Tracking Down Skype Traffic

Dario Bonfiglio, Marco Mellia, Michela Meo, Nicolò Ritacca
Politecnico di Torino – Dipartimento di Elettronica
email: name.surname@polito.it

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

***Abstract*—Skype is beyond any doubt the most popular VoIP application in the current Internet application spectrum. Its amazing success drawn the attention of telecom operators and the research community, both interested in knowing Skype's internal mechanisms, characterizing traffic and understanding users' behavior.**

In this paper, we dissect the following fundamental components: data traffic generated by voice and video communication, and signaling traffic generated by Skype. Our approach is twofold, as we make use of both active and passive measurement techniques to gather a deep understanding on the traffic Skype generates. From extensive testbed experiments, we devise a source model which takes into account: i) the service type, i.e., voice or video calls ii) the selected source Codec, iii) the adopted transport-layer protocol, and iv) network conditions. Furthermore, leveraging on the use of an accurate Skype classification engine that we recently proposed, we study and characterize Skype traffic based on extensive passive measurements collected from our campus LAN.

I. Introduction

The last few years witnessed VoIP telephony gaining a tremendous popularity, as testified by the increasing number of operators that are offering VoIP-based phone services. Skype [1] is beyond doubt the most amazing example of this new phenomenon: 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 [2].

Being the most popular and successful VoIP application, Skype is attracting the attention of the research community [3], [4], [5], [6], [7], [8], and of the telecom operator as well. However, many interesting questions related to its internal mechanisms, the traffic it generates and the behavior of its users' remain, to date, unanswered. The complexity stems from the fact that Skype protocols are proprietary, and that an extensive use of cryptography, obfuscation and anti reverse-engineering techniques [5] are adopted by Skype creators. Finally, Skype implements a number of techniques to circumvent NAT and firewall limitations [4], which add further complexity to an already blurred picture.

In previous work, we devised a methodology that successfully tackles the problem of Skype voice traffic identification [3]. This work aims at contributing to the understanding of Skype mechanisms and traffic in two main directions. First, by refining the source model of [3] via a wider set of active measurements and, second, by performing a characterization of real traffic by means of passive measurements.

The main contributions of this paper are the following. First, we characterize the traffic generated by voice and video calls, by observing their time evolution and the distribution of indexes such as the bit rate, the inter-packet gap, the packet size. Besides distinguishing among various voice Codecs that Skype adopts, we also unveil the different behavior of the traffic source based on the adopted transport layer protocol. Second, we observe how Skype reacts to different and changing network conditions, so that we can assess their impact on the traffic generated by a Skype source. Third, we focus on the users' behavior by analyzing the number of flows generated in the time unit and the call duration – which unsurprisingly is very much related to the tariff policies. Fourth, we analyze the signaling traffic generated by a Skype client, considering also the number of different clients that are contacted by a peer, which gives a feeling about the cost of maintaining the P2P architecture. Finally, we briefly describe how the classification tool proposed in [3] has been extended to cope with videocalls.

While many details about the Skype protocols and internals can be found in [4], [5], few papers deals with the issues of Skype identification [3], [8], and characterization of its traffic and its users [6], [7]. In [8], authors focus on the *identification* of relayed[1], rather than direct traffic, using Skype as an example of application: little results are therefore presented about Skype source characterization. The work in [6] presents an experimental study of Skype, based on a five month long measurement campaign. Lacking a reliable Skype classification engine, authors are again forced to limit the scope to relayed sessions, and they restrict furthermore their attention to the case of UDP transport layer only. The work closest to ours is [7], in which authors focus on the evaluation of the QoS level provided by Skype calls. As the adopted VoIP traffic classification criterion is fairly simple, authors cannot distinguish between video and voice, end-to-end and Skypeout calls, and cannot account for the impact of transport protocols. All previous papers completely ignore Skype signaling traffic except [4], although the focus is different – i.e., they analyze the login phase, and how Skype traverses NAT and firewalls rather than providing quantitative insights on the amount and destination of Skype signaling traffic.

In this paper we instead provide a detailed characterization of Skype traffic, exploiting and refining the fine-grained classification of [3]. After having briefly summarized Skype features in Sec. II, we characterize Skype source in Sec. III and we show how Skype reacts to network congestion and losses. We then analyze the typical Skype users' behavior in Sec. IV, whereas Sec. V quantifies the signaling overhead at both the

[1]A session is *relayed* if packets from a source to a destination are routed through an intermediate node which acts as an application layer relay.

TABLE I
NOMINAL CHARACTERISTICS OF SKYPE CODECS.

Codec	Frame Size [ms]	Bitrate [kbps]
ISAC*	30,60	$10 \div 32$
ILBC	20,30	13.3, 15.2
G.729	10	8
iPCM-wb*	10,20,30,40	80 (mean)
EG.711A/U	10,20,30,40	48,56,64
PCM A/U	10,20,30,40	64
TrueMotion VP7	Unknown	> 20

$*$ *denotes wideband Codec*

network and transport layers (i.e., in terms of packets and flows that Skype generates even when users are idle). Finally, Sec. VI summarizes our findings, while details related to the videocall extension of the classification engine are reported in the Appendix.

II. SKYPE PREMIER

The main difference between Skype and other VoIP clients is that Skype operates on a P2P model, rather than a more traditional client-server model. Only user's authentication is performed under a classical client-server architecture, using public key mechanisms. After the user (and the client) has been authenticated, all further signaling is performed on the P2P network, so that Skype user's informations (e.g. contact list, status, preferences, etc.) are entirely decentralized and distributed among P2P nodes. This allows the service to scale very easily to large sizes, avoiding furthermore a costly centralized infrastructure.

Peers in the P2P architecture can be normal nodes or supernodes. The latter ones are selected among peers with large computational power and good connectivity (considering bandwidth, uptime and absence of firewalls), so that they take part to the decentralized information distribution system which is based on a DHT.

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 a supernode to ease the traversal of symmetric NATs and firewalls. Voice calls can also be directed toward the PSTN using Skypein/Skypeout services, 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.

