Identifying Skype Traffic in a Large-Scale Flow Data Repository

Brian Trammell[1], Elisa Boschi[1], Gregorio Procissi[2], Christian Callegari[2], Peter Dorfinger[3], and Dominik Schatzmann[1]

[1] ETH Zürich, Zürich, Switzerland
[2] Università di Pisa, Pisa, Italy
[3] Salzburg Research, Salzburg, Austria

Abstract. We present a novel method for identifying Skype clients and supernodes on a network using only flow data, based upon the detection of certain Skype control traffic. Flow-level identification allows long-term retrospective studies of Skype traffic as well as studies of Skype traffic on much larger scale networks than existing packet-based approaches. We use this method to identify Skype hosts and connection events to the network in a historical flow data set containing 182 full days of data over the six years from 2004 to 2009, in order to explore the evolution of the Skype network in general and a large observed portion thereof in particular. This represents, to the best of our knowledge, the first long-term retrospective analysis of the behavior of the Skype network based solely on flow data, and the first successful application of a Skype detection algorithm to flow data collected from a production network.

1 Introduction

In the last few years the Internet conferencing and instant messaging application Skype has become a key method of communication among users on the Internet. The proprietary nature of its algorithms and protocols and its extensive use of encryption make Skype traffic identification a challenging task. Traffic identification serves two broad purposes: the identification of nodes using Skype, and long-term retrospective analysis of the network-level behavior of a useful application whose protocol specifications are unpublished.

Unlike other VoIP applications, Skype uses a peer-to-peer overlay network for communication among Skype clients, and uses "supernodes" for message relaying and handling metadata such as user profile and presence information. Skype clients that are accessible from the open Internet (i.e., not NATted or firewalled) with adequate bandwidth and resources may be promoted to supernode status. Thus, Skype does not own most of the infrastructure of its network, itself providing only a relatively small set of "bootstrap" supernodes, gateways to the public switched telephone network, and a login server for identity management. Bootstrap supernodes are contacted only when a client is newly installed, and even the login server does not need to be contacted at each login, as clients may

J. Domingo-Pascual, Y. Shavitt, and S. Uhlig (Eds.): TMA 2011, LNCS 6613, pp. 72–85, 2011.

use a supernode to relay login information. This architecture has allowed Skype to rapidly grow its network while minimizing its need to build out infrastructure.

Research on Skype has to date focused on techniques working with packet-level data, including payload. A *flow*, however, typically represents a set of packets sharing the same IP addresses, ports, and protocol; or one side of an exchange between two IP sockets. Flow data contains only this flow key, timestamps, and byte and packet counters. This represents a significant data reduction over packet traces for the same traffic. For many large-scale networks flow data is all that is available, or practical to collect.

In this paper, we build on existing work in packet-level Skype traffic classification to develop a novel method for determining the presence of Skype clients and supernodes on a network using only flow data, based upon the detection of certain Skype control traffic. We then apply this approach to the historical study of Skype traffic over six years in an existing flow data set collected from a medium-sized national-scale network. The use of flow data enables an examination of a much larger observed portion of the Skype network than in previous works.

The remainder of this paper is organized as follows: we first review related works in section 2. In section 3 we elaborate on the specific features of Skype signaling traffic that lend themselves to flow-based analysis, then provide details of the algorithm which enables that analysis. Section 4 evaluates our approach, against a proxy identification method for Skype traffic in flow data, active detection of Skype nodes, and an existing packet-based Skype detector. We then apply our algorithm in Section 5 to the examination of long-term trends in a very large network dataset covering the six years from 2004 to 2009.

2 Related Work

Research on Skype traffic classification and characterization has been an area of interest for some time, but has to date been focused largely on packet-level traffic measurements. Initial work focused on reverse engineering. In [3], the authors give a detailed overview of Skype architecture and functionality; and in [4] the authors examine the application from a network administrator's viewpoint. Control traffic classification at the packet level is detailed in [8] over a five-month period on a small network. Full classification follows in [6], which presents a real-time framework for Skype traffic detection based on two complementary techniques. The first technique is based on pattern recognition, looking for Skype fingerprints in the packet structure. The second leverages packet arrival rate and packet length statistics to feed a decision mechanism based on naive Bayesian classification.

More recently, Rossi *et al.* [13] study Skype signaling mechanisms through passive measurements and provide insights on the complexity of managing the Skype peer–to–peer overlay network. Bonfiglio *et al.* [5] characterize Skype traffic using both passive and active measurements. The authors use their classification to investigate user behavior, as well.

