# Characterizing and Detecting Skype-Relayed Traffic

Kyoungwon Suh, Daniel R. Figueiredo, Jim Kurose, Don Towsley

Department of Computer Science
University of Massachusetts
Amherst, MA 01003
Email: {kwsuh,ratton,kurose,towsley}@cs.umass.edu

*Abstract*— Networked application developers have recently started to use end-users' computers as relay nodes – application instances that also act as bridges between pairs of hosts running the same application. Relay nodes can bring costs to both users and network operators, at least in terms of increased bandwidth consumption. An interesting problem is to characterize the nature of relayed traffic and to detect its presence in the network. This paper focuses on characterizing and detecting relayed traffic generated by Skype, a popular voice over IP application that uses relays. Our technique relies solely on flow-level properties rather than on application- or protocol-specific information. Using two different controlled experimental environments we generate and collect a large amount of Skype-relayed traffic. We propose several metrics to characterize the nature of relayed traffic. These metrics together with the results obtained from the experimental characterization of Skype-relayed traffic are used to detect Skype-relayed traffic traversing the access point of a large network. We show that the metrics proposed can be used to reliably detect Skype-relayed traffic. Finally, we believe the metrics proposed could be applied more broadly in the detection of relayed traffic generated by other multimedia applications.

## I. INTRODUCTION

Networked applications are using network resources in increasingly ingenious ways to achieve greater scalability and circumvent security limitations. One such example is the emergence of relay nodes – application instances that not only provide application services to local users, but also act as bridges between remote nodes running the same application. The use of a relay allows two nodes that could not otherwise communicate (e.g., due to firewall or NAT [1] restrictions) to do so, and can improve the quality of the communication between two nodes by avoiding congested or faulty paths [2].

Although relay nodes are useful to application designers, they have disadvantages from the perspective of users and network operators. A node chosen to act as a relay must bear the cost of relaying traffic. This cost is evident to users in the form of slower communication (due to bandwidth use by the relayed traffic) or financial costs, if payment for network access is a function of bandwidth usage. Network operators and small ISPs may also see drawbacks, as relayed traffic can increase the amount of traffic entering and leaving their network. For example, a single machine running Skype [3] within our campus network relayed voice traffic at an average rate of 2.1MB per hour over a period of 20 days.

Given the increasing use of application-level relay nodes and their potential costs to users and network operators, it is important to characterize the nature of relayed traffic and to detect its presence in a network. Network operators would probably like to know if traffic is being relayed through hosts that belong to their network. Although the concept of relaying is not new, an increasing number of applications have started to make use of relay nodes. We can distinguish such applications into two categories: multimedia applications (where data flow continuously and have a moderate or high average bit rate, e.g., greater than 15 kbps), and interactive applications (where data flow sporadically in short bursts and generally have a lower average bit rate, e.g., less than 5 kbps). Examples of multimedia applications that use relays are Skype [3] and End System Multicast (ESM) [4], while examples of interactive applications are Internet Relay Chat (IRC) [5], MSN Messenger, and SSH. Multimedia applications that use relays are likely to be more costly to users and network operators, as these applications are much more bandwidth demanding.

In this paper, we characterize relayed traffic generated by a popular voice over IP application, namely Skype [3], and propose a methodology for detecting Skype-relayed traffic. We propose the use of several metrics that can effectively characterize the nature of Skype-relayed traffic. In order to obtain relayed traffic, we develop two different experimental environments to generate and collect a large amount of Skype-relayed traffic. This data is characterized using the different metrics proposed. The observations obtained from this characterization are used to propose a methodology for detecting Skype-relayed traffic. Detection is performed by setting thresholds for the various metrics. By tuning the thresholds, the desired balance between true positives and true negatives can be obtained. Finally, we apply the detection methodology to a large aggregate traffic trace collected at the access point of our university. Our results show that the metrics considered for characterizing relayed traffic can reliably detect Skype-relayed traffic.

An important consideration when designing a traffic classification methodology is the use of application- or protocol-specific information, such as well-known port numbers. Since applications are increasingly becoming more flexible and diverse, new methods for traffic classification no longer rely on this information [6]. As in such approaches, our methodology for characterizing and detecting relayed traffic also does not rely on application- or protocol-specific information. However,

our methodology does take into consideration the fact that the application is transmitting real-time voice traffic. For this same reason, we believe our methodology could be extended to characterize and detect relayed traffic generated by other multimedia applications.

The evaluation of our detection methodology is also a challenge, since it is not trivial to know precisely when traffic is indeed being relayed. In order to obtain a benchmark for Skype-relayed traffic detection, we propose a heuristic for identification of Skype traffic that uses Skype-specific information. Using this information, we obtain the true set of Skype-relayed traffic. Although of independent interest, the heuristic is used only to evaluate the detection methodology.

The detection of relayed traffic has been addressed in other contexts. The problems of detecting "stepping stones" [7–9] and identification and reconstruction of large attacks [10] have been extensively investigated within the intrusion detection context. The problem of flow correlation in mix networks [11, 12] has more recently been studied in the context of anonymity systems. However, none of these efforts have investigated relayed traffic generated by multimedia applications, such as Skype. Moreover, the methodology developed for these prior studies do not necessarily apply in our context. To the best of our knowledge, this is the first work to characterize relayed traffic generated by a multimedia application (i.e., Skype) and to propose a specific methodology for its detection.

The remainder of this paper is structured as follows. In the next section we define relayed traffic and present the different metrics used for its characterization. In Section III, we discuss our controlled experimental environment and characterize Skype traffic using the proposed metrics. Section IV presents the payload-based Skype traffic identification heuristic. In Section V, we evaluate the performance of detecting Skype-relayed traffic by analyzing an aggregate traffic trace collected at our university's gateway. Section VI presents a discussion of the related work. Finally, Section VII concludes the paper with a summary of our contributions and discussion of future work.

## II. PROBLEM DEFINITION

Relayed traffic is traffic generated by a given end-host, that is passed through one or more end-hosts ("relays") before reaching its final destination. Figure 1 shows an example of relayed traffic. End-host $n_1$ generates traffic that first passes through end-host $n_r$ before reaching its final destination, end-host $n_2$. From the perspective of the transport layer, there are two bidirectional connections in this example: one between $n_1$ and $n_r$ and another between $n_r$ and $n_2$. As we will see shortly, these connections may differ in type (e.g., one connection may be TCP and the other UDP), duration, and throughput.