From a protocol perspective, Skype uses a proprietary solution which is difficult to reverse engineer due to extensive use of both cryptography and obfuscation techniques [3], [4], [5]. Though Skype may rely on either TCP or UDP at the transport layer, both signaling and communication data are preferentially carried over UDP. A single random port is selected during application installation, and it is never changed (unless forced by the user). When a UDP communication is impossible, Skype falls back to TCP, listening to the same random port whenever possible, or using port 80 and 443

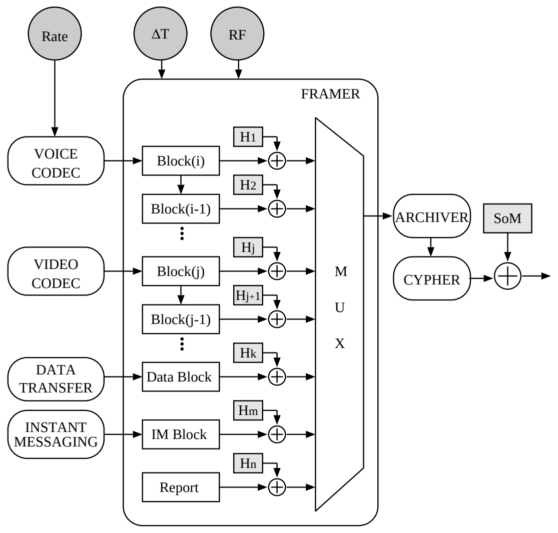

Fig. 1. Schematic diagram representing the Skype message building process.

which are normally left open by network administrators to allow Web browsing.

For what concerns the voice service, Skype can select between different Codecs according to an unknown algorithm. It is however possible to force Codec selection and we exploit this feature to observe the different behavior of the Skype source when using different Codecs. The Codec name, nominal frame size and bitrate are reported in Tab.I, where Wideband Codec (offering 8 kHz bandwidth) are labeled by a "$*$" symbol. All Codecs are standard except the ISAC one, which is a proprietary solution of GlobalIPSound [9]. Some are Constant Bitrate (CBR), while others are Variable Bitrate (VBR) Codecs. ISAC is the preferred Codec for E2E (End-to-end) calls, while the G.729 Codec is preferred for E2O (Skypeout) calls. For what concerns the video, Skype adopts TrueMotion VP7 Codec, a proprietary solution of On2 [10], which provides a variable bitrate flow with minimum bandwidth of 20 kbps. No other detail is available. In this paper, we focus on the characterization of both voice and video communication services, being them the most popular and peculiar Skype services, and of the signaling traffic peers generate.

III. VOICE AND VIDEO STREAMS CHARACTERIZATION

In order to derive a source model, we performed several experiments in a controlled environment: our testbed involved several PCs connected by a Linux NAT / Firewall / Router / Traffic-Analyzer boxes. Different versions of Skype were installed, running under different operating systems such as Windows, Linux and Pocket-PC. Several network scenarios were emulated by using NIST Net [11] to enforce various combinations of delay, packet loss and bottleneck bandwidth, so to observe how Skype reacts to different network conditions.

A monodirectional flow is identified by using the traditional tuple (IP source and destination addresses, UDP/TCP source

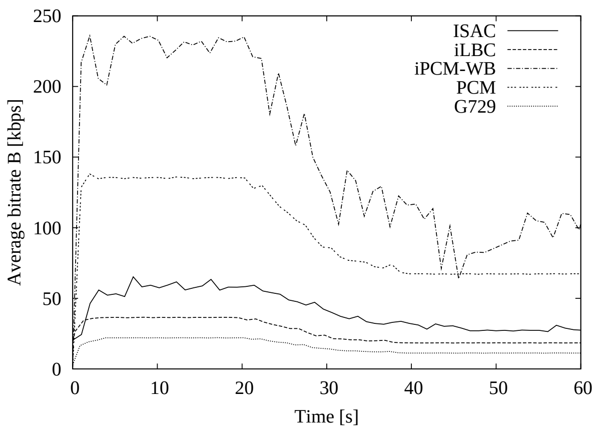

Fig. 2. Bitrate traces versus time for different voice Codec types - UDP at the transport layer, no artificial delay and loss.

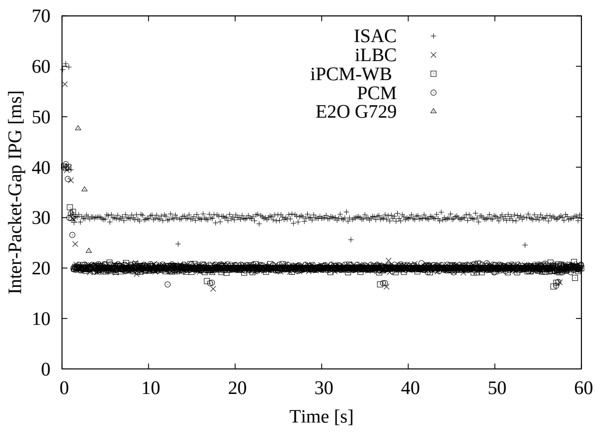

Fig. 3. IPG traces versus time for different voice Codec types - UDP at the transport layer, no artificial delay and loss.

and destination ports, IP protocol type)[2]. A flow starts when a packet with the flow tuple is first observed, while it is ended by either an inactivity timeout (conservatively set to 100 s) or, in case of TCP, by observing the connection tear-down sequence if present. Flow characterization is provided by the following measurement indexes, which are typical of streaming services over packet networks:

- Average Bitrate (B): the average amount of bits generated at application layer in a time interval of 1 second.
- Inter-Packet-Gap (IPG): the time elapsed between two consecutive packets belonging to the same flow.
- Payload length (L): the number of bytes carried by TCP or UDP. The corresponding IP packet size can be determined by adding the transport and network layer overheads.

