A Real-Time Algorithm for Skype Traffic Detection and Classification

D. Adami, C. Callegari, S. Giordano, M. Pagano, and T. Pepe

Dept. of Information Engineering, University of Pisa, Italy
{adami,callegari,giordano,pagano,pepe}@netserv.iet.unipi.it

Abstract. In the last years Skype has gained more and more attention from both the users and the scientific community. Namely, the users are interested in its ability to provide a free and reliable way to make phone calls over the Internet, while the scientific community is interested in the reverse-engineering process, because of the proprietary design of the application. In more detail, both Skype protocols and algorithms are unknown and use strong encryption mechanisms, making it very difficult to even reveal Skype presence inside a traffic aggregate. This issue is of primary interest for the scientific community and, above all, of big economical relevance for the operators.

In this paper we propose a novel algorithm for detecting Skype traffic, based on both signature-based and statistical approaches. The proposed algorithm is able to reveal in real time the presence of Skype clients in the monitored network, and to distinguish among the several types of Skype "activities": direct calls, calls with relay node, SkypeOut calls, and file tranfers. To assess the effectiveness of our method we have tested the system over several traffic data sets, collected in different networks. Moreover we have compared the performance offered by our system with those provided by "classical" classification techniques, as well as by the state-of-the-art Skype classifier.

1 Introduction

In recent years, the popularity of VoIP-telephony has progressively grown and the majority of network operators has started offering VoIP-based phone services. Skype [1] has rapidly become the most well-known example of this consolidated phenomenon: originally developed by the entrepreneurs who created the pioneering Web applications Kazaa, Skype ended 2008 with 405 million user accounts (more than 42 million of real users [2]), a 47% increase from 2007 [3]. According to TeleGeography Research [2], in 2008 Skype users spent 33 billion minutes talking to people in other countries, representing 8% of all international voice traffic. Moreover, Skype usage hit an all-time peak on March 30, 2009, when more than 17 million users were online at the same time [4].

Unlike other VoIP applications, which generally rely on the client-server architectural model, Skype operates on a P2P overlay network. The communication among Skype users is established according to the traditional end-to-end paradigm except when a relay node is used for symmetric NATs and firewalls traversal. Skype offers several free services: voice and video communication, file transfer, chats, buddy lists, and SkypeIn/SkypeOut to direct call towards the PSTN.

Due to its tremendously wide spread and success, Skype has recently attracted the attention of both network operators and the research community. More specifically, network operators require the development of efficient techniques for the identification of the traffic generated by Skype to monitor its impact on network performance, design effective security policies, enforce traffic differentiation strategies, and conveniently charge the use of network services.

However, as Skype protocols are proprietary, cryptography, obfuscation, and anti reverse-engineering techniques [5] are extensively adopted and ad-hoc techniques are used to evade NATs and firewalls, the identification and characterization of Skype traffic have been, and still are, hot research topics.

At the time of writing, the most complete work concerning Skype traffic classification is [6], where the authors present a methodology working in real-time. In more detail, they propose a framework based on two different and complementary techniques, for revealing Skype traffic from a traffic aggregate. The first one, based on Pearson's Chi Square test, is used to detect Skype's fingerprints from the packet framing structure, but is agnostic to VoIP-related traffic characteristics. The second approach, instead, relies on a stochastic characterization of Skype in terms of packet arrival rate and packet length, which are used as features of a decision process based on Naive Bayesian Classifiers. In [7], instead, the authors aim at identifying relayed, rather than direct traffic, and Skype is chosen as a case-study to validate their approach.

Other works focus on understanding how Skype protocols work. In [5] the authors provide an overview on Skype design and functionalities, exploring many operational aspects under different network scenarios. Users with public IP addresses as well as users behind a port-restricted NAT or UDP-restricted firewall are taken into account. The work [8] aims at studying Skype operations by means of traffic measurements over a five months period. The authors analyze the user behaviour for relayed, rather than direct, sessions only. Results pertain the population of on-line clients and their usage pattern, the number of super-nodes and bandwidth usage.

Finally, the paper [9] investigates Skype signalling mechanisms by means of passive measurements, providing insights into Skype signalling mechanisms and analyzing the cost and complexity of managing Skype P2P overlay network.

This paper deals with the identification of Skype traffic and, more specifically, proposes a joint signature-based and statistical approach that outperforms state-of-the-art methodologies. Indeed, as reported in the following sections, our algorithm is able to perform a real-time classification of both signalling and data traffic generated by Skype. A further contribution is related to the detailed description of the signalling traffic, generated during the start-up phase, that has not been reported in any previous work.