A relay node may or may not be interested in the data traffic being relayed, and may or may not perform transformations on the relayed data. For example, node $n_r$ may be watching a streamed video generated by node $n_1$ that is then forwarded to node $n_2$. On the other hand, node $n_r$ might simply act as a bridge between node $n_1$ and $n_2$, receiving and sending
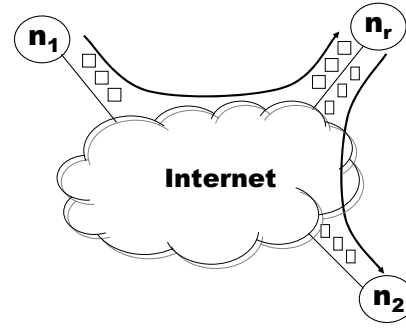


Fig. 1.  Example of relayed traffic between end-hosts $n_1$ and $n_2$

packets. A relay node may also actively change the nature of the traffic being relayed. Many different types of transformations are possible, ranging from simple transformations such as transport protocol changes (e.g., UDP to TCP), application-level header modifications, and fragmentation/composition of data packets, to more complex operations, such as data compression/decompression, and CODEC changes. In principle, a relay node can perform any kind of transformation on the relayed data. In practice, however, current relay nodes perform only simple transformations, most likely to minimize the computational and communication load placed on relay nodes.
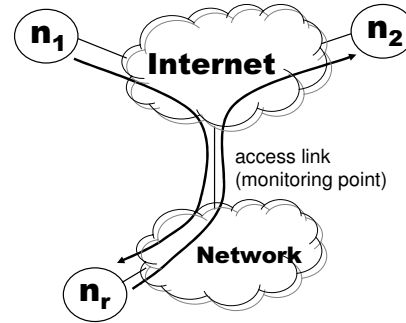


Fig. 2.  Network scenario under consideration. End-host $n_r$ is used as a relay by two end-hosts outside the network

In this paper, we take the perspective of the operator of a large network, who is monitoring incoming and outgoing traffic at an access link, as illustrated in Figure 2. Our goal is to determine whether traffic is being relayed through some end-host belonging to the network, using only IP and transport header information collected by the monitor to make this decision. In general, this problem is difficult and we will focus on relayed traffic generated by Skype. We will use the fact that Skype-relayed traffic is voice traffic (which poses constraints on maximum delays and minimum bit rates) but will not use application specific information such as well-known port numbers or distinct packet sizes, nor will we examine packet payloads.

Thus far, we have only informally defined a relay and the traffic being relayed; let us now make this definition more precise. Intuitively, we know that a relay takes an input "flow"

of data and forwards the data to its final destination. But what is meant by a "flow"? In many measurement studies, a "flow" is defined as a sequence of packets with the same 5-tuple (source and destination IP addresses, source and destination port numbers, protocol number) where adjacent packets are not too far apart from each other (e.g., a 60 second maximum inter-arrival time between adjacent packets in the same flow) [13, 14]. But this definition alone will not suffice for our purpose for several reasons. First, we add the notion of a direction to a flow (e.g., source-to-relay, or relay-to-destination). Second, a flow can carry both data and control traffic, although only data traffic is relayed. Third, a single flow can contain disjoint sets of relayed data traffic interleaved with control traffic. These characteristics can appear in flows generated by Skype for example. Figure 3 shows a single Skype flow containing two voice calls that were relayed (the first call starting at approximately 200 seconds and ending at approximately 400 seconds, and the second call starting at approximately 800 seconds) and were separated by some control traffic.
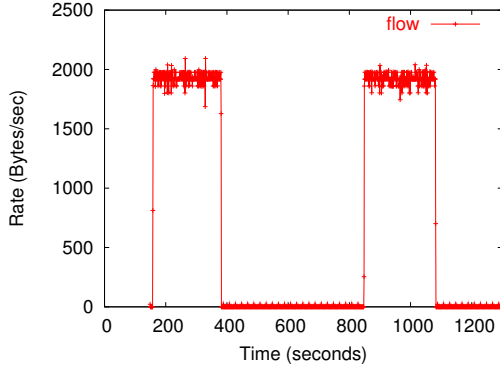


Fig. 3. Example of a single Skype flow containing two different relayed voice calls (each call is mapped into a separate burst of packets)

Given these considerations, we introduce the notion of a *burst of packets* (to avoid confusion with the colloquial use of the word "flow") in order to characterize relayed traffic. A burst of packets is a contiguous piece of a flow and is defined as a sequence of packets that has a minimum average data rate (e.g., at least 10 kbps) and a minimum duration (e.g., at least 30 seconds). By requiring a burst of packets to have a minimum average bitrate we can distinguish between data and control traffic, since the average bitrate of data traffic is much higher than that of control traffic. This distinction is important, as we are interested in characterizing only relayed traffic. Also, note that a single flow can contain multiple bursts of packets, which in Skype can correspond to multiple voice calls, as illustrated in Figure 3.

We use a simple change detection algorithm to detect the start time and end time of a burst of packets within a flow. In particular, we adopt an EWMA (Exponential Weighted Moving Average) to keep track of the data rate associated with a flow. The EWMA of a given flow is defined as follows:

$$A_i = (1 - \alpha)A_{i-1} + \alpha I_i \qquad (1)$$

where $I_i$ represents the average data rate of the $i$-th second of the flow and $A_0$ and $I_0$ are set to zero. Note that the EWMA is updated every second since the start of a flow. A burst starts if the EWMA goes above a certain threshold ($A_i > R_1$) and ends when the EWMA goes below another threshold ($A_i < R_2$). In order to detect the start of a burst we use a more aggressive parameter for the EWMA (large $\alpha$). However, we use a more conservative parameter to detect the end of a burst (smaller $\alpha$). In particular, we use $\alpha = 0.75$ and $\alpha = 0.15$ for the two cases, respectively. Each flow has a single EWMA value and its parameter, $\alpha$, is changed as soon as the start/end of a burst is detected.

Each burst $i$ contains a sequence of packets and has several quantities of interest:

- $N_i$: number of packets in burst $i$
- $T_i^j, j = 1, \ldots, N_i$ : timestamp of the $j$-th packet of burst $i$
- $Z_i^j, j = 1, \ldots, N_i$ : size in bytes of the $j$-th packet of burst $i$
- $S_i = T_i^1$ : start time of burst $i$ in seconds
- $E_i \geq T_i^{N_i}$ : end time of burst $i$ in seconds
- $D_i = \lceil E_i - S_i \rceil$: duration of burst $i$
- $Y_i^k, k = 1, \ldots, D_i$ : total number of bytes sent during the $k$-th second of burst $i$ (mathematically, $Y_i^k = \sum_{l=1}^{N_i} I(T_i^l - S_i \in [k-1, k))Z_i^l$, where $I(\cdot)$ is the indicator function, returning 1 when its argument is true and 0 otherwise)
- $B_i = \sum_{j=1}^{N_i} Z_i^j$ : total number of bytes carried by burst $i$
- $R_i = B_i/(E_i - S_i)$ : average data rate of burst $i$

Note that it suffices to monitor TCP/UDP packet headers to construct flow and burst structures.

We use the notion of a burst to characterize and detect relayed traffic. We assume bursts carrying relayed traffic must have opposite directions (one entering, the other leaving the network), have the same end-host (same IP addresses) within the network being monitored and have different end-hosts (different IP addresses) outside the monitored network [1]. The relay detection problem is to determine if two bursts that could be carrying relayed traffic (e.g., candidate bursts) are in fact carrying relayed traffic.

In order to detect relayed traffic we focus on a few statistical metrics involving a pair of bursts. Let $i$ and $j$ denote two bursts. We consider the following metrics:

- $S_{i,j} = |S_i - S_j|$ : the difference between the start times of bursts $i$ and $j$
- $E_{i,j} = |E_i - E_j|$ : the difference between the end times of bursts $i$ and $j$
- $B_{i,j} = B_i/B_j$ : the ratio between the number of bytes carried by bursts $i$ and $j$
- $X_{i,j}$ : the maximum cross correlation between time series $Y_i$ and $Y_j$

---

[1] We will not consider the scenario where traffic is relayed through more than one end-host within the monitored network. In particular, note that Skype uses at most one relay node for any given voice call.