We use the Skype source model proposed in our previous work [3] and sketched in Fig. 1. The source generates information *blocks* that can be voice/video/data/chat/report blocks. In order to cope with the potential loss of the immediately preceding frames or to modify the message generation rate, one or more blocks can be multiplexed in a *frame*. Once a frame has been created, it is then arithmetically compressed and encrypted. Finally, an additional non-ciphered header (called Start of Message - SoM) may be present too. The output of this process is a Skype *message*, that is then encapsulated in either a UDP or TCP segment. At the input side, three parameters determine the characteristics of the generated traffic:

- *Rate* is the bitrate used by the source, e.g., the Codec rate used for the communication;
- ΔT, that represents the Skype message framing time, is the time elapsed between two subsequent Skype messages belonging to the same flow;
- *RF* is the Redundancy Factor, i.e., the number of past blocks that Skype retransmits, independently from the adopted Codec, along with the current encoded block.

[2]We separately analyze and track monodirectional flows, so that each call is built by two flows.

We point out that the above parameters are not fixed but changes during an ongoing call: as we show in the following, Codec *Rate* and *RF* are the preferred knobs used by Skype to react to changing network conditions, but ΔT is also frequently modified as well.

A. Voice flows characterization

In this section we analyze the traffic generated by voice flows. We perform a first set of experiments by generating voice calls between two PCs directly connected by a LAN with no interfering traffic, and no imposed artificial delay or packet loss. We force the voice Codec, and record for each experiment a packet level trace. Flows transported by UDP, the preferred transport protocol, are considered.

Figures 2, 3 and 4 report versus time for different voice Codecs the bitrate B averaged over 1s time intervars, the inter-packet-gap IPG, and the payload length L. Due to the different characteristics of each Codec, a voice call can consume up to 230 kbps and as few as 11 kbps. Independently from the adopted Codec, three phases can be easily distinguished in the traces: during the first 20 s, the bitrate is high; then, a transient period between 20 and 40 s follows, where the bitrate smoothly decreases; finally, during the third portion of the trace, $t \geq 40$ s, the bitrate is roughly half the one at the trace beginning. This is likely due to an initial setting $RF = 2$. This setting is typical of bad network conditions, and it aims at reducing the impact of possible losses: it is apparently used during the flow initial phase, when network conditions are unknown, in order to aggressively enforce high quality call. After a short time, Skype realizes that network conditions are good and RF is set to 1. Observing the IPG in Fig. 3, it can be noted that, after an initial transition, IPG is constant during the three phases, meaning that the bitrate variability is not obtained modifying the IPG. Notice also that, during the very beginning of the traces (roughly 1 s), Skype performs a frame size tuning, reflected in the IPG taking values in 30,40,60 ms before assuming the regime value which is equal to 30 ms for ISAC and 20 ms for all the other Codecs.

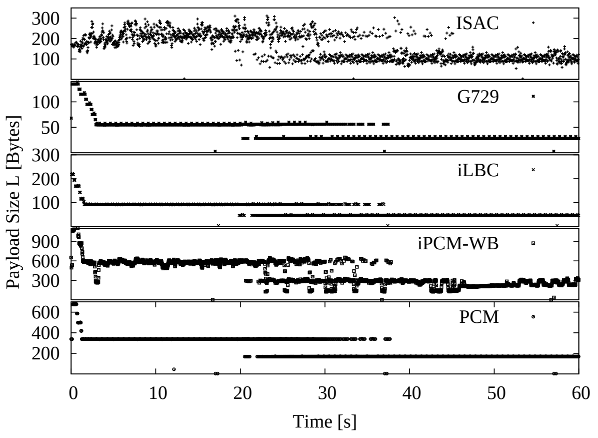

Fig. 4. L traces versus time for different voice Codec types - UDP at the transport layer, no artificial delay loss.

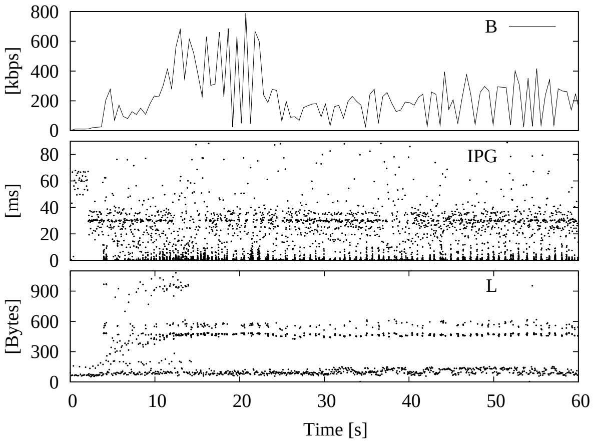

Fig. 6. B, IPG and L traces versus time for a videocall - UDP at the transport layer, no artificial loss, ISAC Codec.

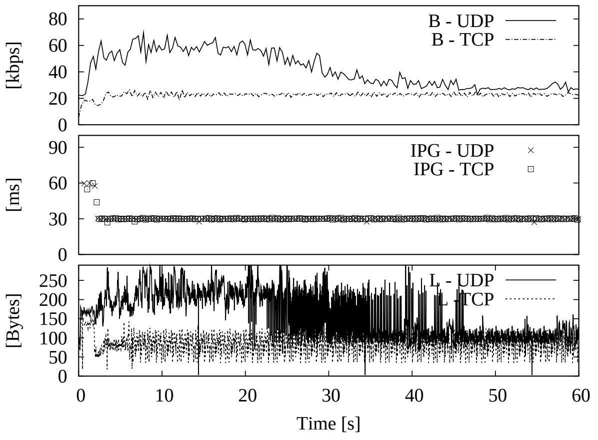

Fig. 5. Bitrate, IPG and message size traces versus time - UDP or TCP at the transport layer, no artificial loss, ISAC Codec.