The paper is organized as follows. Section 2 describes the main characteristics of Skype traffic, focusing on the details that are used to perform the detection. Then Section 3 details the proposed algorithm, while Section 4 presents the experimental results. Finally Section 5 concludes the paper with some final remarks.

2 Skype Traffic

Before detailing the algorithm implemented to detect Skype traffic, we present a brief analysis of Skype features, focusing on those aspects that are relevant to the detection phase.

Despite in this paper, for sake of brevity, we do not provide all the details related to Skype network architecture, it is important to highlight that this knowledge is necessary for a complete understanding of the paper and can be found in [6], where the authors present a deep description of Skype architecture and traffic. In more detail, we assume the reader knows the difference between a Skype Client (SC) and a Skype Supernode (SN), the structure of the Start of Message (SoM), the main characteristics of the voice traffic, and the different types of possible calls (e.g. with or without relay node).

2.1 Skype UDP Ping

The Skype UDP Ping is a message exchange, carried out periodically by all the SCs, which consists of two *keep-alive* messages. These messages are characterized by the function field of the message equal to 0x02. It is worth noticing that some UDP flows only consist of the exchange of such messages.

2.2 Skype UDP Probe

The Skype UDP Probe is a message exchange performed when Skype application is launched, to discover the SNs and the network characteristics (e.g., presence of NATs, firewalls, etc.). The Skype UDP Probe consists of 4 UDP messages (Long Skype UDP Probe): two request messages from the SC to the SN (U1 and U3), and two reply messages from the SN to the SC (U2 and U4). Figure 1 depicts the message exchange (s_x is the size of the payload of packet x).

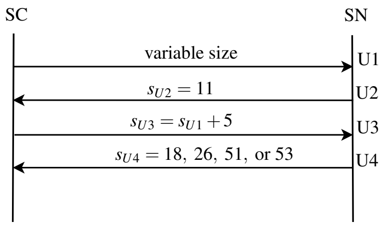

Fig. 1. Long Skype UDP Probe

After the Skype UDP Probe has been executed, the SC performs the algorithm for the SN selection, which consists of the following steps:

```
if $s_{U4} = 18$ bytes then
    the SC opens a TCP connection (on the same port number used for the Skype UDP
    Probe) and the contacted node becomes its SN
else
```

```
    **if** $s_{U4} \in \{26, 51, 53\}$ bytes **then**
        the contacted node will not be taken into account any longer
        # it is likely that these messages can be interpreted as SN "negative" replies
    **end if**
**end if**
```

A modified version of this message exchange, only involving U1 and U4 packets (Short Skype UDP Probe), is repeated until a SN is selected

Moreover, it is worth noticing that the SC periodically repeats this message exchange (Short Skype UDP Probe), so as to be sure to be always connected to an available SN.

2.3 Skype TCP Handshake

Once the SN has been selected, the SC has to establish a TCP connection with the SN, so as to be able to access Skype network. This phase is composed of the following steps:

- a TCP connection is opened towards the first node that has positively answered to the Skype UDP Probe (using the same port number used for the reception of the Skype UDP Probe messages)
- in case the TCP connection fails, another connection is established with another SN that previously acknowledged an Skype UDP Probe
- once the connection has been established, a message exchange, named Skype TCP Handshake, is started

It is worth noticing that, even though the whole payload is encrypted, these packets are characterized by the TCP PSH flag set and, some of them, by a fixed size packet payload. Figure 2 displays the message exchange.

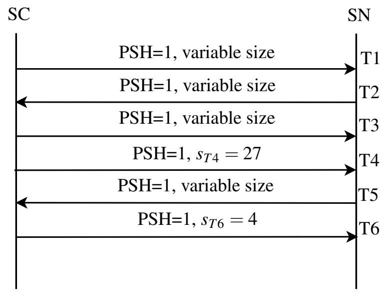

Fig. 2. Skype TCP Handshake

From the figure, we note that six messages (T1, T2, T3, T4, T5, and T6) are exchanged: T1, T2, T3, and T5 are variable size messages, while messages T4 and T6 are 27 and 4 bytes long, respectively.