Here $X_{i,j}$ is defined as $\max_{d \in \{-D_i,...,0,...,D_i\}} x_{i,j}(d)$, where $x_{i,j}(d)$ is the cross correlation between time series $Y_i$ and $Y_j$ at lag $d$ [15]. For completeness, this is defined as:

$$x_{i,j}(d) = \frac{\sum_k (Y_i^k - R_i)(Y_j^{k-d} - R_j)}{\sqrt{\sum_k (Y_i^k - R_i)^2}\sqrt{\sum_k (Y_j^{k-d} - R_j)^2}} \quad (2)$$

The metrics defined above will help us assess if two candidate bursts indeed carry relayed traffic. In particular, consider two bursts corresponding to Skype-relayed traffic. Due to the application requirements (i.e., voice over IP), the start time difference and end time difference of the bursts should be very small, as packets cannot be stored at the relay node for long periods of time. Moreover, since Skype does not perform complex transformations on the relayed traffic, the burst size ratio should be close to one and the time series of the two bursts should have a very high degree of correlation. These observations will allow us to detect Skype relayed traffic.

There are other metrics that can be adopted to characterize and detect relayed traffic. Prior work on the relay detection in other contexts has proposed other metrics, such as the packet counts in ON and OFF periods, cumulative byte count difference. However, they are not applicable to the characterization or detection of multimedia traffic relays. For example, the technique based on On-Off periods, which has been applied to interactive applications [7, 8] fails because multimedia traffic patterns do not necessarily exhibit On-Off behavior. The technique based on *packet counts* does not perform well because packet splitting or merging can (and does in the case of Skype) occur at the relay nodes. Another approach would be to use the cumulative byte count difference between incoming and outgoing bursts as a metric; a similar idea was proposed in [9]. Intuitively, this metric should work well under the assumption that a relayed burst conserves byte count (i.e., number of bytes arriving at the relay should be equal to the number of bytes departing the relay). Unfortunately, byte count conservation is a strong assumption within the domain of multimedia applications and can sometimes be (and is in the case of Skype) violated. Moreover, this metric is fairly sensitive to packet losses that are not retransmitted, as in the case of traffic relayed using the UDP protocol (as in the case of Skype).

## III. CHARACTERIZATION OF SKYPE-RELAYED TRAFFIC

In this section we characterize Skype-relayed traffic using data collected from two different controlled experiments. In the first experiment, we control the two nodes that are used to make a relayed Skype call. This allows us to generate relayed traffic with different characteristics, such as different transport protocols and different packet loss rates. In the second experiment, we control the relay node used to relay traffic between two Skype users. This provides us with real relayed traffic, generated by real Skype users. The goal of both experiments is to collect a large amount of Skype-relayed traffic and then characterize this data using the metrics presented in the previous section.

| Parameters | Value |
|---|---|
| Transport protocol type | UDP_in_UDP_out UDP_in_TCP_out TCP_in_TCP_out |
| Packet loss probability | 0.00 and 0.05 |
| Duration of each call | approximately 3 minutes |
| Number of UDP_in_UDP_out calls | 505 |
| Number of UDP_in_TCP_out calls | 269 |
| Number of TCP_in_TCP_out calls | 307 |
| Total number of relays used | 1081 |

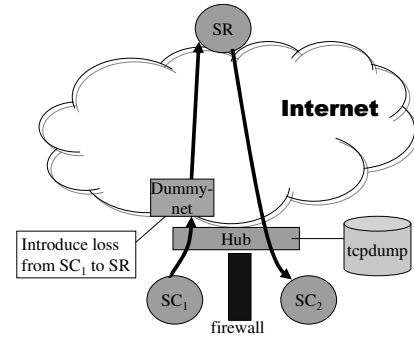### A. Experiment I: Relayed traffic generated between two controlled Skype nodes



Fig. 4. Configuration of controlled experiment I

In this first experiment we control the two hosts running Skype that are used to make a relayed Skype voice call. By using a firewall to block packets between the two hosts we force Skype to use a relay node. This relay node is not under our control and is chosen by Skype. Moreover, by adequately configuring the firewall, we can control the transport protocol used by Skype both to and from the relay node. Since Skype can use both TCP and UDP to deliver voice packets [16], we have four possible pairs of transport protocol combinations: UDP_in-UDP_out, TCP_in-TCP_out, UDP_in-TCP_out, and TCP_in-UDP_out, where "in" and "out" represent traffic flowing to and from the relay node, respectively. Finally, we also introduce artificial packet loss on the path between one of the end-hosts and the relay node. By controlling the packet loss rate on one of the paths, we obtain Skype-relayed traffic under different loss regimes. We used dummynet [17] to generate Bernoulli losses. The experimental setting discussed here is illustrated in Figure 4. Note that data is collected by a host connected to the same hub where the two controlled nodes are connected. Using tcpdump [18] we capture all packets sent from $SC_1$ to $SR$ and all packets sent from $SR$ to $SC_2$. Note that we capture packets sent from $SC_1$ *before* losses are artificially introduced.

The above experimental setting was used to make hundreds

Fig. 5. Empirical cumulative distribution of burst start time difference
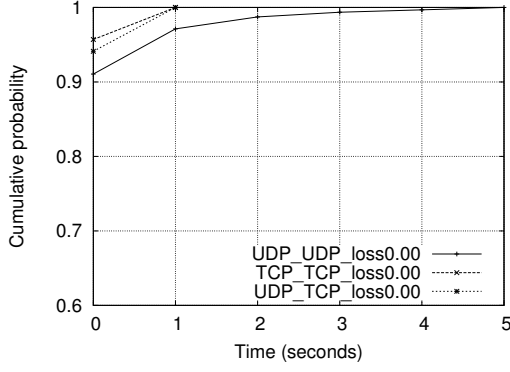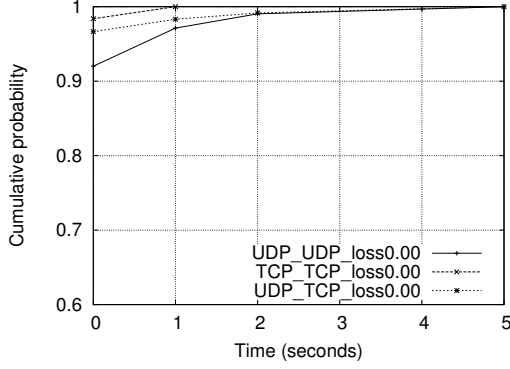


Fig. 6. Empirical cumulative distribution of burst start time difference



Fig. 7. Empirical cumulative distribution of burst end time difference
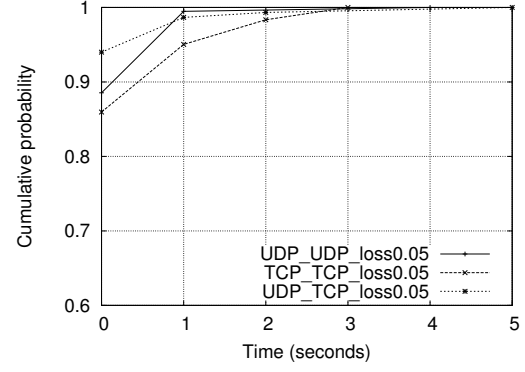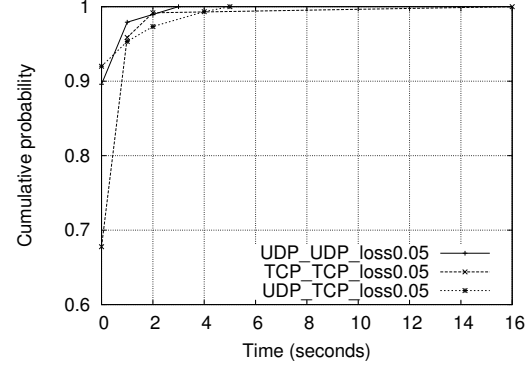


Fig. 8. Empirical cumulative distribution of burst end time difference

of [2] Skype-relayed calls (we automated the call setup/tear down procedure) each of duration three minutes. Throughout the experiment, several different relay nodes were used and although most of them were located in the United States, some relay nodes in Europe and Asia were also used. Table I summarizes the parameters and results from the experiment.