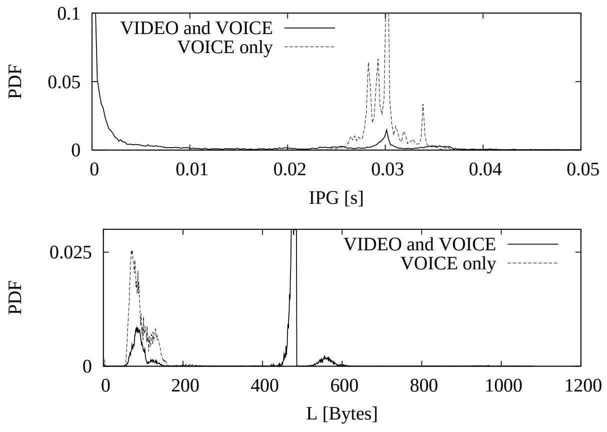

Fig. 7. L and IPG PDFs for pure voice or video and voice streams - UDP at the transport layer, no artificial loss, ISAC Codec.

The variation of B is obtained by Skype modifying the message size L, as Fig. 4 clearly shows. Indeed, messages of double size are transmitted during the initial trace portion, while a mix of double-sized and single-sized messages are observed during the transient phase. This is due to Skype applying a reframing to include more than one Codec block in the same message, e.g., $RF = 2$, but possibly not to all blocks. Notice that VBR Codecs, such as ISAC and iPCM-wb, exhibit larger message size variance, while when CBR Codecs are adopted (e.g., G.729, iLBC and PCM) Skype messages are almost constant sized: in this case, the small but noticeable message size variability is tied to report blocks piggybacked by Skype onto message. Notice that during the transient period, the bitrate exhibits a smooth decrease, whereas message sizes achieve only two possible values. This means that Skype precisely controls the frequency of RF changes, in order to shape the resulting bitrate. It is also possible to observe that L is larger at the very beginning, being the initial ΔT larger too.

We now consider the case of a voice flow transported by TCP. We use the same testbed scenario previously described and repeat all experiments presented above, after having imposed TCP as the transport protocol by means of a firewall rule. The results are presented in Fig. 5 considering the ISAC Codec. Observe that, when using TCP, Skype always sets $RF = 1$. Indeed, since TCP guarantees to recover packet losses, there is no need for setting RF to 2. A couple of additional observations are also worth: first, the SoM header is not present, and the message size is 4 bytes shorter; second, ΔT is still variable, as shown at the initial portion of the trace.

Notice also that the TCP congestion control and segmentation algorithms do not alter L and IPG. This is due to the fact the during the test, no loss was present, so that the TCP congestion window was unbounded. We also suspect that Skype uses the `TCP_NODELAY` socket option to disable Nagles'algorithm, so that the time delays between messages are maintained.

B. Video Flows Characterization

In order to analyze the traffic generated by voice flow, we repeat the same experiments as in the previous section, enabling the video source after about 5 s. Voice Codec is left to

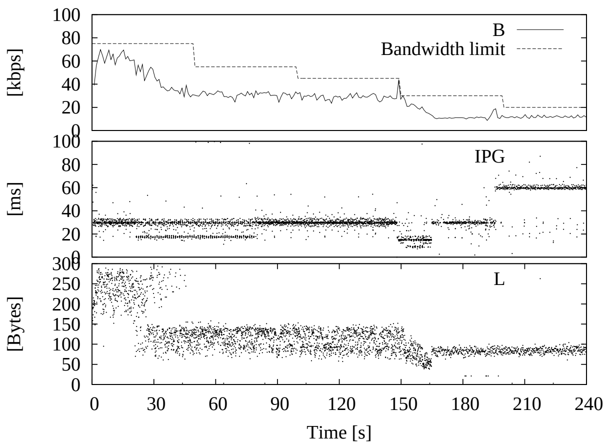

Fig. 8. B, IPG and L during a voice call under decreasing available bandwidth - UDP at the transport layer.

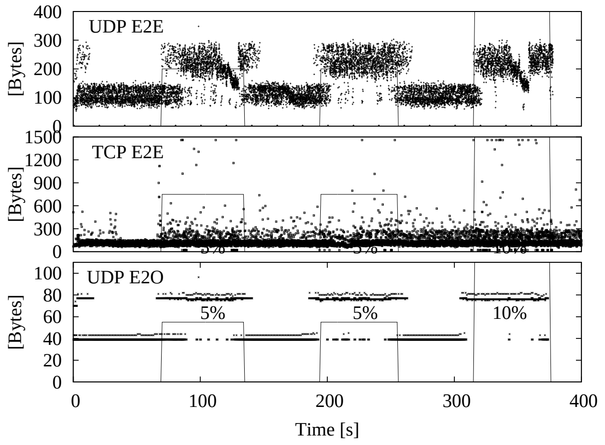

Fig. 9. L traces versus time for a UDP E2E call (ISAC Codec), UDP E2O call (G729 Codec) and TCP E2E call (ISAC Codec) - on-off artifical loss scenario.

the default ISAC choice and UDP is used as transport protocol; neither artificial delay nor loss are imposed.

Results are presented in Fig. 6. From the average bitrate time evolution (top plot), it can be noticed a significantly increased variability with respect to the case of voice flows, ranging from a few kbps up to 800 kbps. Investigating the IPG process (middle plot), it can be observed that the IPG is less regular than in the voice-only case. Indeed, a large number of IPG samples is about 30 ms (the preferred ISAC ΔT), while many other IPG samples are very small. This is due to the fact that Skype is multiplexing voice and video blocks: the first ones are produced by the corresponding voice Codec at a very regular rate; the latter ones are instead bigger, and therfore they are segmented by Skype and transmitted using multiple back-to-back messages. This is reflected by L plot at the bottom of Fig. 6. Let us first focus on the period $t > 20$ s, when $RF = 1$. It is possible to identify three typical message sizes: i) $L \in [0, 150]$ Bytes, for messages containing voice blocks only , ii) $L \in [350, 490]$ Bytes, for messages containing video blocks only, and iii) $L \in [491, 500]$ Bytes when voice and video blocks are multiplexed in a single message. The message size doubles if $RF = 2$, e.g., when $t \in [5, 15]$ s. This behavior is highlighted by the Probability Density Functions (PDF) of L and IPG of a voice only and video plus voice flows, as reported in Fig. 7.

C. Impact of Different Network Conditions

Let us now investigate the impact on the traffic generated by Skype of different network conditions, namely: i) available end-to-end bandwidth, ii) loss probability, and iii) source-destination path delay.

