Controlling P2P Applications via Address Harvesting: The Skype Story

Anat Bremler-Barr*, Omer Dekel*, Ran Goldschmidt† and Hanoch Levy‡
*The Interdisciplinary Center, Herzliya, Israel. Email: bremler@idc.ac.il, omer.dekel@gmail.com
† University of Haifa, Israel, Email: ran.goldschmidt@gmail.com
‡ Tel-Aviv University, Tel-Aviv, Israel Email: hanoch@cs.tau.ac.il

***Abstract*—P2P applications have become a dominant force in the Internet, both as an economic factor and as a traffic contributor. A "battle of power" is ongoing between the application providers and the Internet Service Providers (ISPs) on who will control this traffic. This is motivated by both economic incentives and QoS objectives.**

Little is known to the ISPs about the architecture of such applications or about the identity of their sessions; these are hidden by the application providers (assisted by their distributed control structure) making the ISPs' life harder.

We are interested in Skype, as a very popular representative of distributed P2P applications. We explore the possibility of getting control/blocking Skype sessions by harvesting its Super Nodes (SNs), and blocking the network clients from connecting to them. Using experimental results and an analytical model we show that it is possible to collect a large enough number of SNs to block, with a probability higher than 95%. We further use the model to show that our approach is robust against possible strategies that can be adopted by Skype to maximize its resilience to blocking.

The results derived and the vulnerability to SN harvesting, though discussed in the context of Skype, are general and may hold true for other Super Node based P2P systems.

I. Introduction

P2P applications have become a dominant force in the Internet. A very popular representative of distributed P2P applications is Skype, on which we focus in this work. Skype has revolutionized the way we conduct VoIP, IM and video communications over the internet – reliably, simply and for free. Despite its massive popularity, little is known about Skype's inner-workings. It is a closed-source application and, consequently, Skype Ltd. does not disclose its protocols and architecture. Due to the closed nature of the Skype network, the Skype protocol can be viewed by network administrators as a black-box.

In many cases network administrators wish to control, or even filter out, legitimate P2P services, such as Skype. This happens especially in enterprises which are concerned with the leakage of data using the encrypted P2P file transfer of the Skype services. Without being drawn into the legal aspects, an ISP may wish to control the traffic of P2P services, since the traffic they produce consumes the ISP bandwidth. An ISP may also wish to filter or limit the rate of P2P services which compete with similar services offered by the ISP; for example, Skype may compete with the ISP VoIP services.

The network administrators find themselves powerless against the Skype application which can easily bypass the network's main filtering tool - the firewall. Skype port usage is random and also has the ability to use the standard *http* and *https* ports, which are rarely blocked. Moreover, Skype's traffic is encrypted.

Currently, there exist in the market several commercial products which offer the ability to filter Skype communications, and several research papers have been written on the subject [1], [2], [3], [4]. All these methods are limited, since they all depend on a specific traffic signature of Skype. If Skype Ltd. wishes, it may easily alter these signatures to render these methods useless. Moreover, by nature, signature techniques may suffer from a high false-positive rate.

In this paper we propose and study a new approach that one could use for getting control of Skype traffic (Section III). The approach is not based on identifying individual signatures, but rather on the general architecture of the system, and thus can be applied generically to other P2P applications of similar architecture.

The architectural property addressed by our approach is the reliance of the system on an array of *Super Nodes* (SN). These Super-Nodes are the heart of the Skype network, and they maintain an overlay network among themselves. Skype clients (Skype users) are allocated one Super Node (or a small number of Super Nodes) with which to associate and connect to, in order to receive the Skype service. Our methodology is a very simple brute force approach, based on harvesting the SN addresses and on blocking users from accessing those SNs.

Using 71 machines over the course of 4 days, we collected over 107,000 unique SNs. We demonstrate that this process converges and discovers the vast majority of the currently active SNs. We show, using experimental results (Section IV), that our methodology can block a client's connection with a 95% probability of success. Analyzing the discovered SNs we found that 46% of them belong to dynamic networks, and hence the harvesting process is bound to be an ongoing one, learning each time the current IP addresses of the SNs.

This paper reveals basic properties of Skype SNs. We measure the residual life of SNs and show that 34% of the SNs died (stopped being SNs) after the first four days (Section V). Using our measurement we derive an estimate of the number of Skype SNs available in a given minute and show that it is

around 45,000 SNs (Section V-A). As far as we know we are the first to estimate this number, which is difficult to derive due to the dynamic nature of SNs. While this number is high, we argue that a router or a firewall can block an entire list easily since the list contains exact IP addresses and hence exact match filtering techniques (hashing or bloom filters) can be applied.

In addition to experimentation, we construct analytic models of the address harvesting process and of the blocking process. We start by forming a probabilistic model under the assumption that all SN addresses are *static* (Section VI). We show (Section VIII) that if the addresses are static then a blocking rate of 100% can be achieved.