Using the data collected we characterize Skype-relayed traffic based on the metrics described in the previous section. This characterization provides a benchmark for Skype-relayed traffic that is used to generate guidelines for detection of Skype-relayed traffic. As we next discuss, Skype-relayed traffic exhibits particularities that can be used to identify it without resorting to any application-specific information.

The results are presented for each transport protocol pair and for different packet loss rates. We start investigating the start time difference between two bursts of Skype-relayed traffic (recall $S_{i,j}$, with $i$ and $j$ corresponding to a pair of Skype-relayed bursts). Figures 5 and 6 show the empirical cumulative distribution of the start time difference. Note that all relayed

bursts have a start time difference of less than 5 seconds. In fact, the vast majority of the relayed bursts (i.e., 99% of them) has a start time difference of less than 3 seconds. We also observe that introducing a packet loss rate of 5% has little effect on the start time difference (Figure 6).

Figures 7 and 8 show the empirical cumulative distribution of the end time difference between two bursts of Skype-relayed traffic (recall $E_{i,j}$, where in this case $i$ and $j$ are Skype-relayed bursts). The characterization of the end time difference is similar to the start time difference when no artificial packet loss is introduced. Note that all end time differences are less than 5 seconds in this case. However, for the case with 5% packet loss end time differences are longer, especially for the TCP_in_TCP_out case. This increase is due mainly to packet retransmissions at the end of the voice call and the loss of TCP FIN packets, which are responsible for gracefully terminating the TCP connection. In any case, all end time differences between two bursts of Skype-relayed traffic are less than 16 seconds.

Figures 9 and 10 show the empirical complementary cumulative distribution of the maximum cross correlation between two bursts of Skype-relayed traffic (recall $X_{i,j}$, where in this case $i$ and $j$ are Skype-relayed bursts). We observe that the transport protocol pair significantly impacts the maximum cross correlation. As one might expect, the UDP_in_UDP_out

---

[2]We note that the use of a fixed call duration time (i.e., 3 minutes) had little influence on the characterization of relayed traffic. As long as the call has a minimum duration (such as one minute) the duration of the call will have little impact on the metrics being studied. In particular, Experiment II shows similar results to Experiment I, where voice calls are real and have variable duration.
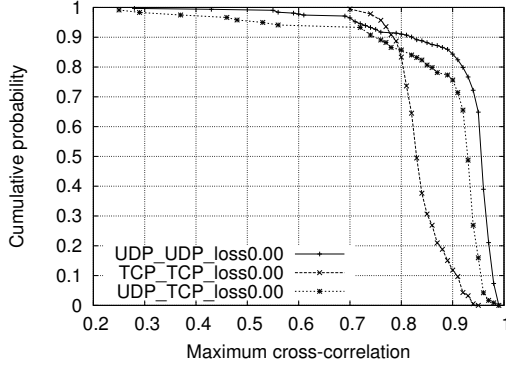
Fig. 9. Empirical complementary cumulative distribution of burst maximum cross-correlation
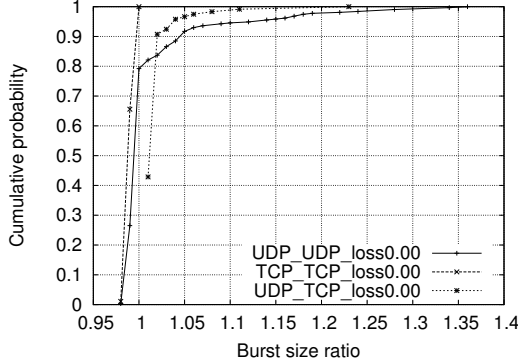


Fig. 10. Empirical complementary cumulative distribution of burst maximum cross-correlation



Fig. 11. Empirical cumulative distribution of burst size ratio



Fig. 12. Empirical cumulative distribution of burst size ratio

combination yields the best maximum cross correlation, as the traffic in both connections is not shaped by any congestion control or flow control mechanism. In particular, note that 85% of the relayed traffic has a maximum cross correlation greater than 0.9 in this case. A similar result is obtained for the UDP_in_TCP_out combination, although with a slightly lower maximum cross correlation. Finally, the TCP_in_TCP_out yields an even lower maximum cross correlation, especially under the 5% packet loss rate. This is expected, as the first TCP burst will contain many packet retransmissions while the second burst will not. In any case, the maximum cross correlation is still surprisingly high, even in the TCP_in_TCP_out. In fact, 95% of all Skype-relayed traffic has a maximum cross correlation of at least 0.37, regardless of transport protocol combination and packet loss rate (not shown in graphs).

Finally, we investigate the burst size ratio between two bursts of Skype-relayed traffic. Figures 11 and 12 show the empirical cumulative distribution of the burst size ratio (recall $B_{i,j}$, where in this case $i$ and $j$ are Skype-relayed bursts). Note that for the case where no artificial packet loss is introduced, the burst size ratio of all Skype-relayed traffic is very close to 1. For the case of 5% packet loss rate, the empirical cumulative distribution for the burst size ratio changes noticeably. In particular, the distribution, which in the case of no artificial
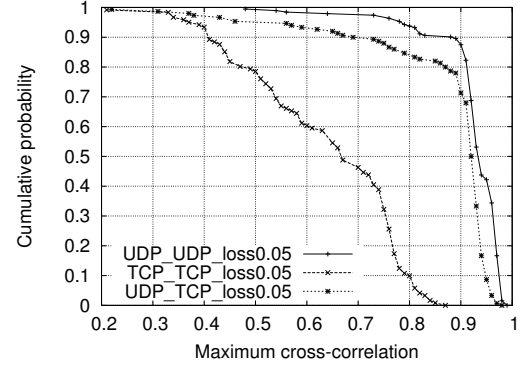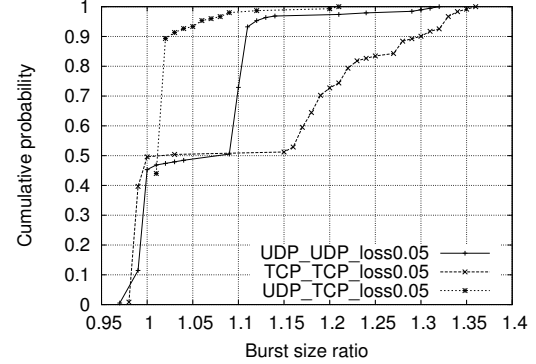
packet loss has a single mode, now has more than one mode. Note that in this regime, the TCP_in_TCP_out protocol combination has the largest burst size ratio, as many packets will be retransmitted by the protocol.

### B. Experiment II: Traffic relayed over a controlled Skype node

In this second experiment we control and monitor the relay node used by two Skype nodes. By running Skype continuously on an end-host with a powerful CPU (e.g., Pentium 4) and with a good connection to the Internet (e.g., 100Mbps Ethernet) for a prolonged period of time (e.g., a few days), the host will end up being used as a relay node by other Skype nodes. Figure 13 illustrates the experimental scenario considered. Note that the relay node is connected to the same hub as the dedicated host used to collect the data. We use tcpdump [18] to capture all packets entering and leaving the relay node. In contrast to the previous experiment, in this experiment the data collection point and the relay node are very close to each other (i.e., in the same LAN).

