Efficient Detection of Delay-Constrained Relay Nodes

Baris Coskun
Polytechnic University, ECE Dept.
Six Metrotech Center Brooklyn, NY
baris@isis.poly.edu

Nasir Memon
Polytechnic University, CIS Dept.
Six Metrotech Center Brooklyn, NY
memon@poly.edu

Abstract

Relay nodes are a potential threat to networks since they are used in many malicious situations like stepping stone attacks, botnet communication, peer-to-peer streaming etc. Quick and accurate detection of relay nodes in a network can significantly improve security policy enforcement. There has been significant work done and novel solutions proposed for the problem of identifying relay flows active within a node in the network. However, these solutions require quadratic number of comparisons in the number of flows. In this paper, a related problem of identifying relay nodes is investigated where a relay node is defined as a node in the network that has an active relay flow. The problem is formulated as a variance estimation problem and a statistical approach is proposed for the solution. The proposed solution requires linear time and space in the number of flows and therefore can be employed in large scale implementations. It can be used on its own to identify relay nodes or as a first step in a scalable relay flow detection solution that performs known quadratic time analysis techniques for relay flow detection only on nodes that have been detected as relay nodes. Experimental results show that the proposed scheme is able to detect relay nodes even in the presence of intentional inter-packet delays and chaff packets introduced by adversaries in order to defeat timing based detection algorithms.

1 Introduction

In a typical network, there can be various situations where a node receives data from another node and forwards it to some others. Such nodes are often called "*relay nodes*". Relay nodes can be employed for many different purposes which can be either legitimate or malicious. Routers and switches are clearly examples of usually legitimate relay nodes. But there are other more ambiguous situations. For instance, an enterprise may be running a legitimate peer-to-peer video streaming application for the benefit of its employees. On the other hand an employee could be violating company policy by running a peer-to-peer application to watch live television on the desktop. Similarly, a system administrator may be connected from home to a server and logs in from that server using ssh to one of the internal machines in order to check its status. This would be an example of a legitimate stepping stone connection. Stepping stones, however, are commonly used by hackers to make attack traceback difficult. In general, regardless of the original intention, relay nodes are a potential threat to networks since they are used in many malicious situations like stepping stone attacks, botnet communication, illegal peer-to-peer file sharing etc. Hence, quick and accurate detection of relay nodes in a network can significantly improve security policy enforcement.

Relay nodes can be divided into two main categories:

1) Store & Forward Relay Nodes: These type of relay nodes often store data before forwarding. Peer to peer file sharing and email relaying are some examples of store and forward relays as email relays forward received emails after few minutes and peer to peer file sharing applications don't forward data until another user make a request.

2) Delay-Constrained Relays Nodes: These type of relay nodes forward the received data within a maximum tolerable delay constraint, which is inherent in the underlying application. Delay-constrained relaying can be done by applications which are either interactive or machine driven. For instance, stepping stones and IM message routing nodes are some examples of delay-Constrained relays with interactive sessions. On the other hand, peer to peer live broadcast and Skype super-nodes are examples of machine-driven delay-constrained relays.

Detection of store and forward type of relays is generally done by identifying protocol features. Usually a target protocol is selected and its distinctive characteristics are identified. Subsequently, a node that exhibits such characteristics is declared as a relay node. Some such protocol features used by researchers include connections to known ports, payload signatures, concurrent use of both UDP and TCP etc. This work will mainly focus on delay-constrained relays. Interested readers can find further details on store

and forward relay node detection schemes in [3,5–8].

To the best of our knowledge, there is no prior work that focusses on the delay-constrained relay node detection problem. However, there has been much work on the closely related delay-constrained **relay flow** detection problem [1,2,9,12,13]. Relay flow detection is harder than relay node detection as detection of relay flows implies identification of relay nodes as well. The basic detection methodology in the proposed delay-constrained relay flow detection schemes is to search for network flow pairs which exhibit strong mutual correlation. This correlation is determined based on various attributes of the flow, including packet content (payload), packet arrival times, packet lengths etc. Regardless of how the correlation is determined, all these methods compare each incoming flow to each outgoing one, usually on a node by node basis. Therefore they require quadratic time for each node, which may be prohibitive for medium to large scale networks with tens of thousands of nodes and thousands of active connections in many nodes.