We then (Section VII) turn to construct a *dynamic SN address* model, which is, naturally, more complicated. Using the *general distribution* of the "life" of an SN we show that a very high blocking rate (above 90%) can be achieved. We further use the model to explore possible strategies (or parameter settings) that can be adopted by Skype to maximize its resilience to blocking. We show that our harvesting approach is robust against those strategies, and hence a more complicated solution needs to be found against this brute force harvesting approach.

II. Skype Overview

Originally, Skype was a free Internet Telephone service. Presently, it offers many other services free of charge including IM, offline messaging, VoIP, a Video, chat service and file transfer. Skype is a P2P overlay-network based upon a partially centralized architecture[5].

There are two main entities in the Skype network, ordinary Skype Clients (SCs) and Super-Nodes (SNs). The SNs are ordinary SC surpassing some set of publicly unknown thresholds (bandwidth, CPU usage, ...). The SC has not been notified of its statues as an SN. The SNs maintain an overlay network among themselves in order to allow the SC to communicate and establish calls. In order to connect to the network the SC must establish a connection to one of the SNs, after connecting to one of the SNs the SC must authenticate itself against one of the Login Servers (LS). The SN relays the login request to the LS, thus allowing the SC to connect to the network. When the SC installs a fresh copy of Skype application it gets a list of hard coded addresses of Skype LTD SNs to which it connects for the first time. After successfully connecting for the first time the SC maintains a list of 200 IPs and Ports of available SNs. This list of SN is updated continuously due to the nature of the P2P networks where users come and leave the network (we analyze the reasons for this in Section V). Using Skype versions 2-2.5 we observe an average 6% of the list is changed per hour. However if the list is removed the SC downloads a new list, where 75% of the SNs in the new list are new.

After connecting to one of the SNs and having been authenticated the SCs can communicate among themselves. In order to communicate the SC queries its associated SN regarding the availability and IP of the destination client. When the source SC gets the answer it prefers to connect directly to the destination SC, however if it is not possible to establish a direct connection between the clients, the communication is routed through the SNs thus effectively bypassing firewalls and proxy-servers.

III. The SN address Harvesting Techniques

Our technique is based on the following simple observation: by compiling a list of SNs to which the SC can connect and blocking access to them - one can block the SC from connecting to the Skype network altogether.

In order to compile a list of SNs we developed two methods for harvesting SNs from the Skype network: 1. *Extracting the SN List* 2. *Monitoring Skype Connections*. The first technique works only if the list is not encrypted, while the second technique works even if the list is encrypted.

We should note that for each SN we collected, both the IP and the port are used as the SN address. This would serve to eliminate false-negatives of our blocking technique, namely, prevent the accidental blocking of another service which uses the same IP address but uses a different port.

A. Extracting the SN List

In this technique the harvesting of the SNs is based on the fact that in Skype versions 2-2.5 each SC holds a list of up to 200 SNs and their connection port in a specific XML file ($\%appdata\%\backslash Skype\backslash shared.xml$). We harvested the Skype network for SNs by repeatedly doing the following on a SC: 1. Extracting the SN address and ports from the XML file; 2. Flushing most of the SN addresses from the list - leaving only specific SNs 3. Restarting the Skype Client and waiting until the list is refreshed with 200 SN addresses. Each such iteration, takes approximately 2-2.5 minutes.

The technique is similar to the work of Guha et. al. [4] but the aim is different. Guha et. al. focuses on collecting the subset of SNs over long periods of time, and hence used only two computers to collect the SNs over a period of four months (therefore, their compiled list cannot be used to block Skype). We focused on harvesting the entire set of active SNs, and hence we used clusters of dozens of computers all harvesting the SNs concurrently.

In order to discover the bootstrap SNs of Skype, which do not necessarily appear in the XML file, we observed the preliminary connection attempts of a newly installed SC. Using netstat and a sniffer application we discovered the addresses of the bootstrap SNs. Most of them had the same specific port. We added these SNs to the list of harvested SN.

B. Monitoring Skype Connections

An alternative method for SN IP harvesting is observing outbound connections (with the aid of the Netstat command or other network monitoring tools). With the command "netstat -b -n -o" the client's computer lists the active connections (IP,port) with an indication of the application that was responsible for opening the connection (in our case indicating that this is a Skype connection). In order to force the Skype

Client to connect to another SN we block the SN with a local or external firewall. The SC then switches to another SN, assuming this SN is not currently available. Note, that due to the nature of P2P, Skype is designed with the assumption that some of the SNs would not be available, since they have left the network - and hence has a built-in fast mechanism to switch from one SN to another. After collecting around 200 SNs, we erased the SN list file - and thus ensure that the client receives a new pool of SNs. This technique works in all versions of Skype, including the newest version, where the SN list is encrypted.

IV. Experimental Results

In this section we show the feasibility of harvesting sufficient active SNs, in order to block Skype with high probability. We focus on the first technique, *Extracting the SN List*. We also did preliminary testing on the second technique, *Monitoring Skype Connections*, and found that it is feasible to collect 30 new SNs per minute. We show in section VIII that collecting at this rate is enough to block Skype with high probability.