Active approaches are more suited to differential provisioning of Skype services on operational networks. Bremler-Barr *et al.* [7] "harvest" supernodes by preventing connection to them in order to enable Skype traffic filtering. Using both experimental results and an analytic model, the authors show that it is possible to collect enough supernode addresses so as to block the service for an arbitrary connecting client.

The present work, however, is most directly inspired by Adami *et al.* [1], who present a novel real–time algorithm for Skype traffic detection and classification based on the combination of signature matching and a statistical approach. The algorithm can recognize and classify some specific signaling message exchanges; we use these features as they prove to exhibit easily recognized signatures in flow data as well.

There have been prior attempts at flow-based Skype detection. Angevine and Zincir-Heywood [2] use augmented flow data generated from packet data as an input to an existing machine learning system for the purpose of detecting Skype traffic, However, we note this mechanism requires three particular flow properties, TCP acknowledgment count and minimum and maximum flow packet lengths, which are not supported by commonly deployed flow generators. Therefore, this flow-based method practically requires access to original packet data, and is not applicable to analysis on existing long-term, large-scale flow data repositories.

3 Methodology

Our methodology draws on some of the same features of Skype traffic identified in [1] to IP flow measurements based upon a detailed analysis of that algorithm and the visibility of the artifacts it measures in flow data. Here we will review the subset of features of the protocol we use for flow-level detection.

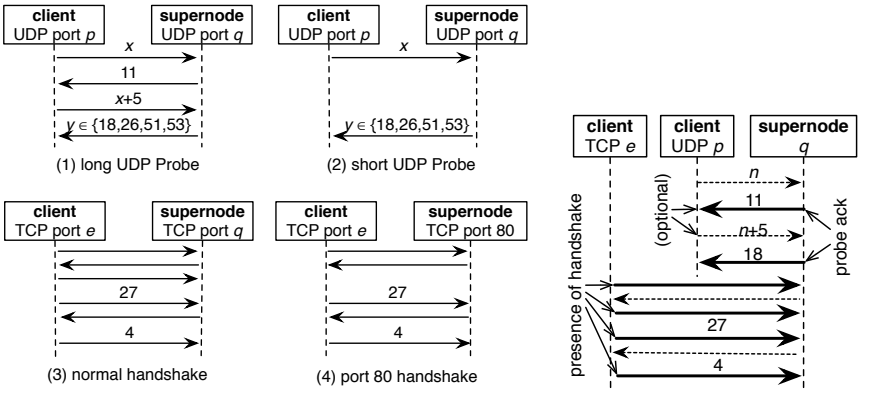
Skype uses a hybrid P2P overlay network of clients and supernodes. Supernodes are specialized client nodes which distribute directory and presence information, and relay messages on behalf of the clients. They can be thought of as the internal nodes of the network, with the clients being the leaf nodes.

Here we focus on two specific message exchanges between clients and supernodes which we call the *UDP probe* and *TCP handshake*.

3.1 UDP Probe and TCP Handshake

The UDP probe consists of a set of messages exchanged to discover a supernode with which to connect to the network, and the characteristics of the connection between the client and supernode (e.g., the presence of NATs, firewalls, etc.). The Skype UDP probe has two forms, a long probe shown in figure 1(a)(1) consisting of two packets in each direction, and a short probe shown in figure 1(a)(2) consisting of one packet in each direction. Either a long probe or a short probe may be seen in connection initiation.

After the UDP probe has been completed, the client initiates a TCP handshake if the size of the UDP payload of the last packet $y = 18$ bytes; we interpret



(a) Probe and Handshake (b) Flow-level visible features

Fig. 1. Packets in Skype connection interactions. Numbers represent packet payload sizes in bytes, which are variable when not shown. Highlighted packets indicate flows used by our approach.

these messages as supernode acknowledgements. If, however, $y \in \{26, 51, 53\}$ bytes, the supernode will not be further contacted by the client in this session; we interpret these messages as negative acknowledgments. The UDP probe is repeated until a supernode is selected. Furthermore, the client periodically repeats the short probe, to ensure it always has an available supernode.

The fact that each packet sent from the supernode back to the client during a UDP probe has a known payload size makes this message exchange useful for detection at the flow level.