By running this experiment for 20 days, hundreds of Skype calls were relayed through our controlled host. Our Skype host relayed traffic for Skype hosts located in several parts of the world, including Europe, Asia, and South America. With the data collected in this experiment, we can characterize Skype-relayed traffic of real voice calls from a more vast end-host population than the two end hosts used in experiment I.
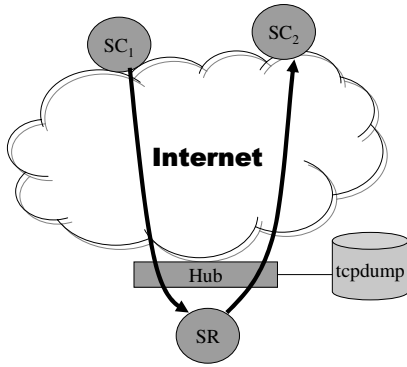
Fig. 13.   Configuration of controlled experiment II

| Parameter | value |
|---|---|
| Minimum burst duration | 30 seconds |
| Minimum packet count for flow | 300 packets |
| Maximum start time difference | 30 seconds |
| Maximum end time difference | 30 seconds |
| Conflict resolution criteria | Smaller burst start time difference |

Although no other application was running on our controlled host, not all traffic entering and leaving the host was Skype voice traffic. In particular, Skype itself can generate and receive a significant amount of traffic that is not voice traffic. Instead, this traffic is required for maintaining the peer-to-peer network and for providing services such as the location of users in the network (i.e., directory service). This kind of traffic becomes significant when a running instance of Skype is promoted and starts to operate in "super node" mode [16] (which occurred during our experiments).

Since we are interested in characterizing only relayed voice traffic, we had to exclude all "non-voice" traffic from the data collected. Separating voice traffic is not a trivial task as it is hard to guarantee that a pair of bursts entering and leaving our host is indeed a relayed voice call. In order to minimize the chances of mistakes, we are aggressive in our approach to exclude "non-voice" traffic (i.e., possibly excluding some legitimate relayed voice traffic, but unlikely considering any "non-voice" traffic). In particular, only pairs of bursts that conform to the parameters shown in Table II are considered as voice traffic. All other bursts collected are discarded.

Some of the parameters chosen in Table II were obtained from the results of the previous experiment. In particular, we use a maximum of 30 seconds for start and end time differences, as such value is much larger than any instance of the previous experiment. Finally, it is possible that more than two bursts of packets start and end within 30 seconds of each other, generating multiple candidates for relayed bursts. In this case, we use a simple priority rule, where the pair of bursts which has the smallest start time difference is chosen as the relay pair. However, no conflict needed to be resolved in the data collected for this experiment.

Over the period of 20 days, we observed a total of 341 Skype-relayed bursts of packets, corresponding to a total of 1.11Gbytes of data. We now characterize the nature of these relays using the same metrics as before. Figures 14 and 15 show the empirical cumulative distribution for the start and the end time differences between two bursts of Skype-relayed traffic, respectively. Note that for 99% of the Skype relayed traffic, the start time difference is less than 6 seconds while the end time difference is less than 10 seconds. Again, we observe

that start and end time differences for the vast majority of Skype-relayed traffic are small. This observation is consistent with the results of the previous experiment when no artificial packet loss was introduced.

Figure 16 shows the empirical complementary cumulative distribution of the maximum cross correlation between two bursts of Skype-relayed traffic. The vast majority of maximum cross correlation observed (i.e., 99% of them) is above 0.41. Note that the distribution for maximum cross correlation is very similar to the distribution of UDP_in_UDP_out and UDP_in_TCP_out protocol pairs of the previous experiment. This is because most of the relays observed in this experiment are of the types UDP_in_UDP_out and UDP_in_TCP_out (i.e., 96% of relays). However, the ratio of relays that have a maximum cross correlation of at least 0.95 is even higher than that of the UDP_in_UDP_out combination in the previous experiment. One possible reason for the higher correlation is that in this experiment, the monitoring point is very close to the relay node (i.e., same LAN). However, in the previous experiment, the monitoring point was very close to the two communicating end-hosts and potentially far from the relay node. This observation is to our advantage as our goal is to detect Skype-relayed traffic by monitoring the access link of a large network, in which case, relay nodes will be close to the monitoring point (i.e., same WAN).

Finally, Figure 17 shows the distribution of the burst size ratio between two bursts corresponding to Skype relayed traffic. Note that 99% of the burst size ratios lie below 1.15, indicating that the vast majority of relayed bursts in Skype have very similar sizes.

## IV. PAYLOAD-BASED SKYPE TRAFFIC IDENTIFICATION

In this section we describe a heuristic that can be used to identify Skype traffic. Although the heuristic uses Skype-specific information present in the packet payload, our relay detection mechanism does not. The goal here is to filter out non-Skype traffic and obtain a set of Skype-only traffic which can then be used as a benchmark for the relay detection mechanism. Note that the heuristic itself may be of interest, as it provides an effective method for identifying Skype traffic when packet payloads are available. However, the heuristic does *not* detect Skype-relayed traffic, it only identifies Skype traffic.

Before describing the heuristic, we first give some details about the functionality of Skype. Along with other peer-to-peer (P2P) applications, Skype does not use any well-known
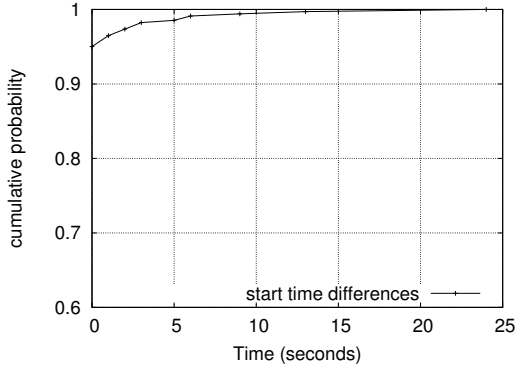
Fig. 14. Empirical cumulative distribution of burst start time differences of voice calls relayed by our Skype node
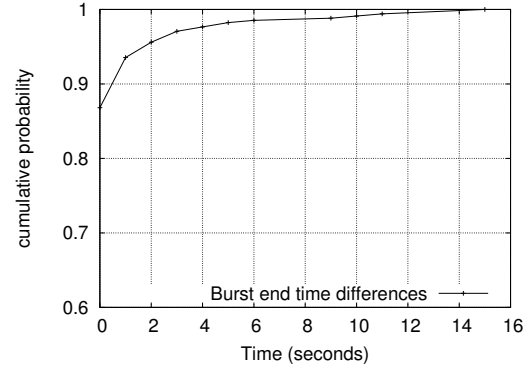


Fig. 15. Empirical cumulative distribution of burst end time differences of voice calls relayed by our Skype node
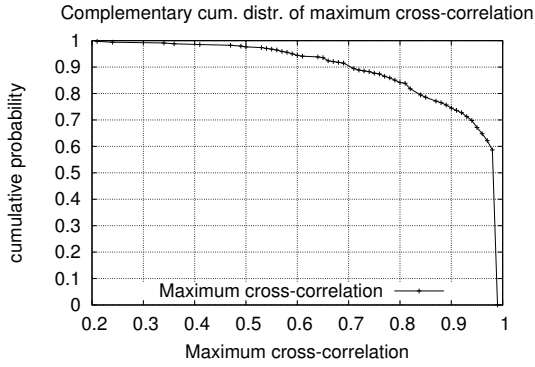


