Pictures from the Skype

Dario Rossi, Silvio Valenti, Paolo Veglia
TELECOM ParisTech, France
first.last@enst.fr

Dario Bonfiglio, Marco Mellia, Michela Meo
Politecnico di Torino, Italy
first.last@polito.it

ABSTRACT

This paper focuses on the characterization and classification of Skype traffic, a nowadays very popular and fashionable VoIP application. Building over previous work, we develop a software tool which can be used to examine the evolution of Skype call classification in an interactive fashion. The demonstrator software focuses on the main aspects of Skype traffic characterization and presents the traffic patterns Skype generates during a call or while idle. In addition, the demonstrator shows the evolution of the internal indexes the Skype classifiers use.

After describing the classification process and the demonstrator software, we use the tool to demonstrate the feasibility of online Skype traffic identification, considering both accuracy and computational costs. Experimental results show that few seconds of observation are enough to allow the classifier engines to correctly identify the presence of Skype flows. Moreover, results indicate that the classification engine can cope with multi-Gbps links in real-time using common off-the-shelf hardware.

Categories and Subject Descriptors

C.4 [**Computer Communication**]: Measurement Techniques; C.2.5 [**Computer Communication Network**]: Internet

General Terms

Demonstrator, Experimentation, Measurement

1. INTRODUCTION TO SKYPE

VoIP is currently becoming a synonym for telephony, and Skype is beyond any doubt the most amazing example of this new phenomenon: developed in 2003 by the creators of KaZaa, it recently reached over 100 millions of users, about 10 millions of which are on-line at the same time, becoming so popular that people indicate Skype IDs in their business cards. By building over the Skype classification engine we proposed in [1], we describe in this paper a demonstrator software whose goals are:

- to show Skype traffic generation, by plotting in real time the packet generation process of a Skype VoIP/video call;
- to demonstrate the Skype classification techniques proposed in [1];
- to monitor Skype activity when idle, showing the signalling messages exchanged to maintain the P2P overlay.

In [1], we proposed a classification framework to reveal the presence of Skype traffic within traffic aggregates. The classification engine, briefly overviewed in Sec. 2, relies on the joint use of two different and complementary techniques. The first approach is based a stochastic characterization of Skype traffic: inter-packet gap (IPG) and packet length are used as features of a decision process of the family of the *Naive Bayesian Classifiers* (NBC). The second technique, named *Chi-Square Classifier* (CSC), detects Skype fingerprint from the packet framing structure, by using a measure of the degree of randomness of the first bytes of the packets payload; the measure is derived from Pearson's Chi Square test and is agnostic to VoIP-related traffic characteristics.

Based on the above classification framework, we setup an interactive demonstrator, described in Sec. 3, that aims at assessing two complementary aspects on the life of Skype peers. First, we illustrate the classification mechanism. The output (i) of the Skype packet generation process and (ii) of the indexes computed by the classifiers are shown and updated during live Skype calls, so that users can appreciate the variability of the traffic Skype generates, and the effectiveness of the classifiers. Complementary information providing insights into the Skype overlay maintenance are shown as well.

Moreover, beside the demonstrator description, this paper focuses on the feasibility of *online* Skype traffic classification, discussing memory requirements and computational complexity: as we will show in Sec. 4, there is no particular hindrance that hampers online Skype classification.

2. SKYPE CLASSIFICATION OVERVIEW

As previously mentioned, the first objective of the demonstrator is to illustrate, in an interactive fashion, the Skype traffic generation and classification processes proposed in [1].

The demonstrator workflow is sketched in Fig. 1, where different subsequent phases are reported from left to right: namely traffic generation, measurement, classification and analysis. Whenever Skype uses UDP as transport level protocol, the client assembles a message by multiplexing several voice/video/chat/data blocks into the payload of a single transport layer segment as sketched in Fig. 1-(a). Encryption is applied to the resulting message to protect the information and to hide protocol details. Since Skype preferred transport layer protocol is UDP, we mostly ignore TCP traffic for the sake of simplicity. A stream of UDP Skype messages is depicted in Fig. 1-(b): from top to bottom, we are interested in the UDP packet payload, message size and inter-packet gap (IPG).