Fig. 8 reports measurements obtained during a voice call between two clients in which we artificially enforced the available bandwidth. Top plot reports B and the imposed bandwidth limit; middle plot reports IPG, and bottom plot reports L. UDP was selected at the transport protocol, and the default ISAC Codec was used. The usual 20 s long initial period is present, in which $RF = 2$. When the available bandwidth is larger than the actual bitrate, no changes are observed with respect to the typical source behavior shown in Fig. 2. As soon as the available bandwidth limit kicks in (after about 150 s), the source adapts B to the new constraints. This is reflected by a change in the message size pattern, since L is constrained to take smaller values, which suggests that the Codec selected a low-bitrate state (recall that the ISAC codec is a VBR Codec). At the same time, the IPG values change to 20, 30 or 60 ms, hinting that the Skype framer modifies the framing time to reduce the protocol overhead. We can then state that Skype implements a congestion control protocol that acts both on the RF, ΔT and Codec bitrate.

We performed a second set of experiments to assess the impact of network losses. Fig. 9 plots the message size B observed during a voice call when artificial packet losses are introduced, (neither bandwidth limit nor artificial delay are present). In particular, time periods with no losses alternate to time periods during which 5% or 10% loss probability is enforced. Results considering a UDP-E2E and TCP-E2E flows (VBR ISAC Codec), UDP-E2O flow (CBR G.729 Codec) are reported in the Fig. 9. Consider first the UDP case. When some losses are detected, Skype implements a greedy policy to mitigate their impact by retransmitting past voice blocks into the same message, i.e., $RF = 2$; on the contrary, when no loss is detected, Skype sets $RF = 1$. This holds for both E2E and E2O, and for both voice and video calls (the latter E2E video case is not reported here due to lack of space). Conversely, if TCP is adopted, no loss concealment mechanism is implemented by Skype, which completely relies on TCP loss recovery mechanism. This results in a much more complex L pattern, since TCP congestion control and segmentation algorithms impose a different framing pattern to the application stream. For example, if a loss is recovered after that the retransmission timeout expired, data buffered at the socket will be immediately sent in one (or more) larger TCP segments.

In order to find out at which average loss rate Skype source triggers the concealment mechanisms, we consider a UDP flow (ISAC Codec) facing increasing average loss from 0% to 10%

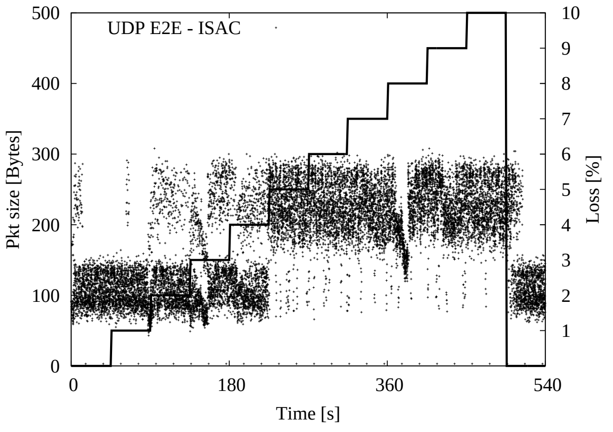

Fig. 10. L traces versus time for a UDP E2E call (ISAC Codec) - increasing artifical loss scenario.

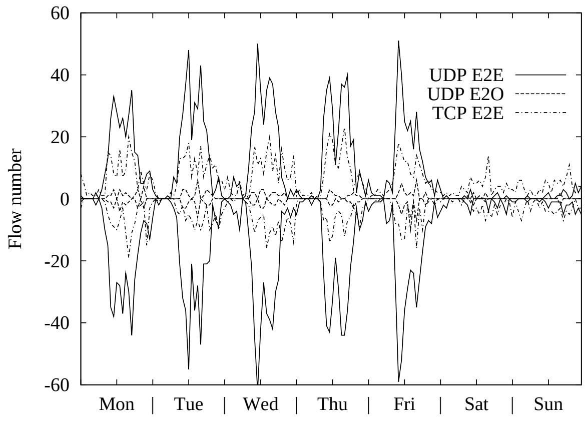

Fig. 11. Number of UDP E2E, TCP E2E and UDP E2O voice calls every 1 hour. Incoming flows on positive values, outgoing flows on negative values.

with 1% step increment every 45 s. Measurements are reported in Fig 10. As it can be seen, Skype selects $RF \geq 1$ as soon as the loss probability exceeds 1%. Conversely, if no losses are detected (e.g., at the end of the trace), RF is set to 1 again. However, it can be noticed that not only the RF range changes, but also the relative occurrence of specific RF values changes as a function of the loss rate: indeed, the vast majority of messages use $RF = 1$ until losses exceed 4%, in which case $RF = 2$ is used with few exceptions.

Some tests were also performed to assess the impact of network delay: no change was observed (and therefore we do not report results). This is quite intuitive, since there is no major countermeasure that a real-time application can implement if the end-to-end delay is large due to physical constraints such as distance.

Comparing the Skype reactions in the above network conditions, we can gather an important remark: when using UDP at the transport layer, Skype not only measures the loss probability, but also implements some technique to measure the available bandwidth. For the sake of clarity, let us consider a specific case, namely E2E calls using the ISAC Codec, as reported in Fig. 8 and in Fig. 9 top plot. In the scenario of Fig. 8, through the probing phase after $t = 150$, Skype determines that the low call quality is due to network congestion (rather than to path losses). It therefore sets $RF = 1$ to avoid overloading the network. Conversely, Fig. 9 shows that some probing phases occur during the time intervals where losses are present. Skype is able to ascribe the low call quality to path losses (rather than to network congestion), and therefore sets $RF = 2$ in the attempt to mitigate loss impact and ameliorate call quality. We conclude that Skype estimates both the available bandwidth and the loss probability: it then implements a technique to adapt to the detected network conditions, reacting by either tuning the bitrate or introducing higher redundancy.