In May 2008 we implemented, the first technique, *Extracting the SN List*, by using a cluster that consisted of 77 harvesters, where a *harvester* denotes a computer that collects the SN addresses.

To achieve some geographical diversity 73 of the harvesters were split across two separate sites (located in Israel), and 4 were placed in Zurich. The experiment was conducted over a period of 80 hours (approximately 2700 iterations).

Using our experiments, the harvesters collected over 40 Million SN addresses (IP, port) pairs, and discovered over 107,000 unique SNs. There are only 106,300 unique SN IP addresses on the list. It is apparent that the difference is negligible, and almost all the SNs only use one port. Over the duration of the experiment (80 hours) 2700 iterations of harvesting were conducted. Figure 1 (a) shows the cumulative number of SNs discovered as a function of the number of harvesters used (one can control this number by disregarding the data collected by some of the harvesters). The order of the harvesters was picked randomly, and the result was stable for any order. It is clear that the majority of the SNs, 100,000, were discovered by the first 30 harvesters. The last 41 harvesters discovered a total of roughly 7,000 SNs. This means, that approximately each one of these harvesters discovered new SNs only in 6.6% of the iterations. This is a good indication for the convergence of our process and that it discovered most of the SNs and more harvesters cannot gather significant number of new SNs.

Figure 1 (b) shows the cumulative number of SNs discovered by the collection of the 71 harvesters as a function of the number of iterations. We can see that the harvesters consistently found new SNs at each iteration (though at a decreasing rate). We used the SORBS project Policy Block List[6] to query whether or not the SN IP addresses are dynamic. We found that 46% of the IPs are dynamic. Hence, we conjecture that many of the new SNs represent a previously found SN whose IP address changed.

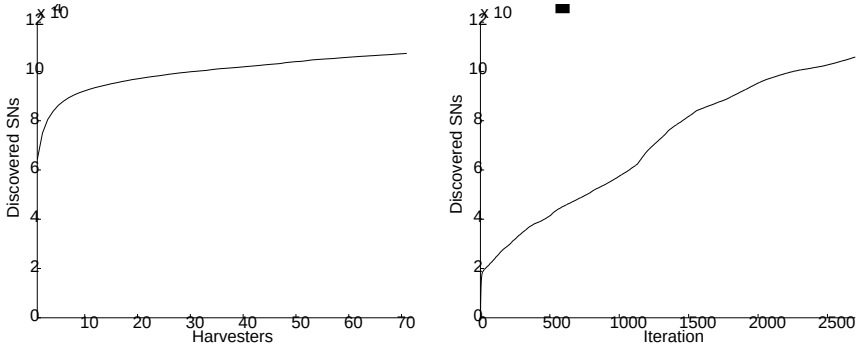
In order to measure the probability of blocking an SC from connecting to Skype from within the Enterprise network, we ran 12 Tester Clients (TC) located in Israel, Switzerland and the USA, that simulate the attempts of a regular user to connect to Skype. Connection attempts from within the enterprise can be classified into 3 types. The first consists of a freshly installed SC. This case is easy to block, since it will attempt to contact only SNs from a hard-coded list, which, as explained, is fixed and thus easy to block. The second consists of an SC which was installed outside of the enterprise (an enterprise user who installed Skype at home on his portable computer) and returns to the enterprise network after a lengthy period (days and even hours). This is also an easy to block option since our experiments found that the harvesters discovered all "old" SN addresses after a short period of time. The third case is the most challenging one to block, and we focus on testing it: It consists of a user who retrieves a very 'fresh' list of SNs. This is a case of a user who installed the SC, connected and used it in some environment that allowed it to connect (home or an internet coffee shop). After connecting and refreshing the SN list, the user returned, after a very short time, to the enterprise network, which now aims at blocking Skype.

In order to simulate this case, our Tester Client performed rounds of 20 minutes, each consisting of the following steps by a TC: Start Skype, log in, stay idle for 10 minutes and shut down for 10 minutes. After each round we checked whether the client can connect to Skype, namely, whether the client's SN list contains an SN which has not yet been discovered by the harvesters.

For each of the 12 TC's we performed 240 such rounds over a period of 4 days. Table I depicts the fraction of the attempts (denoted as blocking probability) that could not successfully connect to the Skype network. This is calculated as the fraction of tests in which all the TC addresses were discovered earlier by the harvesters. We measured this probability of blocking immediately after the completion of the TC round, and 10 and 30 minutes after each round. These different scenarios aim at simulating realistic cases, representing the time it takes the user to return to the enterprise network and attempts connecting to Skype. Our results show that the average probability of blocking is very high, ranging from 94.5% to 95.5%. In other words, we capture 99.95% of all the SNs that appeared in the SN list of 20 TCs. [1]

We analyze the characteristics of the discovered SNs, emphasizing those characteristics which might be useful in designing the blocking mechanism (for more details full paper [7]). In the experiments we collected $2700 * 200 * 77 = 40M$ records. It appears that the 10% of the most frequent SNs cover roughly 80% of the total collected records. We did not identify ports that are used more frequently then others. Also, we did not find a dominant AS. However, we observed that several countries are more dominant than others, The United States covered 20% of all SNs, The United kingdom covered

[1] 95% blocking probability means that for 19 out of 20 TC we harvest all 200 SNs in the SN list. The one TC out of 20 that succeeds to connect the internet, was due to one or two SNs in the SN list that were not harvested.