Since this phase is based on the opening of a TCP connection over an arbitrary chosen port, it is obvious that this packet exchange can only be realized if there are no restriction on the usable ports. On the contrary, if the SC resides behind a NAT or a firewall, the connection is established over TCP port 80 (Skype HTTP Handshake) or 443

(Skype HTTPS Handshake). This is justified by the fact that the firewalls do not usually restrict the usage of such ports, since they are respectively used by the HTTP and HTTPS protocols. It is worth noticing that Skype uses Transport Layer Security (TLS) protocol over port 443 and a proprietary protocol over port 80. The message exchange over both ports is depicted in figure 3.

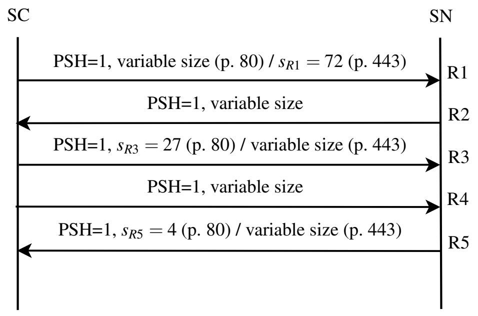
SC
SN
PSH=1, variable size (p. 80) / $s_{R1} = 72$ (p. 443)
R1
PSH=1, variable size
R2
PSH=1, $s_{R3} = 27$ (p. 80) / variable size (p. 443)
R3
PSH=1, variable size
R4
PSH=1, $s_{R5} = 4$ (p. 80) / variable size (p. 443)
R5

Fig. 3. Skype HTTP/HTTPS Handshake

Let us analyze the two cases separately:

- connection over port number 443
  - the payload of the first packet is 72 bytes long and the first 56 bytes have a fixed value (named A_1 in the following), corresponding to the Client Hello of the TLS 1.0 protocol
  - the size of R2 is variable, but the first 79 bytes have a fixed value (named A_2 in the following), corresponding to the TLS Server Hello message

  Moreover, some of the fields of Skype SoM, that should be variable, are instead fixed in these packets (i.e., *gmt_unix_time* field (bytes 12-15) and *random_bytes* field (bytes 16-43))
- connection over port number 80
  - the payload size of the R3 and R5 messages is 27 bytes and 4 bytes, respectively.

2.4 Skype TCP Authentication

Once the connection has been established, the SC needs to authenticate itself to the server. In the TCP authentication phase the SC opens a connection with a Login Server, that is the only centralized server of Skype architecture. Four distinct packets are exchanged during the Skype Login Server Connection phase (L1, L2, L3, and L4), as depicted in figure 4.

The main characteristics of these messages are:

- L1: the payload size is 5 bytes and it contains a fixed value (named A_3 in the following), that is a TLS Server Hello message
- L2: the payload size is 5 bytes and it contains a fixed value (named A_4 in the following)

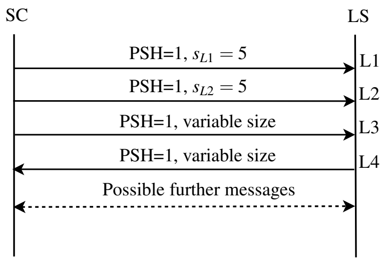

Fig. 4. Skype Login Server Connection

- L3: the first 15 bytes have a fixed value (named A_5 in the following), while in a variable position we find A_4
- L4: the first 4 bytes have fixed value (named A_6 in the following)

To be noted that Skype application allows the user to save some authentication credentials. In that case, this message exchange is not realized when the application is launched.

2.5 Skype HTTP Update

This message exchange is performed every time Skype is launched and it is aimed at retrieving the application updates. It consists of one unencrypted message, which varies depending on Skype version, but presents either a fixed value (named A_7 in the following) in the first 29 bytes (Linux versions) or a fixed value (named A_8 in the following) in the bytes 95-124 (Windows versions).