IV. User Characterization

In this section we analyze some characteristics of Skype users' behavior. We report results that were collected by passive monitoring our campus access link, and applying the classification framework presented in [3]. We monitored our campus access link for more than a month starting from April the 22nd 2007. More than 7000 different hosts are present in our LAN, which is used by both students and staff members. The classificator proved to be very robust producing practically no false positives [3], even considering the extension to video-call identification according to the algorithm presented in the Appendix. The total number of flows that were identified are 17595, 9136, 1393 and 1145 considering UDP E2E, TCP E2E, UDP E2O voice and UDP video calls, respectively. Notice that most of the calls are "free" E2E voice calls, with video enabled in only 6% of UDP E2E flows.

Fig. 11 reports the number of calls per hour in a typical week, showing outgoing flows (source IP address belonging to the campus LAN, destination IP address not belonging to it) with positive values, and incoming flows with negative values. Skype preferred transport protocol is UDP, being it used in more than 68% of cases. Notice that this can dramatically change in a different network setup, e.g., when NAT or firewall are extensively used. As expected, the number of calls is larger during the working hours, with a negative bump during launch time, while during nights and weekends fewer calls are present. The peak number of calls accounts about 75 Skype calls per hour. Asymmetry is due to the fact that the two directions of the same call can use different transport layer protocols, which is observed on roughly 15% of the cases. Specifically, our campus is more likely to accept UDP connections, whereas other parties may be in more restrictive network conditions that force Skype to adopt TCP, as can be gathered by the smaller number of UDP E2E incoming flows with respect to the outgoing ones.

Fig. 12 shows the Cumulative Distribution Function (CDF) of flow holding time (i.e., the call duration), defined as the time elapsed from the first until the last packet of the flow. It can be noted that the holding time for E2E calls is much larger than the one of E2O calls. This can be justified by the fact that E2E calls are free. Notice also that the measured holding

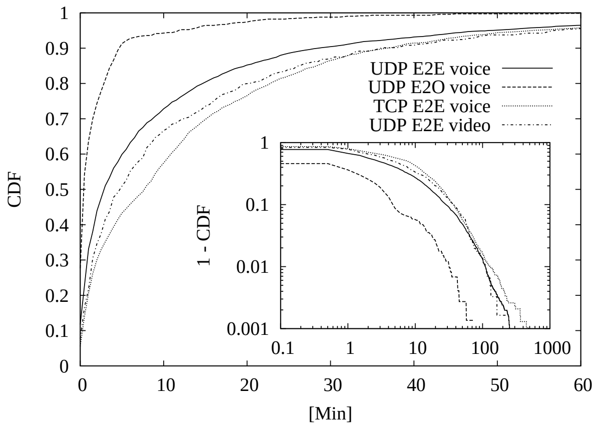

Fig. 12. Flow holding time CDF and 1-CDF in the inset.

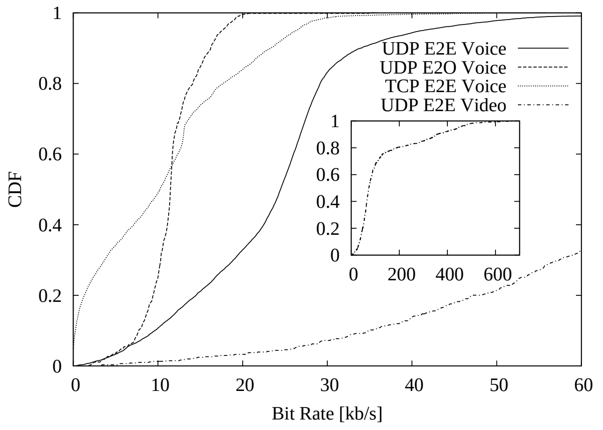

Fig. 13. Bitrate CDF for different voice and video Codec and transport protocols.

time is slightly larger when the video is enabled.

On the contrary, the larger TCP E2E holding time is at first surprising, since there is no reason for the user to talk more when TCP is adopted. Investigating further, we noticed that Skype delays the TCP tear down sequence, keeping the connection alive even if the call has been hung up. This protocol specific bias is difficult to remove. Note that this affects resource usage on both end hosts and the possible full-state NAT, since the TCP connection must be managed until the tear-down sequence is completed.

Fig. 13 shows the CDF of the average flow bitrate (averaged over the entire flow lifetime) of different flow types. The figure shows that UDP E2E flows exhibit a bitrate ranging from few kbps up to 50 kbps, since both the ISAC Codec is VBR, and RF can be larger than 1. TCP E2E flows exhibit bitrate values that are about half the previous case, since $RF = 1$ when TCP is adopted. Considering the UDP E2O case, we notice that the preferred G.729 Codec produces a less variable stream bitrate, being it a CBR Codec. The variability of the E2O flow bitrate is due to Skype varying the RF factor to cope with network losses. Finally, videocall bitrate takes much larger values, ranging up to 500 kbps, the average bitrate being 193 kbps.

V. Signaling Characterization

In the following, we focus on the signaling activity of Skype peers[3], considering the same week of Fig.11. Let us start by considering the schematic representation of the time evolution of the typical Skype activity pattern depicted in Fig. 14. We select two specific peers, namely the most active peer that do not perform any call (left plot) and a randomly picked peer having both signaling and voice flows (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

[3]To identify Skype signalling we leverage on the identification of the socket address used by Skype for a given host.

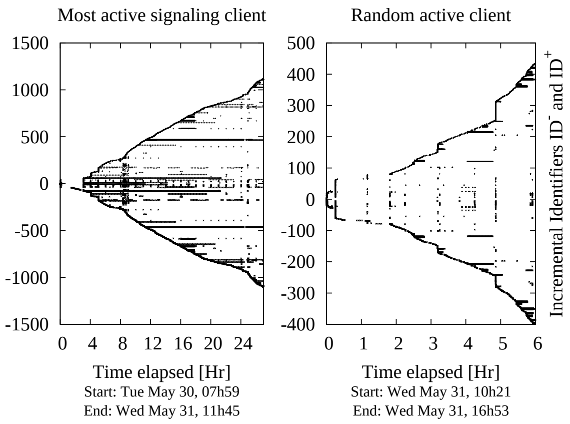

Fig. 14. Pictorial representation of Skype activity pattern for most active (left) and a random peer (right).

sent messages to p. The range of the y-values corresponds to the number of different Skype peers with whom the selected peer p is exchanging messages.