(a) (b)

Fig. 1. (a) The cumulative number of SNs discovered as a function of the number of harvesters (b) The cumulative number of SNs discovered as a function of the iteration number

	0 minute offset	10 minute offset	30 minute offset
Blocking probability	94.5%	95%	95.5%

TABLE I
THE BLOCKING PROBABILITY 0, 10 AND 30 MINUTES AFTER EACH ROUND

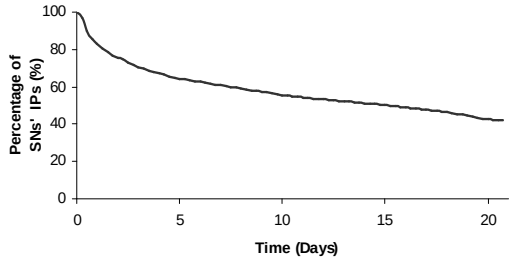


Fig. 2. The percentage of SN IPs with residual life length $> x$

9% of all SNs and the rest of the countries were between 4% to 5% each.

V. THE DYNAMIC NATURE OF SNS

In the next section, we model the Skype network, in order to model the effectiveness of the harvesting mechanism in blocking Skype. In order to do this we need to understand one more key issue about the SNs of Skype - the churn of Skype. The churn in P2P networks is very important and has been extensively studied [8], [9], [10], [11], [12], [4]. It measures the continuous change of nodes joining and leaving the network. The churn of the SNs plays an important role in understanding the blocking probability of our harvesting technique. On the one hand, some of the SNs in the client list may be unavailable at the time the client tries to connect to Skype (and this increases the blocking probability). On the other hand the harvester needs to repeatedly learn the current active SNs - since new SNs may join the network, which may make it difficult for the harvester to learn all the current SNs, and hence reduce the blocking probability.

The churn of Skype is the result of numerous reasons: New Skype clients, that become SNs and the opposite, the case of dropout clients that were SNs and left the network permanently; SNs that temporarily left and then rejoined the network , i.e. closing/opening the Skype client or, more likely, shutting down/opening the computer; SNs with dynamic IP addresses, that change their addresses, and thus appear as new entities in the SN list; The system may choose to abandon old SNs and nominate new ones.

We did not find any research that can give us an exact model of the churn of Skype. Guha et al[4], did extensive research on the churn of the Skype Super-Nodes, but it focussed on the availability of SNs that lived for more then a three month period.

Here we give an estimate of the number of new SNs and the dead SNs over time. An estimate of the number of new SNs that joined Skype per time unit can be learned from Figure 1 (b). At each iteration (2000-25000) on average 9 new SNs were discovered, where iteration as recalled takes approximately 2 minutes. Note, that since we saw that the testers achieved high blocking probability of around 95%, it is clear that most of them are new ones and not SNs that the harvester technique did not succeed in discovering.

In order to understand the number of SNs that died per time unit, we take a sample size of 5026 from the discovered SNs collected in a very short time. We then ping them using the same technique of [4] with an encrypted UDP packet for a period of 21 days. In order to avoid temporary unavailability an SN is counted as dead if it did not, at least, respond in the last three tests of the experiment. Figure 2 shows the residual life of IP address of an SN. Approximately 18% of the SNs died on the first day, 8% on the second day, 5% on the third day and only 3% on the fourth day, a total 34% of the SNs died in the first four days.

We speculate that the residual life of the SNs is less than 4 days for 34% of the SNs, this is related to the fact that a large portion of the SNs belong to dynamic IP networks (usually home users connected by cable, xDSL and so on..). In this case the SN's IP died since the IP address of the SN has been replaced. However, there is a good chance that the SN is alive but with a different IP address.

A. Estimation of Skype active SNs

Using the churn information, we can estimate the number of active SNs per time unit, an important parameter for our model. We assume for simplicity that this size is fixed. We suspect it only gives a good enough approximation, since the active SN population may change due to different loads on the network caused by working hours and holidays.

We estimate the population size by assuming that the number of SNs that died in each iteration is equal to the number of new SNs discovered at each iteration. As we explain, the number of new nodes that were discovered per iteration is around 9 (from Figure 1 (b)) , and the percentage of SN that died in each iteration is 0.000208 (according to the number of dead SNs in the first iteration of our test, where we test a representative sample of Skype SNs close to the time the set was collected). Hence $M \cdot 0.000208 = 9$ where M is the active SNs per time unit. We can estimated M to be approximately 45K.

We note that as a result of the dynamic nature of the list, there is a need for a garbage collector, that eliminates IP addresses that are no longer active from the list of active SNs. This, in order to make sure that the list of SNs will not blow up. We plan to investigate the design of such a garbage collector in future work.

