constant bitrate (CBR) ones (e.g., G.729, iLBC and PCM).

However, irrespectively of the Codec, in order to cope with the potential loss of a voice block or to modify the message generation rate, the source may multiplex one or more blocks of encoded voice in a single

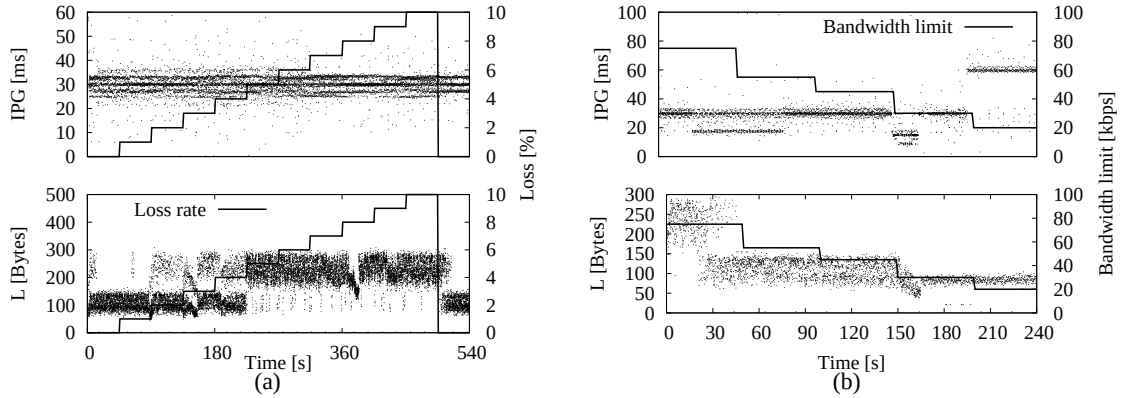

Fig. 3. Active testbed measurement: Skype inter-packet-gap and message size for (a) increasing packet loss rates and (b) decreasing bottleneck bandwidth.

message [10]. Therefore, to better observe the Skype message generation process, and to further investigate how Skype reacts to different network conditions, we setup a testbed involving several PCs connected by a Linux box. Restricting our attention to the default ISAC Codec, we perform a set of experiments by generating voice calls between two PCs directly connected by a LAN, with no interfering traffic. In order to emulate different network conditions, we control the packet loss rate and the available bottleneck bandwidth by running NIST Net [16] on the Linux box.

A. Reaction to Losses

To analyze Skype reaction to packet losses, we impose a Bernoulli message loss probability increasing from 1% to 10%, with 1% step increment every 45 s, and record the corresponding packet level trace. Fig. 3-(a) reports the inter-packet-gap IPG (top) and the message size L (bottom) observed during a voice call encoded by ISAC; for reference purposes, the solid line reports the enforced loss probability.

During the first 45 seconds, no artificial loss is enforced. At the very beginning of the connection $t < 20$ s, the message size alternates between small (around 125 Bytes) and large (around 250 Bytes) values; then, during $20 \leq t < 45$ s, Skype emits only small-sized messages. Thus, whenever a Skype call starts and network conditions are still unknown, Skype possibly multiplexes two voice blocks into the same message – so to reduce the impact of possible packet losses and to proactively enforce a good QoS for the end-user (still, a *mix* of double-sized and single-sized messages are present, meaning that not all voice blocks are sent twice). Then, once Skype detects that no losses occur in the network, it stops retransmitting old voice blocks along with newer ones, so that the message size stabilizes around small values. When $t > 45$ s, the link starts discarding packets: as it can be seen, Skype starts retransmitting old-blocks along with new ones as soon as the loss probability exceeds 0%. The old-blocks retransmission probability changes along with loss probability: the vast majority of messages have no old-blocks multiplexed until losses exceed 4%, in which case multiplexing is used with few exceptions. When no losses are detected any more, at the end of the trace, old-blocks retransmission suddenly stops. Interestingly, the inter-packet-gap L measured before the loss point is not affected at all by the loss rate, meaning that Skype do not change neither the VBR codec state nor the framing time to react to packet loss.

B. Reaction to Bandwidth

Let us now investigate the impact of the available end-to-end bandwidth on IPG and L. Fig. 3-(b) reports the results obtained by restricting the available bandwidth according to the pattern shown by the solid line. When the available bandwidth is larger than the actual bitrate, no changes are observed with respect to the behavior previously shown in Fig. 3-(a): at the beginning of the connection, Skype aggressively probes the network conditions, then stops multiplexing old voice blocks. However, as soon as the bandwidth limit kicks in (at about 150 s), Skype adapts the bitrate to the new constraint: the message size pattern changes and L takes smaller values, which means that the Codec is working at a lower bitrate. At the same time, the IPG values change to 20, 30 or 60 ms, suggesting that the Skype framer also modifies the framing time to reduce the protocol overhead.

V. Discussion and Conclusions

This paper proposes a detailed analysis of Skype sources, via both active experiments in a controlled testbed and passive measurements from a real network. We consider *signaling* and *service* traffic: the first is related to the maintenance of the P2P overlay, whereas the second is due to voice and video calls among peers.