3 Detection Algorithm

In this section, we describe the algorithm that allows us to perform a real-time classification of Skype traffic, detailing at first the approach used for UDP traffic and then the one used for TCP traffic. Before discussing the algorithm, we list the parameters that are used in the following:

- D_i: payload size of packet i
- F: function field of Skype SoM
- t_i: arrival time of packet i
- $B_{i,j}$: bytes from i to j of the packet payload
- FLOW: UDP or TCP flow, identified by the “classical” 5-tuple
- tentative: preliminary decision
- N_{Ox0d}: number of packets with function field equal to 0x0d
- B_{sd}: number of bytes sent from the source to the destination
- B_{ds}: number of bytes sent from the destination to the source

3.1 UDP Traffic

Given a flow $FLOW$, the procedure aimed at detecting and classifying the signalling traffic over UDP consists of the following steps:

```
if first packet and $F == Ox02$ then
  $dimfirstpkt = D_1$
end if

if second packet and $dimfirstpkt \neq 0$ then
  if $F \neq Ox02$ and $F \neq Ox07$ and $D_2 \notin \{11, 18, 26, 51, 53\}$ bytes then
    $dimfirstpkt = 0$
    $FLOW \neq Short\ Skype\ UDP\ Probe, Long\ Skype\ UDP\ Probe, Skype\ UDP\ Ping$
    EXIT
  end if
  if $F == Ox02$ and $D_2 \in \{18, 26, 51, 53\}$ bytes and $t_2 - t_1 < 10s$ then
    $FLOW = Short\ Skype\ UDP\ Probe$
    record the event into the log file “skype_events.txt”
  else
    if $F == Ox02$ then
      $FLOW = Skype\ UDP\ Ping$
      record the event into the log file “skype_events.txt”
    end if
  end if
end if

if third packet and $dimfirstpkt \neq 0$ then
  if $F \neq Ox03$ and $D_3 \neq dimfirstpkt + 5$ bytes then
    $dimfirstpkt = 0$
    $FLOW \neq Long\ Skype\ UDP\ Probe$
    EXIT
    if $F \neq Ox02$ then
      if $FLOW == Skype\ UDP\ Ping$ then
        $FLOW \neq Skype\ UDP\ Ping$
        record the event into the log file “skype_events.txt”
      else
        $FLOW \neq Skype\ UDP\ Ping$
      end if
    end if
  end if
end if

if fourth packet and $dimfirstpkt \neq 0$ then
  if $F \neq Ox02$ then
    if $FLOW == Skype\ UDP\ Ping$ then
      $FLOW \neq Skype\ UDP\ Ping$
      record the event into the log file “skype_events.txt”
    else
```

```
      $FLOW \neq Skype\ UDP\ Ping$
    **end if**
  **end if**
  **if** $F == Ox02$ and $D_2 \in \{11, 18, 51, 53\}$ bytes **then**
    $FLOW = Long\ Skype\ UDP\ Probe$
    record the event into the log file "skype_events.txt"
  **end if**
**end if**
```

As far as the detection and classification of the data traffic are concerned, the system, given a flow, performs the following operation: for each received packet, if the interarrival time with respect to the previous one is less than 3 seconds, computes the quantities N_{Ox0d}, B_{sd}, and B_{ds}. When N_{Ox0d} exceeds a given threshold (experimental results have shown that a value of 95 packets allows us to obtain good performance with an acceptable detection time), the system computes $ratio = B_{ds}/B_{sd}$. On the basis of the value of $ratio$, the system distinguishes three events:

- direct call if $ratio > 0.5$
- file transfer if $0 < ratio < 0.5$
- call with relay if $ratio = 0$

Moreover, the system is also able to correctly detect SkypeOut calls, by using a similar procedure (that we do not detail here, for sake of brevity).