VI. Blocking Skype: Modeling and Analysis of a Static Address Model

Below we provide a simple model aiming at evaluating the effectiveness of the harvesting mechanism in blocking Skype. Our objective is to evaluate the probability of successfully blocking Skype as a function of the number of iterations preformed by the harvesters. This will yield an indication as to the effectiveness and practicality of the methodology. Specifically, if too many iterations are required, it may lead to the infeasibility of the technique, mainly due to the fact that the SN population is dynamic and needs to be learned continuously.

Formally we define M to be the number of SNs (which are distinct objects) in the system. We estimate that M is to the order of $50,000$ (see Section V). Let N be the number of SN addresses in the list of a Skype client. In our case $N = 200$. We consider the following SN selection process: We randomly select (with uniform distribution) N distinct objects (from the set of M objects). We now return the N objects and again randomly select N distinct objects. We repeat this K times. We then conduct this at the $K+1$st time and ask what is the probability that each of the objects selected at the $K+1$st time was selected earlier. This will represent the probability that a real Skype user ($K+1$st) will be blocked as a result of the harvesting conducted by the K selections.

Note, that this is a somewhat oversimplified model of Skype for the following reasons: First, we assume that the SN population of Skype is fixed over time. That is, over time, no SNs are removed or added to the SN set. This assumption will be relaxed in the dynamic model in Section VII. Second, we assume that the selection is done using a uniform distribution. However, from our SN population characteristics analysis this is not precise.

A. Approximate Analysis

Since the exact analysis is provided by a recursion (see full paper [7]), we here provide an approximate closed form analysis that can provide an insight into the results and how they depend on the system parameters.

Consider address (object) x (1 of the 50,000). Let p be the probability that x is selected in iteration i. Since the experiments are independent of each other and of the history, then p is not a function of i. Then we have: $p = Pr[$ *x is selected on* $i] = N/M = 200/50,000 = 0.004$ and $Pr[$ *x is not selected on* $i] = 1 - p$.

Now let $q = Pr[$*x is not selected in experiments 1,...,K*$]$. Since all experiments are independent of each other we have: $q = (1-p)^K = (1 - N/M)^K$. Thus q is the probability that after K experiments x is still an unselected address. Now, we are interested in the probability that a Skype client is blocked after conducting the Kth iteration. Thus, we are interested in conducting the $K+1$st iteration and in asking whether all N addresses of this iteration are already discovered addresses, i.e. blocked address.

Recall that for each of the N addresses received at the $K+1$st reading, the probability that this is a blocked address is given by $1-q$. To conduct the approximate analysis we will assume that for any two addresses $1 \leq i, j \leq N$ the events that these are already discovered addresses are independent of each other. This is not exact since these are dependent events.

Now consider a user who has $N(=200)$ addresses and ask what is the probability that all of them are blocked addresses and hence the user is blocked. Under the assumption of independence we get:

$$\begin{aligned} B &\approx Pr[\textit{all N addresses are blocked}] \\ &= (1-q)^N = (1-(1-N/M)^K)^N \end{aligned} \tag{1}$$

Since the above analysis is not exact - we provide a lower bound on B. To this end note that q is the probability that a single address is free (non blocked).

Let A_i be the event that address i $(i = 1, .., 200)$ is free. Let A be the event that either of the addresses is free. Then we have $Pr[A] = Pr[A_1 \cup A_2 \cup A_3 \cup \ldots A_n]$. Using Boole's inequality we get $Pr[A] = Pr[A_1 \cup A_2 \cup A_3 \cup \ldots A_n] \leq Pr[A_1] + Pr[A_2] + \ldots Pr[A_n] = Nq$. Thus we finally have

$$B = 1 - Pr[A] \geq 1 - Nq. \tag{2}$$

Remark 6.1: Note that the difference between the approximation and the lower bound (Eq. 1 and Eq. 2) is given by

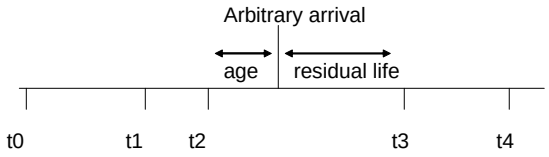


Fig. 3. Illustration of the age and residual life

$\sum_{j=2}^{N}\binom{N}{j}(-q)^j = O(q^2)$. Thus, as $K \to \infty$, $q \to 0$ and the relative difference between the these expressions tends to 0.

In the full paper [7] we demonstrate that the lower bound is tight to the approximation in high blocking probabilities. Moreover, we found out that the exact solution is between the lower bound and the approximation. From this point we will use the approximation equation as our base formula for understanding the model.

Remark 6.2: **(Multiple Harvesters)** If H concurrent independent harvesters are used at each experiment then Eq. 1 changes to: $B \approx (1-q)^N = (1-(1-N/M)^{HK})^N$.