Concurrently with UDP probes at network connection time, the client will attempt TCP handshakes with any supernode that positively acknowledges the probe to TCP port q, which is the same port number to which the UDP probe was sent. The TCP handshake exchange is shown in figure 1(a)(3). If the TCP handshake connection cannot be opened, the client will instead use port 80 as shown in figure 1(a)(4), or fall back to port 443. These ports are selected as they are less likely to be blocked; we note specifically that the Port 80 exchange does not use HTTP.

3.2 Flow Level Detection

We illustrate the features of this exchange that are useful for flow-level Skype detection in figure 1(b). Both the short and long UDP probes have easily recognizable signatures in flow data, and are followed relatively rapidly by the handshake, which is attempted on one of three predictable TCP ports $q, 80, 443$. Specifically, the downstream side of a long probe consists of a 2-packet UDP flow with 85 total bytes, while the downstream side of a short probe consists of a 1-packet UDP flow with 46 total bytes. We derive these flow sizes by adding the size of the IP and UDP header on each packet to the sizes of the packet

payloads in the UDP probe.[1] Simply searching for this pattern in flow traffic as shown in algorithm 1 is then sufficient to recognize Skype supernodes and clients given a traffic stream with a high degree of fidelity.

Algorithm 1. Recognition of supernodes and clients given a set of flows

```
acklist ⇐ ∅, supernodes ⇐ ∅, clients ⇐ ∅
for all f ∈ flowstream do
  if protocol(f) = UDP and port_dest(f) ≥ 1024 then
    if ⟨packets(f), bytes(f)⟩ ∈ {⟨1, 46⟩, ⟨2, 85⟩} then
      sn ⇐ address_source(f), q ⇐ port_source(f), cl ⇐ address_dest(f)
      acklist ⇐ acklist ∪ ⟨sn, {q, 80, 443}, cl⟩
    end if
  else if protocol(f) = TCP and
          packets(f) ≥ 3 and
          PSH ∈ flags(f) (if flags present) then
    sn ⇐ address_dest(f), q ⇐ port_dest(f), cl ⇐ address_source(f)
    if ⟨sn, sp, cl⟩ ∈ acklist then
      supernodes ⇐ supernodes ∪ sn
      clients ⇐ clients ∪ cl
    end if
  end if
end for
```

In this approach we take two empirically-determined measures to reduce false positives. First, we reject UDP traffic on well-known ports, as Skype does not in the general case use ports lower than 1024 for supernode acknowledgement, and because the pattern detected by this algorithm on port 53 is consistent with a TCP answer to a UDP DNS query. Second, the *acklist* in algorithm 1 must also be periodically expired; in our implementation, all *acklist* entries are guaranteed a minimum lifetime of one second, as experimentation showed that the vast majority of successful acks are answered by handshakes within one second.

Algorithm 1 recognizes client connection events at the flow level, enabling the study of Skype traffic in data sets containing only flows[2]. It does not attempt general traffic classification. It specifically ignores the data plane, or signaling other than that at connection time.

The algorithm does have a few important limitations. Relying as it does on UDP and TCP, it is incapable of identifying Skype client connections from networks where UDP is blocked or disabled at the client [14], and the client uses an alternate handshake method; this, we suspect, is a significant component of the

[1] The presence of IP options may increase the size of each of these flows in 4-byte increments; however, in our initial evaluation of this algorithm on several days of flow data, enabling IP option detection had no impact on the results; all handshakes were seen after 46- or 85-byte flows.

[2] Note that the examined data set is missing TCP flag data, so our evaluations do not make use of the PSH flag check in this algorithm. Including TCP flag data may increase detection fidelity, but we have not evaluated this case.

real false negative rate, as we elaborate in section 4. It relies as well either on flow traffic captured in both directions, or additional TCP flag information in the supernode-to-client direction that would allow the identification of a successful client-initiated TCP handshake flow. It is not adaptable to sampled flow data, or flow data assembled from sampled packets, because it observes multi-flow interactions and requires specific flow packet count and size data.

The development of any detection algorithm must be concerned with the possibility of evasion. In this case, the interactions we detect are fairly basic to the operation of Skype protocol, so the properties of these interactions themselves would need to be redefined in order to evade this detector. How easy this would be, in a backward-compatible and realistically deployable way, is unknown. However, this possibility should be taken into account in any effort to adapt this algorithm for online Skype client detection, for example, for differentiated services purposes.