Let us focus on UDP payload first. As UDP offers only a connectionless unreliable service, there is no guarantees that data will be delivered entirely and in-sequence: therefore, a Skype receiver must extract information to detect and deal with possible incorrect cypher stream lining (e.g., due to the loss of a message) directly from the received packet. Thus, an UDP message must contain an application layer header, that cannot be ciphered but rather only

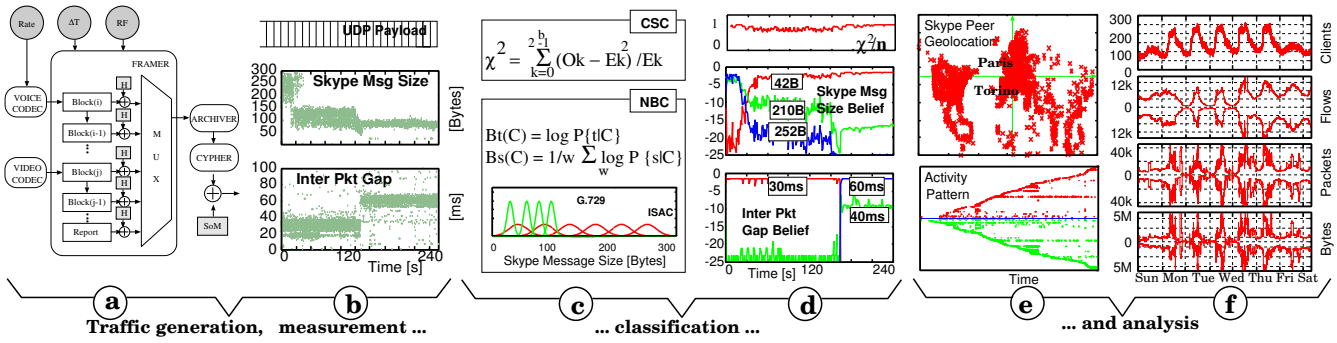

Figure 1: Synopsis of Skype Traffic Generation, Measurement, Classification and Analysis Process

obfuscated [2]. As a consequence, it turns out that a few bits of the Skype message are actually *deterministic*, whereas all the others appear as *random* due to the encryption process: this fact is exploited by the Pearson CSC classifier at the top of Fig. 1-(c), intuitively, the CSC quantifies the amount of randomness in groups of bits of the UDP payload.

Let us now focus on the message size and IPG, shown in bottom part of Fig. 1-(b), which are determined by both the voice/video encoder, and the Skype framing and congestion control algorithms. As shown in bottom plots of Fig. 1-(b), these metrics may vary widely over time during the same call. Bottom part of Fig. 1-(c) sketches a simplified NBC classifier, which computes the "belief" (i.e., the likelihood) that a sequence of messages is a voice stream, by comparing real measurements with an a-priori description of known codecs. This description is given in terms of the Probability Density Function (PDF) of the typical Skype message size and IPG for different "modes" of any given codec. The belief varies over time as well as in Fig. 1-(d), reflecting the different framing and congestion control policies adopted by Skype during the call as seen early in Fig. 1-(b).

The CSC "payload randomness" and the NBC "voice likelihood" information are then (conservatively) combined to eventually classify the current flow as a VoIP flow which has been generated by Skype.

3. THE DEMONSTRATOR SOFTWARE

The demonstrator software is a Python wrapper around the classification framework which is implemented in TSTAT [3, 4], a flow level logger and traffic analyzer originally developed at Politecnico di Torino. The demonstrator software provides a Qt graphical user interface (GUI), where configuration parameters and output results are arranged in different "tabs", as the screenshot of Fig. 2 shows. The source code of the software is available [5], along with a detailed manual of the demonstrator installation and usage.

An important point to stress is that the engine implemented in TSTAT executes the classification algorithms considering *all* traffic flows exposed from the link under examination. The demonstrator software, on the contrary, shows real-time results that are related to an *individual* Skype peer at a time.

As previously stated, the demonstrator software allows real-time interactive analysis of the traffic generated by a single Skype peer. The user can freely select the peer the demonstrator software should focus on, i.e., the PC running Skype. This choice is possible since Skype uses one UDP endpoint for all data transmissions, corresponding to the port number selected during installation (and configurable by the user): any Skype peer can be thus identified by the pair (IP, UDP port). The demonstrator software allows to configure the particular (IP_{probe},$port_{probe}$) endpoint to analyze.