Discussion: Assume the number of SNs is $M = 50,000$ and the number of addresses stored in a Skype client is $N = 200$. Then, using the above analysis one can easily verify that at after conducting $K = 2450$ iterations one reaches a 99% blocking probability both by the approximation and by the lower bound. In practice each iteration by a harvester takes about 2 minutes. If one uses 71 harvesters then one would reach this blocking probability after 70 minutes.

VII. Modeling and Analysis of Dynamic SN Adresses

The analysis so far is based on the assumption that SNs are permanent and have permanent IP addresses. Below we consider the approximate model proposed in Section VI-A and extend it to dynamic addresses.

A. Preliminaries - The Theory of Backward and Forward Recurrence Times

The following definition is taken from [13]: Suppose events occur at times $T_1, T_2, ...$ such that the inter-event times $T_k - T_{k-1}$ are mutually independent, positive random variables with a common cumulative distribution function. Choose an arbitrary time t. The *backward recurrence time* at t is the elapsed time since the most recent occurrence of an event prior to t. The *forward recurrence time* at t is the time from t to the next occurrence. Note that the backward and forward recurrence times are also called the *age* and *residual life* (see, e.g. [14]). See Figure 3 for illustration.

Let L be a random variable denoting the durations $T_k - T_{k-1}$ with a with a common cumulative distribution function, $L(x) = Pr[L \leq x]$ and probability density function $l(x) = dL(x)/dx$. Let L_F, L_B be random variables denoting the forward recurrence time and backward recurrence time associated with L, respectively. It is well known that L_F and L_B have the same distributions whose density function obey (see. e.g.,[14])

$$l_F(x) = l_B(x) = l(x)/(1-L(x)). \tag{3}$$

B. Model and Analysis

We use the following to model the dynamics of IP addresses: An IP address that is given to an Internet entity will remain valid for a period of time that we call "the life of the address". Once this duration is over, the address changes and the new address will be valid for another period. We assume that the durations of all these periods are mutually independent random variables (each denoted L) with a common cumulative distribution function, $L(x) = Pr[L \leq x]$ and density function $l(x) = dL(x)/dx$. A study that provided some estimates of $l(x)$ is [15].

The processes of address harvesting and IP address dynamics interact as follows: At time $t_0 = 1$ the harvesting starts. At that epoch there are M SN addresses, all unknown to the harvesters. The harvesters then operate for a period consisting of K harvesting iterations. We normalize time to the duration of a single harvest, thus harvesting operations start at $t = 1, 2, ...$ and the SN collection iterations occur at times $t = 1, ..., K$. Address changes are assumed to occur at epochs t^+ (namely, just after the harvest starts), and assumed to be relevant to the harvest process only at t+1. For simplicity of presentation it is also convenient to assume that random variable L is given in discrete time units (and so are $L(x)$ and $l(x)$), as well as the corresponding entities for the backward recurrence time L_B.

We are now interested in deriving the number of addresses that are unknown to the harvesters at time K. To this end consider an arbitrary address a whose age (backward recurrence time) at time K is L_B. Then, a is unknown at K if and only if it was not harvested in iterations $K-L_B+1, K-L_B+2, ..., K$. Thus, similarly to the derivation of Eq. 1, the probability q that a is unknown to the harvester at K is given by

$$q = (1-N/M)^{min\{K,L_B\}} \tag{4}$$

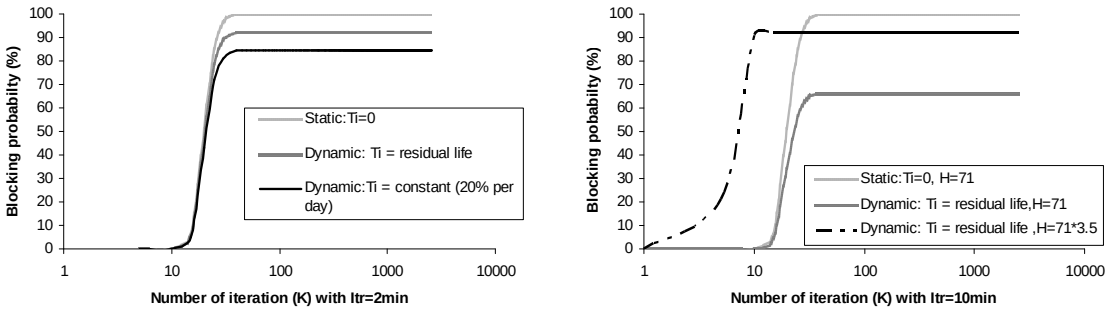
where $min\{K, L_B\}$ is the number of harvesting attempts applied to a after the last time it changed its address. Now, using the distribution of L_B, q is given by:

$$q = \sum_{x=1}^{\infty} l_B(x)(1-N/M)^{min\{K,x\}}. \tag{5}$$

Now, following the analysis of the static addresses and our findings (see Section VI) that the formula in Eq. 1 approximates very well the exact blocking probability, we may *assume* that the blocking events of the different addresses are *mutually independent*. While this is not exact, the analysis in Section VIII suggests that this leads to a very good approximation. Thus, the blocking probability can be approximated, similarly to Eq. 1:

$$B \approx \left(1 - \sum_{x=1}^{\infty} l_B(x)(1-N/M)^{min\{K,x\}}\right)^N. \tag{6}$$

Remark 7.1: **(Multiple Harvesters)** If H independent harvesters are used, then Eq. 6 changes to $B \approx \left(1-\sum_{x=1}^{\infty} l_B(x)(1-N/M)^{Hmin\{K,x\}}\right)^N$.