4 Evaluation

We implemented this algorithm in a detector, which we dubbed `snack` because it tracks supernode (SN) positive acknowledgments (ACK) of UDP probes. We then set about measuring its accuracy and performance. First, we ran a test protocol containing two distinct clients, one of which starts four times, amid web surfing, secure shell, and VPN background traffic; both clients and all five connection events were correctly identified in each of three different trials, all within twenty seconds of the start of the Skype application, without any false positives.

We then sought to evaluate our algorithm against data collected from larger networks. Here we have a problem with selecting a good proxy for ground truth. We use three different evaluation methods, each with its own advantages and disadvantages: passive comparison against traffic to a host known to be used by the Skype client, verification using active Skype service discovery, and comparison to an existing packet-based detector. In all three of these, we focus on the ability of `snack` to detect clients. The summary of these results is shown in Table 1.

4.1 About the Observed Network

The flow data used in evaluation and exploration in this work comes from a flow data set from the border of SWITCH [15], the Swiss national research and education network. SWITCH operates a production network providing connectivity to the Internet for universities and research laboratories across Switzerland.

This network advertises prefixes for about 2.31 million IPv4 addresses, and the typical daily traffic volume is between 50 and 100 terabytes. The data set is made up of hourly data files containing on the order of 200 megabytes to two gigabytes of compressed flow data per hour. We studied one small portion of this dataset representing four full days in February 2009 for the evaluation against

Table 1. Results of snack evaluation on the measured network in sections 4.2, and 4.3, and 4.4

update server eval (4.2)	*clients*	*updaters*	$c \cap u$	FP_{max}	FN_{max}
Su 15 Feb 2009 (24h)	1846	1811	1224	34%	32%
Mo 16 Feb 2009 (24h)	7913	7116	5648	29%	21%
Tu 17 Feb 2009 (24h)	8087	7457	5798	28%	22%
We 18 Feb 2009 (24h)	7769	7386	5513	29%	25%
mean				30%	25%
union (96h)	13323	11809	9823	26%	17%
active nmap eval (4.3)	*clients*		*verified*	FP_{max}	
Th 6 May 2010 (5.0h)	4008		3618	10%	
packet-based eval (4.4)	$clients_{pkt}$	$clients_{flow}$	$c_p \cap c_f$	$size_{pkt}$	$size_{flow}$
Tu 6 Oct 2009 (7.5h)	17	13	13	11000	41
Tu 13 Oct 2009 (10.3h)	17	15	15	48000	100
Fr 30 Oct 2009 (34.3h)	18	16	16	46000	165

the update server in the section 4.2, one small portion of this data on one day in May 2010 for the evaluation against active probing in the section 4.3, and one larger portion of the dataset representing 182 full days[3] of traffic in both directions across the network border for the entire month of August for each year from 2004 to 2009, inclusive, in section 5 to examine long-term traffic trends in the observed portion of the Skype network.

During the period under observation in the larger dataset, the size of the network was more or less constant in terms of routable IP space, fluctuating less than 9%. It grew from about 2.28 million IPv4 addresses in 2004 to 2.48 million addresses in 2006, falling slowly to 2.31 million addresses by 2009.

The measured network is an access and interconnection network for research institutes and universities without a significant residential population, so it is somewhat biased toward weekday, working-hour traffic. The total flow volume exhibits a characteristic two-peak daily seasonality, with peaks around 07:00 and 12:00 UTC (09:00 and 14:00 local time), corresponding to daily activity cycles of the connected users; we will see this pattern emerge in Skype traffic as well.

4.2 Evaluation against Update Server Activity

Skype clients periodically query an update server[4] to determine whether a newer Skype client is available for download. We assume in this evaluation that this host is used only by Skype clients. We can therefore use the presence of traffic to this server in the data set to evaluate worst-case false positive and false negative rates.

We estimate an upper bound on the false negative client detection rate by counting any host internal to the network contacting the update server within

[3] 2005 excludes four days of data due to a measurement system outage.

[4] As of February 2009, the update server was `ui.skype.com (204.9.163.158)`.

a given day, but not detected as a client, as a false negative. Note that this is only an upper bound for two important reasons. First, there is no temporal correlation between the connection events detected by `snack` and update server contact, and second, `snack` cannot detect internal clients contacting internal supernodes. However, we do estimate that a non-trivial component of this false negative rate is real, corresponding to nonmeasurement because UDP traffic is blocked or disabled at the client.

We can also provide an upper bound on the false positive client detection rate by counting any host inside the network detected as a client, but not contacting the update server within a given day, as a false positive. Similar to the case above, this is only an upper bound, as the Skype application allows the user to disable checking with the update server [14].