Fig. 16. Empirical complementary cumulative distribution of burst maximum cross-correlation of voice calls relayed by our Skype node
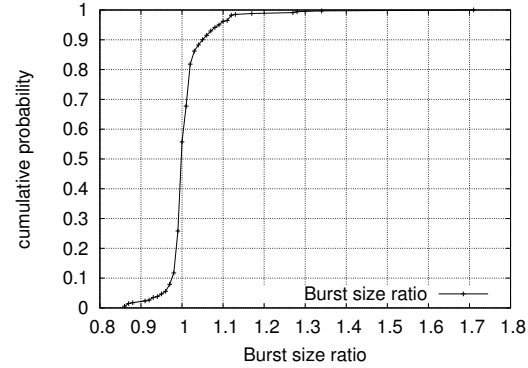


Fig. 17. Empirical cumulative distribution of burst size ratio of voice calls relayed by our Skype node

port number. However, it does use a single port number which is randomly chosen when the application is first installed. This port number is used to receive incoming TCP connections and UDP packets. Moreover, all outgoing UDP packets also contain this port number. After being randomly chosen, the port number is saved locally and used on all future executions of the application. Note however that users can explicitly change this port number by configuring Skype to use either port 80 or 443 (well-known port numbers for HTTP and HTTPS protocols).

Skype users must first logon and authenticate themselves before using the application. The logon process is composed of two subprocesses: software version verification and user authentication. For the software version verification, Skype makes a TCP connection to a well-known server (i.e., ui.skype.com) and reports the application version currently running [16, 19]. We noticed that even if the option for automatic Skype version verification was explicitly disabled, Skype still contacted the well-known server. The second subprocess is user authentication. This is done either directly by contacting some predetermined logon server or by contacting one or more Skype *super nodes*. Besides the communication during the logon and authentication process, the application also contacts a set of Skype super nodes in order to register itself into the

P2P network, which is done by sending UDP packets to the port numbers they are listening to.

The key observation is that these control packets departing a Skype node have as source port the randomly chosen port number that will be used for Skype voice traffic, including relayed traffic. Our heuristic works as follows. We first identify the IP address of all hosts that have executed Skype. This is done by inspecting the payload of packets destined to the well-known server [3]. Because of the mandatory version verification process, which is always done using a single server, it is easy to determine the IP address of hosts running Skype within a network. However, determining the port number used by a given Skype host to send/receive voice traffic is more difficult. We correlate the version verification message with the fact that super nodes are contacted in order to obtain the (possibly) random port number used by a given Skype host. Note that these two events usually occur closely to each other in time (i.e., as soon as the application is launched). In particular, we count how many times a given source port number is used right after (or before) a Skype version verification packet is observed. If the same port number is used many times

---

[3]In practice, we do not need to inspect the payload of the packet as all connections to port 80 of the well-known version verification server are requests initiated by Skype clients (the server does not serve any web content).

TABLE III

| Date | Day | Start | Dur | Direc. | Packets | Bytes | Num. of flows |
|------|-----|-------|-----|--------|---------|-------|---------------|
| 2005-05-09 | Mon | 15:00 | 17 hours | Bi-direc. | 328M | 414 GBytes | 1501K |

to different hosts within the next few packets, we say that this port is the Skype port number. Knowing the IP address and port numbers of identified instances of Skype within the network allows us to later identify Skype voice traffic sent/received by this end-host.

It has recently been argued that some versions of Skype always contacts a limited number of hard-coded bootstrap servers using UDP messages [16]. Therefore, we could alternatively obtain the UDP port number of a given Skype instance by simply monitoring messages to these known servers. However, it is not clear whether the list of hard-coded IP addresses changes from one version of Skype to another.

It is important to note that the heuristic above would not work with Skype instances that run on hosts using DHCP or that are behind a NAT. The DHCP could assign a different IP the host in subsequent executions of Skype, as a NAT box could assign a different port number in subsequent executions. However, most of the end-hosts in our campus network, including the hosts in dormitory area, have a fixed IP address and are not behind NATs. Moreover, hosts running behind NAT boxes are not used to relay Skype traffic.

## V. DETECTION OF SKYPE-RELAYED TRAFFIC

In this section, we evaluate the effectiveness of using the proposed metrics as a mechanism to detect Skype-relayed traffic. We establish the true population of Skype-relayed traffic present in a large packet trace and use this to measure the performance of the proposed metrics. After presenting the results, we discuss some learned lessons.

### A. Experimental setting

We consider the experimental scenario illustrated in Figure 2, where the access link of a large network is monitored. In particular, we consider our campus network, which is composed of thousands of computers and users. We monitor the gigabit access link connecting the campus network to the commercial Internet service provider. A packet capture card (called a *DAG* [20] card) copies all packet headers (including part of the payload) traversing the link (in both directions) to disk along with an accurate time stamp. Using the monitoring infrastructure, we collect a 17-hour packet trace. Table III shows the details of the trace collected.

### B. Determining the true population of Skype-relayed traffic

In order to identify as many Skype instances as possible within the campus network, we monitored the access link of the campus network for a period of 10 days before collecting the 17-hour packet trace. Whenever a Skype client was identified within the monitored network (using the heuristic described in Section IV), we registered its IP address and port number into a database.

This set of IP addresses and port numbers was then used to filter out all Skype traffic from the 17-hour packet trace that was subsequently collected. However, not all Skype traffic is voice traffic. As discussed earlier, Skype nodes can generate control traffic. We discard such traffic by ignoring flows that do not have any bursts or that have a very low average bit rate (i.e., less than 10 kbps). We are left with a packet trace containing (almost surely) only Skype voice traffic.

Finally, we determine the set of Skype-relayed traffic by analyzing the Skype-only packet trace. In particular, we compare the respective start and end time of pairs of Skype voice traffic bursts that have a common host within our network and that are in different directions (one flow entering, the other leaving the network). If the respective differences are smaller than 30 seconds, we say that these two bursts are indeed Skype-relayed traffic. Using this method, a total of 381 Skype-relayed traffic bursts were identified in the 17-hour trace collected. We will refer to this set of relayed bursts as the *true population* of Skype-relayed traffic. In particular, these relays will provide a benchmark to evaluate our detection mechanism which does not use any application-specific information.

### C. Evaluation of results

In order to evaluate how the metrics proposed fare in detecting Skype-relayed traffic, we will consider true positive and true negative ratios. Formally, the true positive and true negative ratios are given as follows:

- true positive ( = 1 - false negative) =

$$\frac{\text{Num. of true Skype relays classified as Skype relays}}{\text{Num. of true Skype relays}}$$

- false positive ( = 1 - true negative) =

$$\frac{\text{Num. of false Skype relays classified as Skype relays}}{\text{Num. of false Skype relays}}$$