3.2 TCP Traffic

Given a flow $FLOW$, the procedure aimed at detecting and classifying the signalling traffic over TCP consists of the following steps:

```
**if** first packet **then**
  **if** destination port== 443 and $D_1 > 55$ bytes and $B_{1,56} == A_1$ **then**
    $tentative = Skype\ HTTPS\ Handshake$
  **end if**
  **if** $payload == A_3$ **then**
    $tentative = Skype\ Login\ Server\ Connection$
  **end if**
  **if** destination port== 80 and $D_2 \in \{167, 169, 175\}$ bytes and ($B_{1,29} == A_7$ or
  $B_{95,124} == A_8$) **then**
    $FLOW = Skype\ HTTP\ Update$
    record the event into the log file "skype_events.txt"
  **end if**
**end if**

**if** second packet **then**
  **if** $D_2 > 4$ bytes and $tentative == Skype\ Login\ Server\ Connection$ and $B_{1,5} == A_4$
  **then**
    $tentative = 0$
  **end if**
```

```
    if $D_2 > 78$ bytes and $tentative == Skype\ HTTPS\ Handshake$ and $B_{1,80} \neq A_2$ then
        $FLOW = Skype\ HTTPS\ Handshake$
        record the event into the log file "skype_events.txt"
        # the source is a SN
    end if
end if

if third packet then
    if $D_3 > 4$ bytes and $tentative == Skype\ Login\ Server\ Connection$ and $B_{1,15} \neq A_5$
    then
        $tentative = 0$
    end if
    if $D_3 == 27$ then
        $tentative = Skype\ HTTP\ Handshake$
    end if
end if

if fourth packet then
    if $D_4 > 3$ bytes and $tentative == Skype\ Login\ Server\ Connection$ and $B_{1,4} \neq A_6$
    then
        $FLOW = Skype\ Login\ Server\ Connection$
        record the event into the log file "skype_events.txt"
    end if
    if $D_4 = 27$ bytes then
        $tentative = Skype\ TCP\ Handshake$
    end if
end if

if fifth packet then
    if $D_4 == 4$ bytes and $tentative == Skype\ HTTP\ Handshake$ then
        $FLOW = Skype\ HTTP\ Handshake$
        record the event into the log file "skype_events.txt"
        # the destination is a SN
    end if
end if

if sixth packet then
    if $D == 4$ bytes and $tentative == Skype\ TCP\ Handshake$ then
        $FLOW = Skype\ TCP\ Handshake$
        record the event into the log file "skype_events.txt"
        # the destination is a SN
    end if
end if
```

In the case of TCP traffic, Skype application completely encrypts the packets payload, making it impossible to reveal a call by means of any signature. Our method is thus based on a statistical analysis of the traffic. In more detail, we are able to detect

Skype traffic (in this case we cannot distinguish between calls and file tranfers) in two distinct ways: revealing the Skype TCP Ping traffic or the connection tear-down signalling traffic.

In the first case, the idea is to reveal Skype calls based on the fact that during a call, Skype application periodically sends some messages with a payload size of 4 bytes and with an inter-time which is multiple of 1 second. Thus, the system counts the number of 4 bytes long received packets, which respect the following conditions: $t_i - t_{i-1} \in [1,10]$ seconds and the difference between the microseconds part of the two timestamps $\in [-15,15]$. If such number exceeds a given threshold and the mean packet dimension $\in [10,600]$ bytes, the traffic flow is decided to be a Skype flow. Moreover, if the number of packets is greater than 4, the mean packet dimension $\in [100,600]$ bytes, and the total number of bytes belonging to the flow is greater than 40 Kbytes, then the flow is decided to be a call, otherwise it is considered as signalling traffic.

The other approach for detecting a Skype call is based on the detection of the traffic generated by the tear-down phase. In this case, the system counts the number of 19 bytes long received packets, for which $t_i - t_{i-1} < 3$ seconds. If such number exceeds a given threshold and the mean packet dimension $\in [10,600]$ bytes, the traffic flow is decided to be a Skype flow. Moreover, if the number of packets is greater than 6, the mean packet dimension $\in [100,600]$ bytes, and the total number of bytes belonging to the flow is greater than 40 Kbytes, then the flow is decided to be a call, otherwise it is considered as signalling traffic. To be noted, that in this case, the reception of a packet, whose dimension is greater than 150 bytes, resets the counter. This is justified by the fact that we should not observe data packet in the call tear-down phase.

4 Experimental Results

To verify the effectiveness of our algorithm, its performance have been compared with those of standard statistical classifiers and the state-of-the-art Skype classifier originally proposed in [6].

To this aim, we have tested the system both off-line and on-line (real time). Regarding the off-line testing, we have collected around 3 GB of traffic data (in libpcap format), partly over a laboratory LAN in our Dept., and partly over a small LAN connected to the Internet through an ADSL link. Moreover, the traffic traces considered in [6] have also been used. In this way, our data are representative of the types of traffic generated in a research lab as well as by home users. These data have been processed by TStat [10] that isolates the different connections and calculates a set of significant features for each connection. Instead, for the on-line testing, we have installed the system over the gateway of our laboratory LAN.

As far as Skype traffic is concerned, we have considered calls between two end hosts running SCs (including voice calls, chats and file transfers) as well as calls to traditional PSTN phones (SkypeOut calls). For sake of generality, we used Skype SCs running under Linux (versions 1.4 and 2.0) as well as under Windows (versions 3.2, 3.6, and 4.0).

The goal of our algorithm is to correctly identify Skype flows, hence as performance metrics, we have considered the percentage of False Positives (FP) and False Negatives (FN), according to the definitions given in [6]. Since the lengths of the analyzed flows are quite different, these quantities have been estimated on a per-flow and on a per-byte basis, in order to have an insight not only on the number of flows incorrectly classified, but also on the involved amount of traffic.