Here we consider each day separately and take the mean of the false positive and false negative rates, yielding a mean maximum false positive rate of 30% and a mean maximum false negative rate of 25%. When considering the union of each address set over the four days, these rates go down to 26% and 17% respectively. This reflects the general lack of temporal correlation between connection and update.

4.3 Bounding False Positives: Active Supernode Identification

The widely-used network scanning tool `nmap` supports active Skype detection, as described in [11]. In this evaluation, we took addresses of Skype nodes from `snack` fed these to `nmap` for verification. This evaluation has two important limitations. First, it can only provide a false positive rate (detected by `snack` but not `nmap`). Second, due to delays inherent in the data collection and distribution system, a maximum of four hours pass between the `snack` Skype detection and the `nmap` probe. We presume that supernodes are relatively more stable than the average host, and will have longer uptimes, which should minimize the impact of this delay. We attempt to control for host shutdown by ensuring the host responds to ICMP echo requests (pings), but this method is itself imperfect: first, not every host will respond to pings, or may be behind an ICMP-blocking firewall. Second, this method cannot control for application shutdown or dynamic addressing changes. Therefore, we can only provide an upper bound on false positive rate as in section 4.2.

We ran this evaluation over five hours during the workday on May 6, 2010. We fed a sample of 14149 detected Skype nodes to `nmap`. Of these, 4008 were up at the time of the `nmap` probe, 3618 of which were positively identified as Skype nodes. This translates to a maximum false positive rate of less than 10%, significantly better than that indicated by the evaluation based on update server contact.

4.4 Comparison to Existing Packet-Based Approach

In this subsection, we compare the performance of snack to that of the packet-based detector developed by Adami *et al.* [1]. This comparison was done on

packet data collected during October 2009 from a small network, in this case a single 100Mbit link of a small local cable provider which is not part of the network described in section 4.1. This link provides Internet access to about 30 small businesses and about 70 residential users representing a range of different local network topologies (e.g. NATted, firewalled, and directly connected client machines) and client operating systems. This makes it necessary for Skype to adapt its connection process to the various conditions.

We then generated flows from the packets using YAF [9]. As shown in the *size* solumns in table 1, the original packet traces were between about 200 and 500 times larger than resulting flow files, illustrating the data reduction typical of flow-based processing.

The flow files were then given to `snack`, and the packet files to the Skype detector detailed in [1]. Here, note that if we treat the packet-based detector as ground truth, on this network `snack` has a false positive rate of zero; i.e., every client detected by `snack` is also detected by the packed-based detector. The false negative rate is between 11% and 23%, within the bounds but lower than the maximum false negative rate detected in section 4.2. Manual inspection shows that at least half the difference between the packet-based detector and `snack` can be accounted for due to the inability of `snack` to detect connection events in the absence of UDP traffic during association.

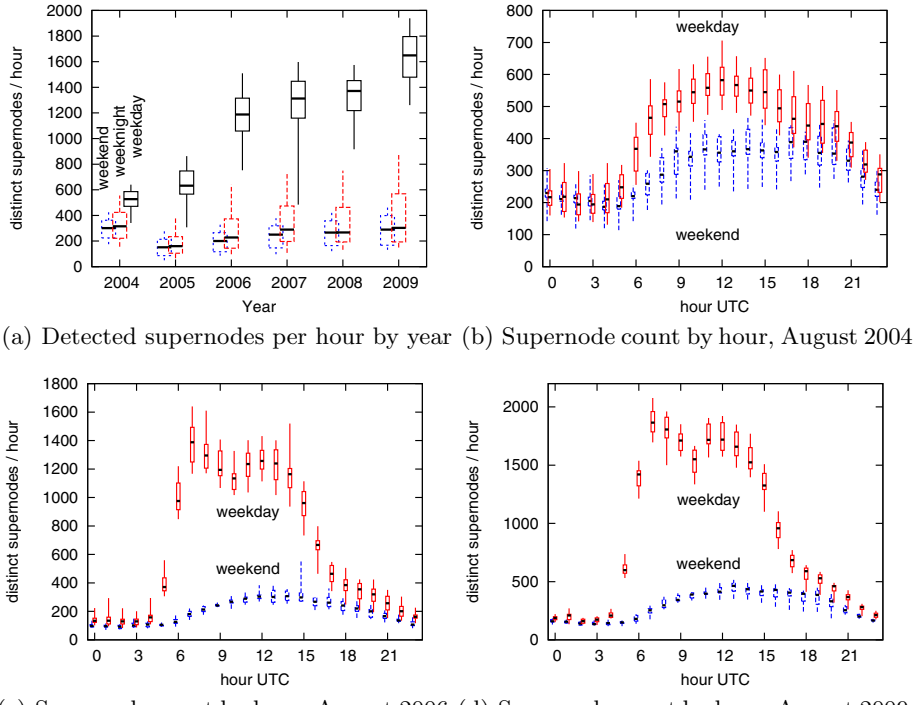
We note that, even factoring in the time required to generate flows from the packet trace, `snack` significantly outperforms the packet-based approach, requiring about seven minutes (434s) to find clients in the first (7.5h on Tuesday 6 October) trace, as opposed to three and a half hours (12423s), for a speedup factor of about 30.