The entire Skype-like relayed traffic population is obtained by considering as relays all pairs of bursts that conform to the parameters summarized in Table II. This yields a total of 12193 possible relayed bursts, from which only 381 are true Skype relays. Of course, most of these pairs of bursts are not actual relays. By a careful choice of threshold for each of the proposed metrics, we can drastically reduce this set, while removing very few Skype relays. In particular, each choice
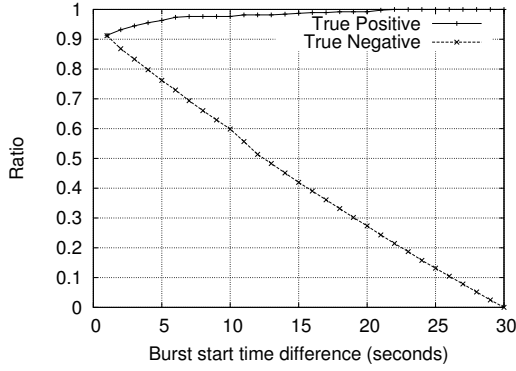
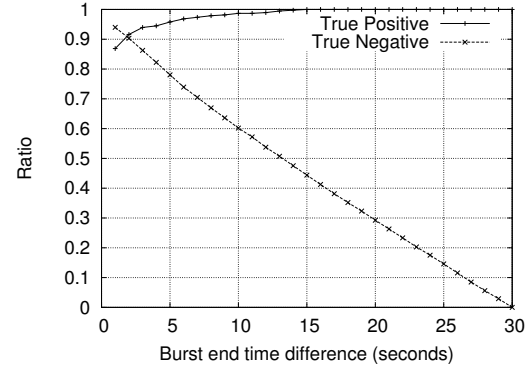Fig. 18. True positive vs. True negative using start time difference as a classifier



Fig. 19. True positive vs. True negative using end time difference as a classifier
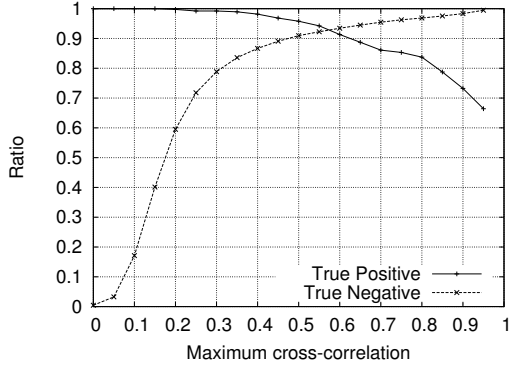


Fig. 20. True positive vs. True negative using maximum cross correlation as a classifier
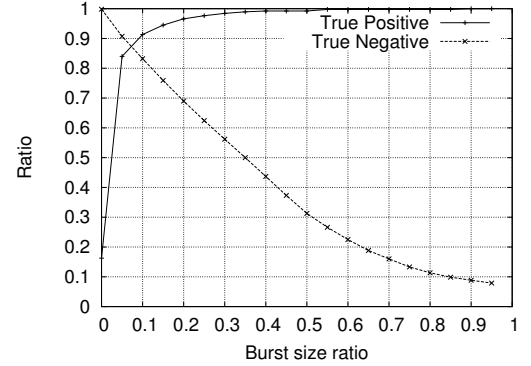


Fig. 21. True positive vs. True negative using burst size ratio as a classifier



Fig. 22. Receiver operator characteristic curve (ROC) for each classifier

of threshold for each parameter induces true positive and true negative ratios.

We first consider each metric in isolation. This will give insights on how each metric performs, individually, in detecting Skype-relayed traffic. We will also consider setting thresholds for multiple metrics at the same time. Intuitively, this should yield a better detection of Skype-relayed traffic.

Figure 18 shows the true positive and true negative ratios when the start time difference is used as the sole criteria to detect Skype-relayed traffic. Note that each threshold value for the start time difference induces different true positive and true negative ratios. As expected, by increasing the threshold we obtain a higher true positive ratio but at the same time a lower true negative ratio (as more non Skype-relayed traffic is wrongly identified as Skype relays). Note that by considering pairs of bursts that start at most 1 second apart (threshold of 1 second) we obtain true positive and true negative ratios larger than 0.90. This indicates that the start time difference can effectively be used to correctly identify Skype-relayed traffic. Figure 19 shows the true positive and true negative ratios when the end time difference is used as the sole criteria to detect Skype-relayed traffic. The results are similar to the start time difference.

Figure 20 shows the true positive and true negative ratios when maximum cross correlation is used. Note that this metric
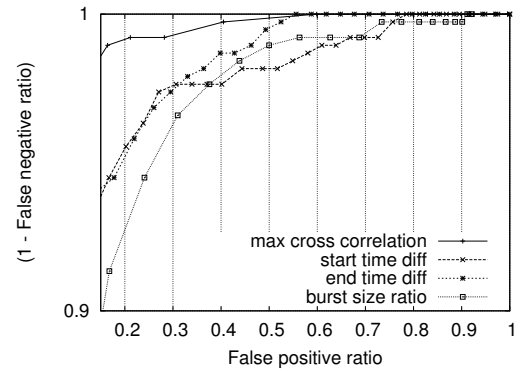
alone yields 0.92 true positive ratio and 0.92 true negative ratio (when the threshold is set to 0.55). Within the different metrics, maximum cross correlation provides the best criteria for detection of Skype traffic. Finally, Figure 21 shows the true positive and true negative ratios when the burst size ratio is used. Figure 22 shows the receiver operator curves for each metric, which also indicate that the overall detection accuracy is best when maximum cross correlation is used.

From these results, we can observe a clear trade-off between true positive and true negative ratios. However, it is desirable

to have both ratios as close as possible to one. Note that by using any of the metrics alone it is possible to achieve a 0.88 ratio for both true positive and true negative, under the right choice for the threshold.

We also consider the use of multiple metrics to detect Skype-relayed traffic. In order to obtain the best combination among all possible threshold values for the four metrics, we perform a brute-force search over the parameter space. We discretize the threshold values for each metric (as shown in the figures) and compute the true positive and true negative ratios for each combination of thresholds, i.e., using the metrics simultaneously. Using this algorithm we can achieve a 0.96 true positive ratio and 0.96 true negative ratio. The thresholds that yield this performance are: 11 seconds for start time difference, 13 seconds for end time difference, 1.33 for burst size ratio, and 0.38 for maximum cross correlation. Interestingly, each of the thresholds corresponds to the threshold that we need to choose in order to obtain 99% of relays in our controlled experiment II respectively (see Figures 14, 15, 16, and 17).

### D. Lessons and guidelines

Although maximum cross correlation achieves the best results for a single metric in terms of true positive and true negative ratios, better ratios can be obtained if all metrics are used simultaneously. However, this improvement is not very significant. This occurs because the metrics are somewhat correlated. For example, a relayed traffic burst that has large start time and end time differences is likely to have a low maximum cross correlation. It would be interesting to explore how these metrics correlate with one another, as this can guide the development of an even better detection.

We have also noticed that traffic generated by applications other than Skype is sometimes misclassified as being Skype-relayed traffic. By visually inspecting the time series associated with the bursts and looking at port numbers used, we identified several of these applications. A few examples are HTTP proxy, PlanetLab [21] overlay applications, a popular online arcade game (i.e., Kartrider), a Gnutella-variant P2P file sharing application (i.e., Bearshare), and a popular online game running on XBOX. Interestingly, it is possible that some of the traffic identified as Skype-relayed traffic is indeed relayed traffic (albeit not Skype). In particular, some of these false positives have a very high maximum cross-correlation and a burst size ratio near one. We suspected that these burst pairs may indeed be other types of relays. Interested readers are referred to our technical report [22] for the details of our further investigation.

Although our results indicate that we can effectively identify Skype-relayed traffic, the performance of our methodology in accurately detecting Skype-relayed traffic could be in jeopardy as more and more applications start to make use of relays. In particular, it seems that in order to accurately identify the application responsible for generating a particular relayed traffic, application-specific information will have to be used. This observation is supported by the fact that relayed traffic from a given application domain, such as multimedia traffic, will likely have very similar characteristics under the metrics we have defined.