The figure shows that the most active peer has contacted (was contacted by) about 1100 other peers, whereas the random peer by about 450. Interestingly, the number of contacted peers exhibits an almost linear growth with time, hinting to P2P network discovery being carried on during most of the peer lifetime. Signaling is mainly built by single message probes, to which (most of the times) some kind of acknowledgment follows. Some of the 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. 14 every hour). The fact that p knows the address and ports of valid (but previously un-contacted) Skype peers means that the above information is carried by signaling messages.

To give better intuition of signaling message generation process, the inter-packet gap CDF over *all* packets generated

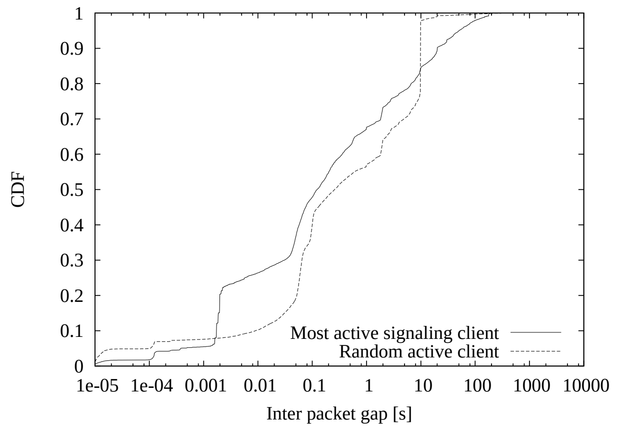

Fig. 15. Inter packet gap distribution for random and most active peers

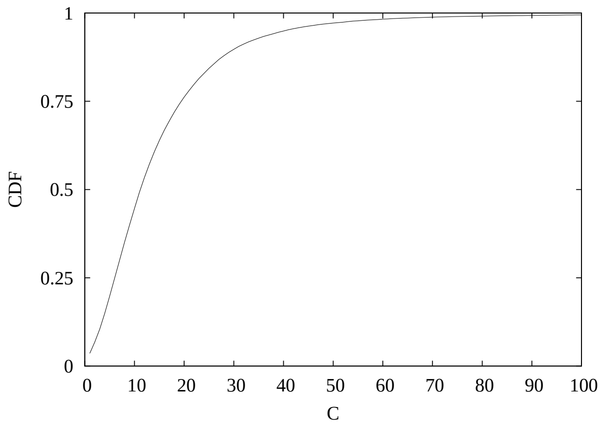

Fig. 16. Distribution of the number of peers contacted by internal clients during time-windows of 300 seconds.

by peer p is reported in Fig. 15. The main result is that the different types of asynchronous parallel activities run by Skype are such that the inter-packet gap is more uniformly distributed than expected. Still, different timers are visible that correspond to different types of activity. For example, considering the randomly selected client (dotted line) and starting from small timescales, a small fraction of packets is spaced by tens of milliseconds; these packets are very likely due to: i) the parallel probing that is triggered periodically every hour, and possibly to ii) packet-pair techniques used to probe the available bandwidth as earlier conjectured. The steep increase at around 30-60 milliseconds is due to inter-packet gap typical of voice calls. Finally packets spaced by tens of seconds are keep-alive messages sent ,e.g., to force NAT entries refresh. Considering the case of the most active signaling client (solid line), typical voice timings are no longer present, while there is a visible peak at about 2 ms, probably due to burstiness in the parallel probing of several contacts.

Further insight about Skype signaling activity is given by Fig. 16. Let $C(p, i)$ be the number of different peers contacted by peer p considering the i-th time interval of 5 minutes since the start of peer p activity. Intuitively, this metric expresses the number of signaling flows that p generates in the time unit. Distribution of $C(p, i)$ over all internal peers and over the whole measurement interval is shown in Fig. 16. In 90% of cases, the number of signaling flows generated in 5 minutes is smaller than 30, with mean number equal to 16. In 1% of the cases, this number is larger than 75. Note that this metric is of particular interest since it is related to the burden a Skype client poses in any layer-4 device that keeps per flow state, e.g., a entry in a NAT table, a lookup in a firewall ACL table, etc. As Fig. 14 showed, many signaling flows are single-packet probes that create new temporary soft-state entries, rarely used later on.

To complete the signaling traffic characterization, flow length (in bytes) and duration (in seconds) complementary distribution are reported in Fig. 17, where both axis are in log-scale. As already noted, about 80% of the signaling flows consists of single packet probes. This percentage exceeds 99%

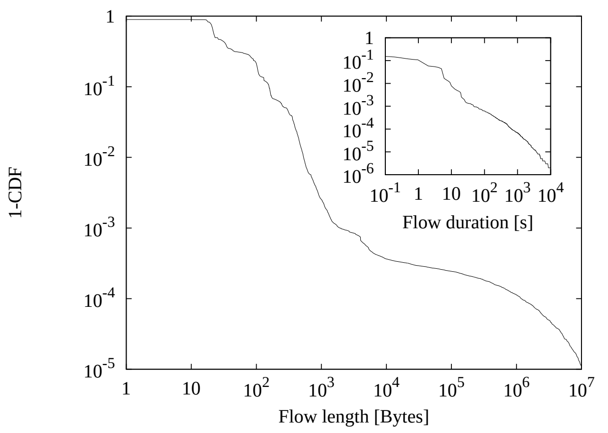

Fig. 17. Distribution of the signaling flow size (outset) and duration (inset).

when we consider flows shorter than 6 packets. As most of the flows are single packet probes, the bulk of the signaling flows duration and size is very small: e.g., 50% of flows carry no more than 25 bytes of payload (more than 25% of which are single-packet probes of 21–23 bytes); moreover, 90% of the flows are shorter than 1 second and carry about 150 bytes and 99% of the flows are shorter than 10 seconds and are about 500 bytes long. At the same time, is it possible to observe persistent signaling activity transferring a few MBytes of information over several thousand packets and lasting for hours, as the tails of Fig. 17 show: indeed, the single packet probes account not even for 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.