4.5 A Note on Performance

`snack` is quite lightweight, and intended to be integrated into existing large-scale flow processing workflows for retrospective and on-line analysis. The analysis for this study could easily be run in "real time" on a national scale network: as an example, during the run of the study in section 5, `snack` processed 101 gigabytes of compressed flow data in export order[5], covering 10800 minutes (one full week) and containing 230793 connection events, in 670 minutes.

However, the key performance benefit of flow-based analysis is the data reduction achievable with flow data. The data set described in section 4.1 would require on the order of 10 terabytes, 100 terabytes, or one petabyte of storage per month to support analyses on packet header, partial packets (128 bytes of payload) or full packets, respectively. Long-term full packet storage of this magnitude is neither technically nor legally feasible. Indeed, it is the multiple-orders-of-magnitude improvement over packet data in storage efficiency and the reduced privacy impact that makes large-scale network studies and operational measurement such as this one practical, and the primary reason we sought a flow-based method for Skype detection.

[5] Our flow data set is stored as exported by the router; therefore, performance figures here include the time required to reorder the flows by end time as required by `snack`.



(a) Detected supernodes per hour by year (b) Supernode count by hour, August 2004

(c) Supernode count by hour, August 2006 (d) Supernode count by hour, August 2009

Fig. 2. Growth of the Skype network

5 Insights on the Skype Network

As noted in section 4.1, we ran `snack` over a flow data set from the border of SWITCH. We explored trends over time in the development of the observed portion of the Skype network, using the number of distinct supernodes seen per hour as a proxy measurement for the size of the Skype network.

This measurement shows strong weekly and daily seasonality, which is to be expected given that the number of observed supernodes correlates to human communication activity; therefore, in the remainder of this section we will examine quartiles[6] of hourly data sets as opposed to raw time series.

5.1 Network Size

In figure 2(a), we show the observed supernode counts for weekends, weeknights (16:00-05:59 UTC), and weekdays (06:00-15:59 UTC) in August in each of the measured years[7]. The median observed daytime network size increases rapidly

[6] Quartile plots in this section show the 5th, 25th, 50th, 75th, and 95th percentile measurement of each examined variable.

[7] 2007 data excludes data during the 16 August Skype outage. See [12] for more.

to 2006 as Skype grows more popular, then continues increasing more slowly thereafter, with another smaller peak in 2009.

The increase in the size of the Skype network during the day well outpaces the increase in the size of the SWITCH network mentioned in section 4.1.

In comparison, observed night and weekend network size remain relatively constant. This may derive from the relatively light weekend and overnight traffic on the observed network, but we nevertheless interpret this finding to mean that the observed portion of the network has a minimum base size; as clients disconnect from the network at the end of the day, the Skype network maintains a certain number of supernodes. The network then adapts above this base to cope with client load.

In more detail, from 2004 (in figure 2(b)) to 2006 (in figure 2(c)), the median peak hourly size of the network increases from about 600 to about 1400 distinct supernodes; then to about 1850 through the stabilization phase in 2009 (in figure 2(d)). From 2006, the daily pattern shows two distinct peaks at 07:00 and 12:00 UTC (09:00 and 14:00 UTC+2, Central European Summer Time, which is the local time on the observed network). As noted in section 4.1, this pattern is a characteristic of all traffic on this particular network, and we interpret it to represent two distinct peaks in human activity split by a mid-day break.