(a) (b)

Fig. 4. (a) Blocking probability as a function of the iteration number for iteration length of 2 minutes for the different models with parameters N=50K,M=200,H=71 (b) Blocking probability as a function of the iteration number for iteration length of 10 minutes for the different models with parameters N=50K,M=200,H=71

To conclude the dynamic address analysis one needs to estimate the density function $l_B(x)$. Estimating the age of an address might be impractical (hard to measure when an address recently changes in the past). However, estimating $l_F(x)$ is much simpler since it can be done by collecting a set of addresses, and tracking them from the collection epoch until they become unavailable namely "die". Alternatively one can attempt estimating $l(x)$ as done in [15]. One can then use Eq. 3 to compute $l_B(x)$ from $l_F(x)$ or from $l_(x)$. For our analysis, as will be demonstrated in Section VIII, we choose the former.

VIII. Harvesting Effectiveness: Performance Evaluation

In this section we study the effectiveness of address harvesting using the equations derived in the modeling sections. We also explore possible strategies (translated to parameter settings) that can be adopted by Skype to maximize its resilience to blocking. We show that our harvesting approach is robust against these strategies.

A. Effect of Address Dynamics

We start by investigating the blocking probability as a function of the iteration number for three of the following models, assuming M=50K, N=200, H=71 (Figure 4 (a)). First, we consider a static address model, depicted in the upper curve. It is clear that in the static model - the blocking probability can reach the maximum, 100%, quickly, just after 50 iterations. Note that in the static model, where the population is fixed, one can always reach the 100% blocking probability - regardless of the number of harvesters. The number of harvesters only influences the time it takes to reach 100% blocking. In the dynamic models, in contrast, there is a clear upper bound on the maximal blocking probability one can achieve with a specific number of harvesters. This is due to the fact that the population changes with time, hence if there are not enough parallel harvesters one cannot learn short life span SNs.

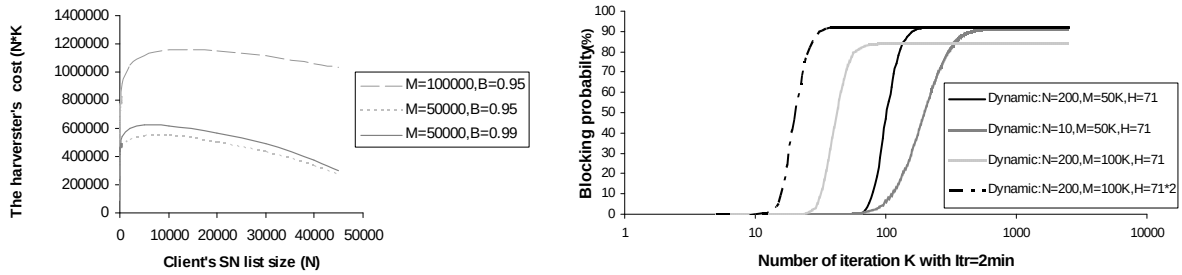
Second, we consider a dynamic address model where the SN residual life is identical to what we recorded in the experimental measurements shown in Figure 2 (18%, 8% , 5% ,3% die in days 1, 2, ,3, 4 ...); Thus it represents a "realistic situation". This is depicted in the middle curve. For this setup the blocking rate achieved is 92%, close to our experimental result of 95%. Third, we examine how the address dynamics rate affects performance, and consider a case where the residual life is distributed as 20% on each of the days 1, 2, 3, 4, 5 (uniform over the day), representing much faster address churn than in the second (realistic case); this is depicted in the lower curve. One can see that the blocking probability is quite sensitive to the address dynamics rate, reaching only 82% in this case.

B. Effect of Skype's "slow down"

In order to reduce its vulnerability to address harvesting, the Skype system can adopt a strategy of releasing addresses to the clients in a "slow mode". This can be very effective when the addresses are dynamic (harvesters do not have enough time to harvest the address). In Figure 4 (b) we depict the blocking probability in a case where the iteration time is 10 minutes (i.e., the harvester receives a new SN list every 10 minutes) as opposed to 2 minutes iteration time in Figure 4 (a). The impact on the maximal blocking probability is significant only for the case of dynamic addresses. One can observe this in the solid-dark curve, depicting the blocking under the experimentally-measured residual life distribution (from Figure 2) and reaching only 60% blocking. However, as can be seen in the dashed curve, one can overcome this Skype's defence by increasing the number of harvesters. Specifically, if the iteration duration is set to be L times longer, then one needs less than L times more harvesters. In the figure we show that if we use $71 \cdot 3.5$ harvesters we can reach the same blocking probability as with the case of 2 minute duration iterations (i.e. blocking probability of 92%).