As far as "general purpose" statistical classifiers are concerned, we implemented the following algorithms:

– Naive Bayes Classifier (NB)
– Linear Discriminant Analysis (LDA)
– k-Nearest Neighbor (k-NN)
– Support Vector Machine (SVM)

In order to squeeze the most out of these classifiers, at first we have extracted the best features, according to the Sequential Backward Selection (SBS) algorithm, and then we have normalized them, so as to obtain homogeneous values. The parameters of the classifiers have been tuned during an appropriate training phase.

The results are summarized in tables 1 and 2 for UDP and TCP flows respectively.

Table 1. UDP flows

	Our Algorithm	Skype Classifier	NB	LDA	k-NN	SVM
FN (Flows) %	4.8	97.57	11.69	9.44	1.9	2.01
FN (Bytes) %	0.06	26.32	95.98	95.98	6.86	7.88
FP (Flows) %	0	2.4	11.83	13.74	10.95	14.94
FP (Bytes) %	0	16.85	6.12	7.27	5.33	17.47

Table 2. TCP flows

	Our Algorithm	Skype Classifier	NB	LDA	k-NN	SVM
FN (Flows) %	27.46	84.95	28.41	20.04	24.59	24.44
FN (Bytes) %	0.64	56.38	33.07	14.45	27.01	39.48
FP (Flows) %	0.01	9.68	4.49	6.95	5.08	3.37
FP (Bytes) %	0.001	94.58	7.27	2.48	0.56	0.11

The previous results highlight that our algorithm outperforms both the general-purpose statistical approaches and Skype classifier [6] since it also employs the knowledge of some Skype "signatures". The high value of FN at flow level for TCP traffic is due to the fact that up-to-date the features characterizing such exchanges of data between SCs have not been identified yet. Nevertheless it is important to highlight that the unclassified flows correspond to a very low quantity of traffic (bytes), which means that the system is not classifying the signalling traffic, while the calls are correctly classified.

5 Conclusions

In this paper we have proposed a real-time algorithm to detect and classify Skype traffic. In more detail, the presented method, by means of both signature based and statitical procedures, is able to correctly reveal and classify the signalling traffic as well as the data traffic (calls and file transfers).

The performance analysis has shown that our algorithm achieves very good results over several types of traffic traces, representative of different access networks. Moreover we have shown that our system outperforms the "classical" statistical traffic classifiers as well as state-of-the-art *ad-hoc* Skype classifier.

Acknowledgment

The authors would like to thank Federico Gentile and Fabio Strazzera for their contribution in support of the presented work.

This work has been partially sponsored by the European Project FP7-ICT PRISM, contract number 215350.

References

1. Skype web site, `http://www.skype.com` (accessed on 2009/04/10)
2. Telegeography web site, `http://www.telegeography.com/` (accessed on 2009/04/10)
3. Skype by the numbers web site, `http://apple20.blogs.fortune.cnn.com/2009/03/31/skype-by-the-numbers/` (accessed on 2009/04/10)
4. Skype users online now web site, `http://idisk.mac.com/hhbv-Public/OnlineNow.htm` (accessed on 2009/04/10)
5. Baset, S.A., Schulzrinne, H.G.: An analysis of the skype peer-to-peer internet telephony protocol. In: INFOCOM 2006. 25th IEEE International Conference on Computer Communications, pp. 1–11 (2006)
6. Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., Tofanelli, P.: Revealing skype traffic: when randomness plays with you. SIGCOMM Comput. Commun. Rev. 37(4), 37–48 (2007)
7. Suh, K., Figueiredo, D.R., Kurose, J., Towsley, D.: Characterizing and detecting skype-relayed traffic. In: Proceedings of IEEE INFOCOM 2006 (2006)
8. Guha, S., Daswani, N., Jain, R.: An experimental study of the skype peer-to-peer voip system. In: IPTPS 2006: The 5th International Workshop on Peer-to-Peer Systems, Microsoft Research (2006)
9. Rossi, D., Mellia, M., Meo, M.: Understanding skype signaling. Comput. Netw. 53(2), 130–140 (2009)
10. Tstat - tcp statistic and analysis tool web site, `http://tstat.tlc.polito.it/index.shtml` (accessed on 2009/04/10)