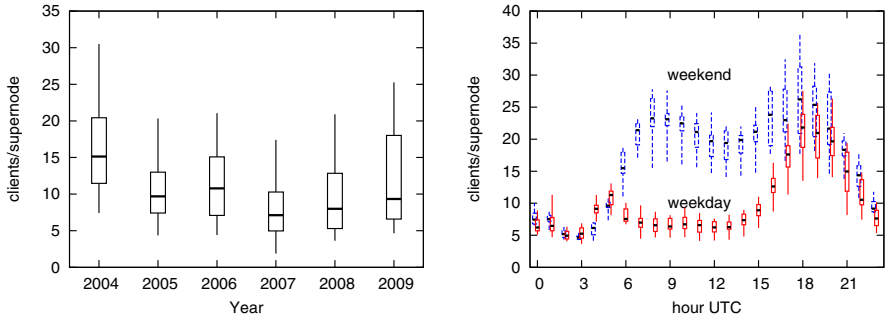
5.2 In-Degree of Supernodes

We then examined the number of distinct clients connecting per supernode per hour, which allows us to estimate the "in-degree" of supernodes. For in-degree measurement, we focus only on supernodes internal to the network[8], as we assume that the fact that the external network is much larger than the internal network implies that the number of clients per supernode is approximately as measured.

Here, in figure 3(a), we see a long-term downward trend from a median in-degree of 15.1 in 2004 to 7.13 in 2007, raising slightly to 9.33 in 2009. Note too that the shape of the distribution changes after 2007, with the 75th and 95th percentile numbers rising from 10.3 to 18.0 and 18.4 to 25.3 respectively. This indicates better balancing of the client load among supernodes throughout the stabilization phase, as there are more higher-degree supernodes in later years than in earlier years.

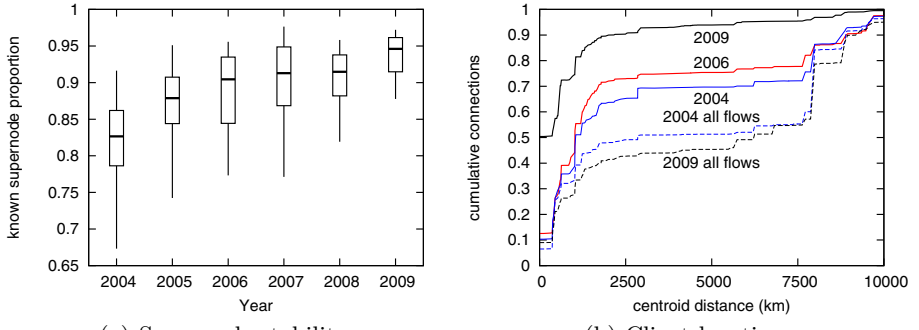
We examine supernode degree in more detail by measuring per hour in August 2009, shown in figure 3(b). Note that at high-traffic times during the working day, the number of supernodes per client is relatively low, with a median between six and seven. As clients and supernodes begin shutting down at the end of the day, external clients must make do with fewer available supernodes; the number of clients per supernode then peaks in the evening, with fewer supernodes (generally on well-connected office networks) serving more home users (generally on less-well-connected residential networks). As with the size of the observed network, weekend and overnight traffic are similar; on weekends, the median number of clients per supernode hovers around twenty.

[8] Approximately one fifth to one third of all supernodes observed, on an hourly basis, are internal to the network.



(a) External clients per supernode per hour (b) Hourly clients per supernode, August 2009

Fig. 3. Supernode in-degree



(a) Supernode stability (b) Client location

Fig. 4. Trends in Skype network maturity

5.3 Network Maturity

In order to further explore the stabilization of the network, we searched for metrics that can be used to estimate the stability of the observed portion of the Skype network. Here we can use the proportion of connections detected by `snack` on a given day involving a supernode that has already been seen that day as a proxy for supernode stability. Here we see a long-term upward trend in figure 4(a). On a median day in 2009, 94.6% of client connections are to a known supernode, as opposed to 82.6% in 2004.

Another proxy for network maturity is its geographic performance. To a first approximation, network distance is related to geographic distance; an overlay network such as Skype is therefore performing well when the traffic of the observed portion of the network is biased toward the locality in which it is measured. In figure 4(b) we show the cumulative geographic distribution of clients external to the observed network, i.e. those which contact supernodes within the network, by the country code associated to IPv4 address by the MaxMind GeoIP database [10], in kilometers distant of each country centroid from the observed

network in 2004, 2006, and 2009. Indeed, here we see a marked increase in locality between 2006 and 2009. Specifically, the proportion of client connections within 2000 kilometers of the observed network, roughly corresponding to the European continent, rises from 63.4% in 2004 to 71.6% in 2006 to 89.6% in 2009. Note also that two large steps in the 2004 and 2006 distributions around 7500 kilometers, corresponding to large numbers of clients in China and the United States, are not present in the 2009 distribution.