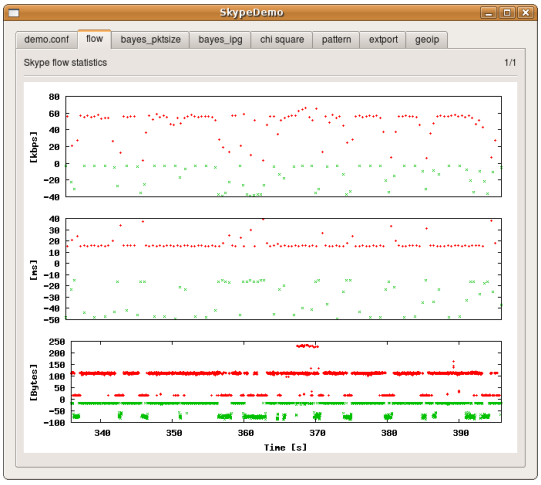

Figure 2: Screenshot of the SkypeDemo software: "flow" tab

The demonstrator software then filters the traffic originated from or destined to (IP_{probe},$port_{probe}$). To pinpoint a particular flow, the engine presents a list of flows generated/terminated to the selected endpoint, from which it is possible to choose the particular flow that should be visualized. Flow selection is done through the first GUI tab, where flows are listed, sorted by decreasing number of packets: the rationale is that as soon as a Skype call starts, the flow corresponding to the actual call can then be easily selected since it will quickly accumulate a lot of packets.

Once a flow has been selected, the demonstrator software starts to visualize its statistical properties (on the "flow" tab), the NBC indexes evolution (on the "bayes_pktsize" and "bayes_ipg" tabs) and CSC index estimation (on the "chi square" tab). For illustration purposes, a screenshot of the "flow" tab is shown in Fig. 2; from top to bottom, it reports the packet size, packet inter-arrival time and throughput evolution over time of the selected flow. Positive values refer to the flow originated from the (IP_{probe},$port_{probe}$) endpoint, while negative values refer to received packets. Since each tab is updated real-time, the demonstrator allows to observe both the packet generation process of Fig. 1-(b) as well as the output of the classifiers of Fig. 1-(c).

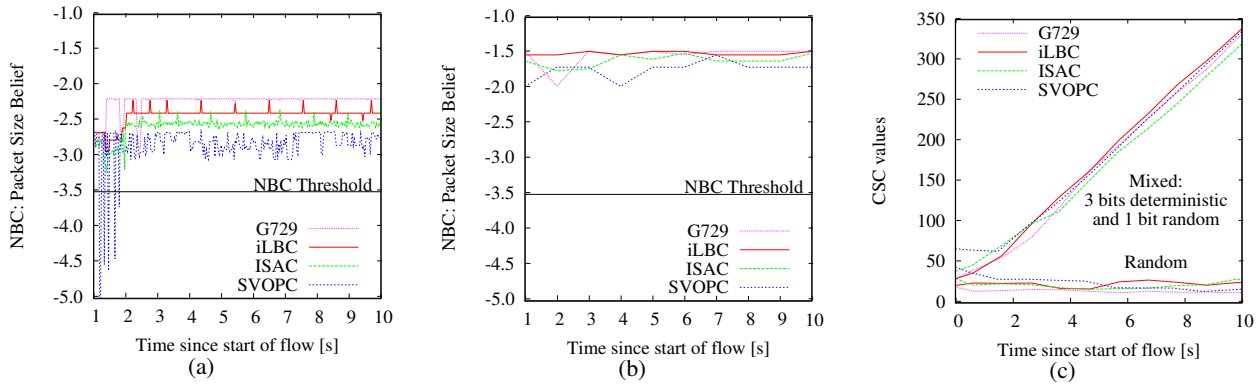

Figure 3: Temporal evolution of: (a) packet size NBC belief, (b) inter-packet gap NBC belief and (c) CSC payload values.

In addition to visualize the Skype voice call properties, the demonstrator aims at showing the activity of the selected Skype peer when *idle*. Indeed, being Skype a P2P application, a significant number of signalling messages is exchanged even if no call is in progress. By observing all the traffic generated by the Skype peer, the demonstrator extends the analysis to other interesting aspects of Skype activity. More on details, the last three tabs of the demonstrator GUI are:

- "pattern": it represents Skype traffic patterns as detailed in [6];
- "extport": it reports the distribution of ports used by Skype peers contacted by the observed peer: since Skype uses random ports, a uniform distribution is expected;
- "geoip": it reports the geographical localization of Skype peers, obtained by resolving in real-time the IP address of any new peer contacted: the histogram of the nationality of the contacted peers is then displayed.

4. ONLINE CLASSIFICATION

In [1], the classification engine is evaluated taking a decision at the flow end, after that the NBC and CSC statistics have been computed for several windows of packets and after that all the transient effects have vanished. In the following, we refer to classification decision taken at the flow end as "late classification". While late classification enables a number of useful tasks (e.g., traffic monitoring, accounting, differential billing, etc.), it precludes other important tasks the network operator may be interested in (e.g., QoS and policy enforcement, traffic filtering, etc.). Therefore, it may be desirable to take the classification decision as soon as possible, possibly after a few packets have been observed – which we will refer to as "early classification".