To gauge the signaling overhand Skype client generates, Fig. 18 reports the CDF of peer average bitrate evaluated as the total signaling messages bits transmitted by a client during its whole lifetime. It shows that the additional costs is almost marginal, accounting to less than 100bps in 95% of cases, while very few nodes generates more than 1kbps of average signaling bitrate (possibly supernodes).

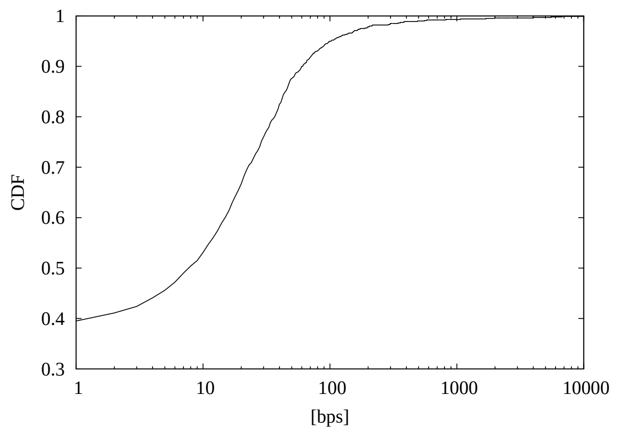

Fig. 18. Distribution of peer signalling average bitrate.

VI. Conclusions

This paper focused on the characterization of Skype, the most popular VoIP application. Our approach is twofold. First, from extensive testbed experiments we enlighten several aspects of the Skype source, considering different service types (i.e., End2End, Skypeout voice and video calls), transport protocols (i.e., TCP, UDP), and network conditions (i.e., loss, available bandwidth and path delay). Testbed measurements are used to refine the picture on the Skype source model, showing what type of mechanisms are used and which conditions trigger them in order to adapt to the different network conditions: specifically, when UDP is used at the transport layer, our measurements show evidence that Skype internal algorithms differently react to path losses and network congestion. Second, leveraging on a consolidated methodology for fine-grained Skype traffic classification, we investigated by means of passive measurements both i) Skype users' behavior and the traffic generated during voice and video communications, and ii) the signaling traffic generated by Skype. Concerning signaling, we have shown that Skype prefers to flood the network with short single-probes toward many hosts – which may be as effective for the purpose of the overlay maintenance as costly from the viewpoint of statefull layer-4 network devices.

VII. Acknowledgments

The Italian team was supported by the Italian Ministry of University, Education and Research (MIUR) under the PRIN RECIPE, and by a research contract with Vodafone Italia. We would like to thank Politecnico di Torino Campus LAN Network Administrators for allowing us to monitor traffic.

References

[1] Skype web site, http://www.skype.com
[2] "International carriers' traffic grows despite Skype popularity", TeleGeography Report and Database. available on line http://www.telegeography.com/, Dec. 2006.
[3] 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.
[4] S. A., Baset, H. Schulzrinne, "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol." *IEEE Infocom'06*, Barcelona, Spain, Apr. 2006.
[5] P. Biondi, F. Desclaux, "Silver Needle in the Skype." *Black Hat Europe'06*, Amsterdam, the Netherlands, Mar. 2006.
[6] S. Guha, N. Daswani and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System", *5th International Workshop on Peer-to-Peer Systems*, Santa Barbara, CA, Feb. 2006.
[7] K. Ta Chen, C. Y. Huang, P. Huang, C. L. Lei "Quantifying Skype User Satisfaction", *ACM Sigcomm'06*, Pisa, Italy, Sep. 2006.
[8] K. Suh, D. R. Figuieredo, J. Kurose, D. Towsley, "Characterizing and detecting relayed traffic: A case study using Skype.", *IEEE Infocom'06*, Barcelona, Spain, Apr. 2006.
[9] GlobalIPSound web site, http://www.globalipsound.com
[10] On2 web site, http://www.on2.com
[11] M. Carson, D. Santay, "NIST Net: a Linux-based network emulation tool." *ACM SIGCOMM Computer Communication Review*, V.33, N.3, July 2003, pp:111-126.

Appendix

The analysis of video flows presented in Sec. III-B allows us to complete the voice call classification tool proposed in [3]. We briefly describe here how the classification tool has been enhanced.

The tool in [3] includes a Naive Bayesian Classifier (NBC), which is based on the stochastic characterization of voice call message length (L) and inter-packet gap (IPG). The NBC identifies a Skype flow when its characteristics are similar to the expected ones. A natural straightforward extension of the NBC to videocalls could thus consist to derive a new stochastic characterization of L and IPG. However, while this approach is very effective in identifying voice calls, it fails with videocalls. Indeed, the effectiveness with voice calls is based on the joint effect of the limited variance of L and IPG and the significant difference of the characterization of voice traffic with respect to other Internet traffic. Both these aspects cannot be exploited with videocalls, as i) video and voice block can be multiplexed on a single frame, and ii) the IPG is not distinctive being video blocks segmented by Skype into several messages and transmitted back-to-back. Our approach therefore consists in applying the NBC to the voice portion of the video call only, and separately detecting the presence of video. More on details, we simply avoid feeding the NBC with messages containing video blocks *only* (i.e., $L \in [350RF, 490RF]$ Bytes), so that the regularity of the voice block can be extracted from the multiplexed messages. After filtering, the IPG takes values that are typical of the Skype voice framing ($\Delta T \in \{20, 30, 40, 50, 60\}\, ms$), and the NBC correctly identifies the voice stream component of the videocall. Similarly, the L NBC correctly classifies the voice stream by feeding it with voice only messages (i.e., $L \in [0, 150RF]$ Bytes).

The presence of a video stream in the Skype call can then be reliably identified a posteriori, by counting the percentage of messages that *also* contain video blocks (i.e., messages larger than 350 Bytes).