We also compare each client distribution with the distribution of external IP addresses for all complete TCP flows on a typical day in 2004 and again in 2009. The proportion of external flows within 2000 kilometers is only 48.2% in 2004, falling to 40.9% in 2009. Skype client connections, therefore, demonstrate considerably better locality than all traffic in general, even as early as 2004, and even as the background traffic becomes more geographically diffuse.

6 Conclusions and Future Work

In this work, we have developed and evaluated `snack`, a flow-level Skype peer detector, by adapting existing work on packet-level reverse engineering of the Skype protocol to flow analysis. This represents, to the best of our knowledge, the first Skype peer detector that operates solely on flow-level data. Significantly, this approach scales to much larger networks than packet-based approaches. We then leveraged this new system to develop insights on the Skype network over a six-year time period from retrospective analysis of a flow-data archive collected from the edge of a national-scale backbone network; this is also the largest such portion of the Skype overlay network to be passively observed in the literature.

We believe the general approach we used in this work demonstrates the impact that packet-level reverse engineering efforts can have on flow-level network traffic analysis, as well as the ability of flow-level analysis to extend the scalability and applicability of features observed at the packet level.

Obvious future directions for the algorithm developed in this work include its application to other similar large-scale flow data repositories. In applications where simply knowing a given address is a Skype client or supernode at a given point in time is enough, the algorithm can be applied directly for operational Skype traffic detection, extending the ability to detect Skype peers to those networks with only flow data available. In cases where more detailed Skype traffic characterization is necessary, the identification of peers provided by our algorithm can be used as a "hint" to a second stage analysis; building such a multistage traffic detector is another area for future work.

Acknowledgments

The authors thank the FP7 PRISM and DEMONS projects for their support of this work. We would like to acknowledge SWITCH, the Swiss Research and Education Network, for providing the data used in this study. Thanks to thank Teresa Pepe and Arno Wagner for their feedback and assistance with evaluation of the approach in this paper.

References

1. Adami, D., Callegari, C., Giordano, S., Pagano, M., Pepe, T.: A real-time algorithm for skype traffic detection and classification. In: Balandin, S., Moltchanov, D., Koucheryavy, Y. (eds.) ruSMART 2009. LNCS, vol. 5764, pp. 168–179. Springer, Heidelberg (2009)
2. Angevine, D., Zincir-Heywood, N.: A preliminary investigation of Skype traffic classification using a minimalist feature set. In: ARES 2008: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security, pp. 1075–1079. IEEE Computer Society, Washington, DC (2008)
3. 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 (April 2006)
4. Biondi, P., Desclaux, F.: Silver needle in the Skype. In: Black Hat Europe 2006 (March 2006)
5. Bonfiglio, D., Mellia, M., Meo, M., Rossi, D.: Detailed analysis of Skype traffic. IEEE Transaction on Multimedia 11(1), 117–127 (2009)
6. Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., Tofanelli, P.: Revealing Skype traffic: when randomness plays with you. SIGCOMM Computer Communications Review 37(4), 37–48 (2007)
7. Bremler-Barr, A., Dekel, O., Levy, H.: Harvesting Skype super-nodes. In: OWASP (December 2007)
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. Inacio, C., Trammell, B.: YAF – Yet Another Flowmeter. In: 24th USENIX Large Installation System Administration Conference, LISA 2010 (November 2010) (to appear)
10. MaxMind. Geoip address location technology, `http://www.maxmind.com/app/ip-location`
11. nmap.org. Version detection using nse, `http://nmap.org/book/nse-vscan.html`
12. Rossi, D., Mellia, M., Meo, M.: Evidences behind Skype outage. In: Proc. IEEE International Conference on Communications 2009 (June 2009)
13. Rossi, D., Mellia, M., Meo, M.: Understanding Skype signaling. Computer Networks 53(2), 130–140 (2009)
14. Skype. Guide for network administrators, `http://www.skype.com/security/network-admin-guide-version2.2.pdf`
15. SWITCH. The Swiss Education and Research Network, `http://www.switch.ch`