Though [1] focused on late classification, the proposed methodology has no intrinsic limitation for early classification. In the following, we show that early classification of Skype traffic is possible and the same accuracy of late classification can be achieved. This is an important finding, which not only widen the classification engine scope of application, but has also an impact on classification complexity, as we discuss in the remaining of the paper.

4.1 Accuracy

To assess early versus late classification accuracy, useful insights are offered by using the demonstrator software. Let us consider a few Skype flows that are representative of different codecs (ISAC, iLBC, SVOPC, G.729). We examine the time-evolution of the NBC and CSC statistics depicted in Fig. 3. More on details, the figure reports from left to right respectively: (a) the NBC belief of the packet size, (b) the NBC belief of the average packet interarrival time and (c) the CSC statistics of two different groups of bits of the Skype payload.

Recalling that a successful identification requires the joint agreement of the three classifiers, let us observe Fig. 3-(a). The plot reports the temporal evolution of NBC belief of the packet size, since the flow started, computed for each packet; the horizontal line represents the threshold above which there is a strong belief that packet sizes are generated employing a given Skype Codec. We recall that the NBC scale is logarithmic so that the higher the NBC values are, the stronger the likelihood of being a Skype flow is. As it can be expected, the NBC belief is higher for Codecs employing fixed size blocks, like the G.729 and iLBC codecs. Most important though, the NBC belief converges very quickly for all Codecs and all NBC belief are above the threshold except for the first few seconds of the SVOPC codec flow. Therefore, as far as the packet size NBC is concerned, a few seconds are sufficient in order to successfully identify any given Codec.

Similarly, Fig. 3-(b) reports the temporal evolution of NBC belief evaluated considering the average inter-packet gap. A window of N=30 packets is used to evaluate the average IPG, so that fewer samples are available. Nonetheless, the IPG NBC statistics quickly converge, so that few seconds are enough to correctly classify the flow.

Finally, the Fig. 3-(c) reports the CSC statistics evolution over time. Two different groups of bits of the Skype message payload are considered. The first group is an example of a "mixed" group in which some bits take random values, and some bits are deterministic. In this case, the CSC values grow linearly with the number of observations. Bits in the second group are the typical example of a group of random bits. As it clearly emerges from the example picture, random and deterministic CSC values quickly diverges, so that it is possible to promptly discriminate them. We can conclude that the CSC decision concerning the Skype protocol fingerprint can be taken after few seconds.

While the assessment of how early the classification can be taken without its accuracy being compromised is outside the scope of this paper, we experimentally verified over the dataset considered in [1] that the accuracy of the classification decision is the same when taken after 5 seconds of flow lifetime.

4.2 Complexity

We now focus on the engine run-time cost, considering both (i) *memory* and (ii) *computational resources* requirements.

For memory requirement, for each tracked flow, each NBC classifier requires only two `double` floating point variables used to accumulate the packet size and inter-packet-gap beliefs. Indeed, NBC belief values are updated on an online fashion, and thus no further state variable is needed. CSC classifiers instead need, for every flows, 2^b counters for each of the G groups of payload under observation (with $b = 4$ and $G = 16$). The size of the counter actually depends on how early the decision is taken: indeed, a `byte` counter is sufficient in the case of early classification (i.e., whenever the decision is taken over the first 255 packets), whereas a `word` counter is more apt for late classification. Let us roughly quantify the number of flows that can be tracked at the same time when 1 GBytes of RAM is devoted to store the classification state variables. From the above description, 272 Bytes are needed per flow in the case of early classification. It follows that about 4 millions of flows can be tracked at the same time. Late classification requires 578 Bytes per-flow, i.e., still more than 2 millions of flows can be classified contemporarily. Thus, memory requirement is not an issue considering today typical PC configuration.

Concerning computational requirement, for each sample, the NBC classifier requires one logarithm and one sum operation to update the belief value. The operations are performed either on every packet of tracked flows (for the packet size NBC) or once every window of $N = 30$ packets (for the average IPG NBC). Conversely, the CSC requires G counters to be incremented at each packet arrival, whereas the computation of the CSC statistics needs $G \cdot 2^b$ multiplications when the classification decision is taken, i.e., once per each tracked flow.

While at first sight the above operations may seem to pose a significant burden in terms of CPU requirement, our benchmark of the classification engine actually shows that the computational overhead is instead very limited.