In terms of Skype, there are a few ideas that make use of application-specific information that could improve the accuracy of the classification methodology. For example, we may exploit the way that Skype uses port numbers. Specifically, when UDP is used for both for incoming and outgoing bursts, the destination port number of incoming packets is identical to the source port number of outgoing packets. This seems to be a unique feature of Skype (at least to the extent that we have observed) and could be used to aid application classification of the relayed traffic. A similar behavior, where port numbers of incoming and outgoing packets are the same, sometimes also occurs when TCP and UDP are used for relaying traffic.

### E. Extensions to other applications

As we have observed with some of the false positives, the methodology proposed for detection of Skype-relayed traffic can indeed be used to detect traffic relayed by other applications. In particular, the methodology could be extended to other multimedia applications such as End System Multicast (ESM) [4]. In fact, we have conducted some preliminary controlled experiments using ESM and our results indicate that some metrics are applicable, while others would need to be modified. In ESM, unlike Skype, a relay node is usually interested in the data traffic being relayed (i.e., the user are also consuming the data). The implication of this characteristic is that the start time difference and the end time difference for the ESM-relayed traffic will not be necessarily small. In fact, they can be arbitrarily large. Interested readers are referred to our technical report [22] for an example of ESM trace showing large start and end time differences and a discussion of other more relevant metrics.

## VI. RELATED WORK

Detecting relayed traffic by monitoring an access link is a topic that has received some recent attention, particularly within the context of intrusion detection [7–10]. However, most of the efforts in that context have focused on detecting relayed traffic generated by interactive applications. Therefore, the various assumptions about traffic behavior and consequently their methodology, do not apply to the domain of multimedia applications, as discussed at the end of Section II. More recently, within the context of attack identification, Sekar et al. [10] suggested using the temporal order of flow start times to identify nodes that are being used as relays. The use of temporal order of start times is similar to our idea of using start time differences. However, their work is preliminary and does not characterize or evaluate the effectiveness of detecting relays based on temporal ordering of flows. Moreover, the use of temporal ordering of flows alone will not lead to an effective criteria to detect Skype-relayed traffic (as the false positive ratio will be very high).

A problem similar to the relay detecion problem, known as flow correlation attacks, has also been investigated in the context of mix networks which are used in anonymity preserving

systems [11, 12]. The goal of these studies is to match (or to make it hard to match) a flow arriving at a node with a flow departing the same node. Since the node is purposely trying to defeat the correct identification of in/out pairs, this problem is generally harder. In any case, techniques based on cross-correlation of packet counts have been successfully suggested for identifying relays [12]. However, the assumptions are usually too strict to be directly applied to multimedia traffic relays.

Another related area of work is the classification of network traffic based on the application generating the data. Several methods to classify traffic have been proposed in the literature [6, 23–26]. Traffic classification can help the detection of relayed traffic when the applications that make use of relay nodes are known and can be identified. For example, traffic classification can be performed by profiling application-specific signatures [23, 26]. However, our focus in this work is to develop a technique that does not rely on any application- or protocol-specific information.

## VII. Conclusion

In this paper, we have characterized the nature of Skype-relayed traffic using different metrics. In particular, we have proposed the following metrics: start and end time differences, byte size ratio, and maximum cross correlation between two relayed *bursts of packets*. Using these metrics, we propose a methodology for detection of Skype-relayed traffic based on thresholds. Our approach relies solely on flow-level traffic characteristics, rather than on application- or protocol-specific information.

In order to generate and collect a large amount of relayed traffic, we developed two different controlled experimental environments using Skype. In the first environment, voice traffic generated by two nodes under our control is relayed over some node in the Internet. In the second environment, two real Skype nodes use a relay node under our control to communicate. The data collected was characterized using the metrics we proposed.

We apply our detection methodology to an aggregate packet trace collected at the access point of our university. Our results show that the proposed metrics can be reliably used to detect Skype-relayed traffic, yielding both low false positive and low false negative ratios. Particularly, when using the maximum cross correlation as the sole criteria for detecting Skype-relayed traffic we obtain a 8% false negative and 8% false positive ratios. However, by simultaneously using multiple metrics we can achieve an even higher accuracy. Particularly, we obtain a 4% false negative and 4% false positive ratios when all metrics are used together. Because of such improvements, we argue that multiple criteria should be used when detecting relayed traffic.

As part of on-going work, we are currently evaluating our methodology on other multimedia applications, such as End System Multicast, and other P2P video streaming applications.

## References

[1] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-peer communication across network address translators," in *Proc. of USENIX annual Technical conference*, April 2005.

[2] N. Feamster, D. Andersen, H. Balakrishnan, and F. Kaashoek, "Measuring the effects of Internet path faults on reactive routing," in *Proc. ACM SIGMETRICS*, June 2003.

[3] Skype, http://www.skype.com.

[4] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the Internet using an overlay multicast architecture," in *Proc. ACM SIGCOMM*, August 2001, http://esm.cs.cmu.edu.

[5] J. Oikarinen and D. Reed, "Internet relay chat protocol RFC," 1993.

[6] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel traffic classification in the dark," in *Proc. ACM SIGCOMM*, August 2005.

[7] Y. Zhang and V. Paxson, "Detecting stepping stones," in *Proc. of USENIX security symposium*, August 2000.

[8] D. Donoho, A. Flesia, U. Shankar, V. Paxon, J. Coit, and S. Staniford, "Multiscale stepping-stone detection: detecting pairs of jittered interactive streams by exploiting maximum tolerable delay," in *Proc. of Int'l Symp. on Recent advances in intrusion detection (RAID)*, 2002.

[9] A. Blum, D. Song, and S. Venkataraman, "Detection of interactive stepping stones: algorithms and confidence bounds," in *Proc. of Int'l Symp. recent advance in intrusion detection (RAID)*, 2004.

[10] V. Sekar, Y. Xie, D. Maltz, M. Reiter, and H. Zhang, "Toward a framework for Internet forensic analysis," in *ACM workshop on Hot Topics in Networks (HotNets)*, November 2004.

[11] B. Levine, M. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix systems," in *Proc. of Financial cryptograhy*, 2004.

[12] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "Flow correlation attacks and countermeasures in mix networks," in *Privacy Enhancing Technologies Workshop (PET)*, 2004.

[13] Cisco Netflow, http://www.cisco.com/warp/public/732/Tech/netflow.

[14] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker, "On the characteristics and origins of Internet flow rates," in *Proc. ACM SIGCOMM*, August 2002.

[15] P. Bourke, "Cross correlation," http://astronomy.swin.edu.au/~pbourke/analysis/correlate, August 1996.

[16] S. Baset and H. Schulzrinne, "An analysis of the skype peer-to-peer internet telephony protocol," in *Proc. IEEE INFOCOM*, April 2006.

[17] Dummynet, http://www.dummynet.com.

[18] Tcpdump, http://www.tcpdump.org.

[19] F. Bulk, "Final project:skype," May 2004.

[20] Endace, "Endace network monitoring cards," http://www.endace.com.

[21] Planetlab, http://www.planetlab.org.

[22] K. Suh, D. Figueiredo, J. Kurose, and D. Towsley, "Characterizing and detecting relayed traffic: A case study using skype," University of Massachusetts at Amherst, Tech. Rep. UM-CS-2005-50, 2005.

[23] A. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proc. ACM SIGMETRICS*, June 2005.

[24] A. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Proc. of Passive and Active measurement workshop*, April 2005.

[25] C. Dewes, A. Wichmann, and A. Feldmann, "An analysis of internet chat systems," in *Proc. of ACM Internet Measurement Conference (IMC)*, October 2003.

[26] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification," in *Proc. of ACM Internet Measurement Conference (IMC)*, October 2004.