C. Effect of List Size and Population Size

One might wonder whether Skype can reduce the vulnerability to harvesting by modifying the client list size N. In the static model, if we take the approximation Equation 1 and find K (the number of harvesting iterations needed) as a function of B (the desired blocking probability) we



(a) (b)

Fig. 5. (a) The harvesting cost as a function of the client list size for analytic static model (b) Blocking probability as a function of the iteration number for the analytic dynamic model with the measured residual life for different parameters of N, M with H=71

find that $K = log_{(1-\frac{N}{M})}(1 - B^{\frac{1}{N}})$. It is easy to see that K is inversely proportional to N. However, the harvesting cost is not dominated by the number of required iterations, but by the number of required records it needs to collect. Hence, we are more interested in KN which is equal to $log_{(1-\frac{N}{M})}(1 - B^{\frac{1}{N}}) \cdot N$ which results in a more complex relationship between N and the harvester cost. On the one hand when the list size is large, the harvester enjoys the fact that it discovers a high number of unique SNs in each iteration. However, on the other hand, the larger the client's list, the more SNs the harvester needs to discover, since it needs to discover all the SNs in the client list in order to block the client.

Figure 5 (a) shows the harvesters cost as a function of the list size for different population sizes, M, and different required blocking probability B for the static model. We can learn from the figure that by using the optimal list size (optimal from Skype's perspective), Skype can increase the cost to the harvesters only by less than 25% (from 450K to around 540K, for $B = 95\%$ and $M = 50K$). Hence, changing the list size, is not an effective remedy against SN harvesting. A direct outcome is that harvesting SNs by *Monitoring Skype Connections* technique, where $N = 30$, (see subsection III-B), will be enough to block Skype with high probability.

Figure 5(b) shows the blocking probability for the dynamic model with the experimentally measured SN residual life for different parameters. It is clear that by reducing N, the list size, again has a minor impact on the maximal blocking probability. It only increases the time needed in order to reach the maximal blocking probability. Increasing the population size, M, has more impact on the achievable maximal blocking probability, and this is true since it influences the number of short life span SNs that the harvesters need to learn in a short time. Again, by increasing the number of harvesters, we can reach the same blocking probability.

IX. Conclusion

In this paper we use measurements and modeling and show that it is feasible to harvest the vast majority of the active SNs of Skype. This makes Skype vulnerable to filtering according to SN lists. We suspect that this vulnerability is not unique to Skype, and may be a fundamental problem of P2P networks whose topology can be discovered and used to block the protocol despite their distributed nature. Thus to make the P2P network more resilient a solution to the harvesting technique needs to be provided and this is part of our future work.

Acknowledgments: We would like to thank Sheldon Ross, Michael Tarsi and Jussi Kangasharju for helpful suggestions.

References

[1] S. A. Baset and H. G. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," in *Proc. INFOCOM 2006*, 2006.
[2] K. F. Suh, D. R. Kurose, and D. J. Towsley, "Characterizing and detecting skype-relayed traffic," in *Proc. INFOCOM 2006*, 2006.
[3] S. Ehlert, T. Magedanz, S. Petgang, and D. Sisalem, "Analysis and signature skype voip session traffic," Tech. Rep. NGNI-SKYPE-06B, 2006.
[4] S. Guha, N. Daswani, and R. Jain, "An experimental study of the skype peer-to-peer voip system," in *IPTPS06, Feb. 2006.*, 2006.
[5] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer voip calls on the internet," in *ACM Conference on Computer and Communications Security (November 2005)*, 2005.
[6] "SPam and Open Relay Blocking system (SORBS)." http://www.au.sorbs.net/.
[7] A. Bremler-Barr, R. Goldschmidt, O. Dekel, and H. Levy", "Controlling P2P Applications via Address Harvesting:The Skype Story." http://www.idc.ac.il/publications/files/539.pdf.
[8] S. S. and S. Gummadi, P. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking. 2002, MMCN'02, San Jose, CA, Jan*, 2002.
[9] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," in *ACM SIGCOMM Internet Measurement Workshop, Nov. 2002*, 2002.
[10] R. Bhagwan, S. Savage, , and G. Voelker., "Understanding availability," in *IPTPS 03, 2003*, 2003.
[11] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, , and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *ACM SOSP, 2003*, 2003.
[12] P. B. Godfrey, S. Shenker, , and I. Stoica, "Minimizing churn in distributed systems," in *ACM SIGCOMM 2006*, 2006.
[13] S. I. Gass and C. M. Harris, *Encyclopedia of Operations research and Management Science, 2nd Edition*. Kluwer Academic Publishers, second ed., 2001.
[14] L. Kleinrock, *Queueing Systems, Volume I: Theory*. John Wiley and Sons, first ed., 1975.
[15] Y. Xie, K. A. F. Yu, E. Gillum, M. Goldszmidt, and T. Wobber, "How dynamic are ip addresses," in *ACM SIGCOMM, 2007*, 2007.