To quantify this overhead, we consider two traces collected in an edge (CAMPUS) and a core (GARR) network. The CAMPUS dataset has been collected from the edge router that connects the Politecnico di Torino campus network to the Internet, while the GARR dataset has been collected on a OC-48 link that connects two backbone nodes in the GARR network. Details can be collected from [7] considering the GARR dataset whereas CAMPUS is detailed in [1, 6]. We compare the performance of TSTAT when the classification engine is turned on or off. Experiments are run on a high-end PC featuring an Intel Quad-Core Xeon X5355 processor clocked at 2.80GHz with 4096kB of cache, equipped with 4 GBytes of RAM memory. Benchmark results are reported in Tab. 1, for both datasets, in terms of TSTAT's packet processing rate (in Kpps, averaged over 10 runs on the same dataset). Several cases are reported: the first row reports TSTAT baseline performance[1], which is obtained when the classification engine is turned off; second row reports results considering the CSC engine only, while third row reports results considering both the CSC and the NBC engines enabled. Finally, the last row refers to the case in which both the CSC and the NBC are enabled, but early classification case is considered, i.e., for each flow, both classifiers run up to when the classification decision is taken after 5 seconds since the flow start. The table also reports the relative (in percentage) overhead, express using pure TSTAT performance as baseline performance.

[1]Interestingly, advances in off-the-shelf PCs technology directly entail a noticeable improvement of baseline performance: indeed, considering the same dataset, baseline packet processing rate increases of more than 30% with respect to 2006 results [7], obtained on a 2.40 GHz Dual-Core Intel Xeon with 512 kBytes of cache and 2 GBytes of RAM memory.

Table 1: Cost Analysis of Online Skype Traffic Classification

	Processing Rate [Kpps]		Overhead [%]	
Engine	GARR	CAMPUS	GARR	CAMPUS
None	334.92	450.78	*0.0*	*0.0*
CSC	329.12	411.63	*1.7*	*8.7*
CSC+NBC	313.76	346.12	*3.6*	*23.2*
CSC+NBC, Early	323.25	401.95	*3.3*	*10.8*

It can be seen that, even when both the classifiers are enabled, and late decision is considered, TSTAT is able to classify about 313-346 thousand packets per second – which considering the typical packet size corresponds to multi-Gigabit rates. Moreover, notice that early classification further increases processing performance to about 323-400 Kpps: therefore, we can conclude that computational performance are not of great concern either.

5. CONCLUSIONS

This paper presents an open-source demonstrator software aiming at assisting and illustrating the characterization and classification of Skype traffic. The software tool allows (i) to examine the temporal evolution of Skype call classification in an interactive fashion, and (ii) to visually depict several properties of the traffic patterns Skype generates, either during a call or while idle.

The demonstrator is build around an open-source classification engine, which we use to testify the feasibility of real-time classification of Skype traffic. Our results show that classification decision is not compromised when it is taken a few seconds after the flow starts, achieving the same accuracy as decision taken at the flow end. Moreover, design considerations concerning memory requirements show that the engine is able to scale up to several millions contemporary flows. Finally, benchmark results testify that the engine is able to cope with multi-Gbps links in real-time, even when using common off-the-shelf hardware.

6. ACKNOWLEDGEMENTS

This work has been funded by the 7th Framework Programme STREP Project "Network Aware Peer-to-Peer Applications under WIse NEtwork" (NAPA-WINE). A real time demonstrator has been exibithed at the ACM Sigmetrics 2008 conference.

7. REFERENCES

[1] D. Bonfiglio, M. Mellia, M. Meo, D. Rossi and P. Tofanelli, "Revealing Skype Traffic: When Randomness Plays with You," *ACM SIGCOMM'07*, Kyoto, Japan, Aug. 2007

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

[3] M.Mellia, R.Lo Cigno, F.Neri, "Measuring IP and TCP behavior on edge nodes with Tstat", *Computer Networks*, Vol. 47, No. 1, pp. 1–21, Jan 2005

[4] TSTAT website, `http://tstat.tlc.polito.it/`

[5] Skype demo, `http://www.enst.fr/~drossi/SkypeDemo`

[6] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca and D. Rossi, "Tracking Down Skype Traffic," *In Proc. of IEEE INFOCOM'08,* Phoenix, USA, Apr. 2008

[7] D. Rossi and M. Mellia, "Real-Time TCP/IP Analysis with Common Hardware," *In Proc. of IEEE ICC'06*, Istanbul, Turkey, Jun. 2006