In the case of signaling traffic, bitrate is not a concern while the number of contacted peers over time could be. Signaling traffic can be discriminated in *probe* and *non-probe* traffic: the first type of signaling activity is composed of a pair of packets exchanged between two peers, and it is used throughout the whole peer lifetime to discover new nodes. Non-probe traffic, instead, is used periodically to exchange information about the status of peers of interest, and to support the overlay network maintenance. Interestingly, the wide majority of signaling flows are single packet probes: yet, an exiguous number of non-probe signaling flows carries the most relevant part of the exchanged signaling bytes. Concerning the Skype overlay, we have shown that the probing mechanism is latency driven, whereas the non-probe peers selection is driven by user habits: the Skype overlay is jointly shaped by the the combination of the this two mechanisms. Thus, peers with whom non-probe traffic is exchanged are selected on the top of the user social network – so to cluster friends. Conversely, the network discovery is carried on with a probing mechanism that takes into account the network characteristics as well – so to minimize the network latency. Finally, active testbed experiments suggest the amount of signaling traffic to be tied to the number of contacts in the user buddy lists, although a larger scale experiment would be needed in order to gather more precise quantitative relationship: an interesting direction that we plan to analyze in the future is to explore the correspondence of social-networks graphs versus the amount of observable Skype signaling flows.

Concerning the service traffic, we explored Skype reaction to network conditions in a controlled testbed: more in detail, our experiments suggest that Skype implements a "network aware" algorithm that measures the available bandwidth and the packet loss and reacts accordingly. Specifically, the possible Skype reactions to adverse network conditions include the tuning of: i) the message framing time, ii) the VBR Codecs bitrate and iii) the amount of redundant old-voice blocks multiplexed into new messages. More precisely, Skype internal algorithms react differently to path losses and network congestion: in the first case, Skype aggressively adds redundancy to encounter the effect of losses, whereas, in case of congestion, it conservatively tries to reduce its bitrate.

We acknowledge that the proposed measurement is forcibly incomplete as a consequence of both our methodology (e.g., single measurement point, small scale of the active testbed) and of the proprietary and obfuscated nature of Skype. Still, we believe this work to be a valuable contribution with two respects: on the one hand, to the understanding of a very popular P2P application and of its traffic dynamics; on the other hand, the provided Skype source description could be readily implemented on one of the many available P2P simulators, enriching the simulation realism.

References

[1] P. Barford , M. Crovella, "Generating representative Web workloads for network and server performance evaluation," *ACM SIGMETRICS'98* Madison, Wisconsin, USA, Jun. 1998

[2] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, J. Zahorjan. "Measurement, Modeling and Analysis of a Peer-to-Peer File-Sharing Workload." *In Proc. ACM SOSP'03*, NY, USA, Oct 2003.

[3] D. Stutzbach, S. Zhao, R. Rejaie, "Characterizing Files in the Gnutella Network," *Multimedia Systems Journal*, 2007.

[4] D. Stutzbach, R. Rejaie, "Understanding Churn in Peer-to- Peer Networks," *In Proc. of ACM Internet Measurement Conference IMC'06*, Brazil, Oct. 2006.

[5] A. Klemm, C. Lindemann, M. K. Vernon, O. P. Waldhorst. "Characterizing the query behavior in peer-to-peer file sharing systems," *In Proc. of ACM Internet Measurement Conference IMC'04*, Italy, Oct. 2004.

[6] "International carriers' traffic grows despite Skype popularity", http://www.telegeography.com/, Dec. 2006.

[7] P. Biondi, F. Desclaux, "Silver Needle in the Skype." *Black Hat Europe'06*, Amsterdam, the Netherlands, Mar. 2006.

[8] S. A., Baset, H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol." *IEEE Infocom'06*, Spain, Apr. 2006.

[9] K. Suh, D. R. Figueiredo, J. Kurose, D. Towsley, "Characterizing and detecting relayed traffic: A case study using Skype.", *IEEE Infocom'06*, Barcelona, Spain, Apr. 2006.

[10] D.Bonfiglio, M.Mellia, M. Meo, D.Rossi, P.Tofanelli, "Revealing Skype Traffic: when randomness plays with you", *ACM Sigcomm'07*, Kyoto, Japan, Aug. 2006.

[11] S. Guha, N. Daswani and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System", *International Workshop on Peer-to-Peer Systems*, Santa Barbara, CA, Feb. 2006.

[12] K. Ta Chen, C. Y. Huang, P. Huang, C. L. Lei "Quantifying Skype User Satisfaction", *ACM Sigcomm'06*, Italy, Sep. 2006.

[13] D.Rossi, M.Mellia, M.Meo, "Following Skype Signalling Footstep", IEEE IT-NEWS (QOS-IP) 2008, Venice, Italy, Feb. 2008.

[14] GlobalIPSound web site, http://www.globalipsound.com

[15] On2 web site, http://www.on2.com

[16] M. Carson, D. Santay, "NIST Net: a Linux-based network emulation tool." *ACM SIGCOMM Computer Communication Review*, July 2003.