For many scenarios, however, instead of identifying relay flows, identifying relay nodes could be sufficient to take appropriate action. In addition, a lightweight and scalable solution to the problem of delay-constrained relay node detection can serve as a first step in a scalable relay flow detection solution that performs one of the known quadratic time analysis techniques for relay flow detection only on nodes that have been determined to be relay nodes. This strategy brings significant computational efficiency to existing schemes since the quadratic flow detection algorithm is now applied to a few selected nodes rather than every node in the network. Hence, in this work we focus on the problem of delay-constrained relay node detection. This slightly relaxed problem is reformulated as a variance estimation problem and a statistical approach employed to solve the problem in linear time which makes it viable to incorporate in large scale networks. Also the proposed technique is robust, to some extent, against adversarial manipulations that change the time structure of the flows such as intentional delays or chaff packets.

1.1 Related Work

Research on delay constrained relay detection has mostly focused on stepping stones due to their obvious potential malicious intention. Perhaps the first such technique was proposed by Staniford and Heberlein [9]. They proposed a content correlation based scheme where flow pairs are compared in terms of thumb-prints of their content. However, content based schemes have limited applicability since flows are usually encrypted and their contents are inaccessible. This fact motivated researchers to focus on layer 3 information which mostly consists of originating and destination IP addresses, packet arrival times etc. In the first work that incorporates layer 3 information [13], Zhang and Paxson detect stepping stones by correlating flows in terms of their ON and OFF periods. The assumption is that correlated flows switch from OFF state to ON state at similar times. In [12], Yoda and Etoh propose a similar timing based algorithm where correlation is defined over sequence number vs. time curves of the flows. Another timing based algorithm is proposed by He and Tong in [4], where authors formulate the flow correlation problem as a nonparametric hypothesis testing. Other than stepping stones, in [10], Suh *et. al.* proposed a similar timing based technique for detecting Skype related relay traffic.

Timing based methods usually fall short when an attacker perturbs the time structure of the relaying flows by means of introducing artificial delays before relaying the received packet or by adding chaff packets into the stream. In [2], Donoho *et. al.* shows that if there is a maximum tolerable delay constraint, instead of using raw timing information, applying wavelet decomposition and analyzing packet timings in lower resolutions will make the effect of the adversarial changes in time structure insignificant. Similarly under a maximum tolerable delay constraint, Blum *et. al.* present confidence bounds on the stepping stone detection problem in [1]. As a completely different approach, in [11], Wang and Reeves propose a watermarking based approach where selected packet timings are slightly adjusted on all incoming flows. In order to identify a relaying flow, a watermark detection procedure is applied to all outgoing flows.

As we have pointed out before, flow correlation based techniques solve the problem in quadratic time. They need to compare each incoming flow to each outgoing flow. Therefore it is not easy to employ these methods in large networks. One could adopt filtering techniques to alleviate this problem to some extent. For instance, in [13] specific flow pairs are filtered out based on packet size, inconsistent source and destination ports, inconsistent packet direction and timing etc. However, discarding information usually brings a potential threat to detection performance since the real relaying flows could be filtered out or adversaries could manipulate flow characteristics to get filtered out. Therefore, a more scalable solution for relay detection problem would be of potential value in many situations.

2 Detecting Relay Nodes in Linear Time

Before formally defining the problem, we make the following definitions:

-Flow: A flow is the collection of packets which share the common the five-tuple of source IP, source port, destination IP, destination port and layer 3 protocol type (UDP or TCP).

-Incoming/Outgoing Flows: For a particular node, if the destination IP of a flow and the IP of that node are the same, then that flow is considered as an incoming flow. Conversely, if the source IP of the flow is equal to the IP of that node, then the flow is regarded as an outgoing flow.

-Time Slot: We consider time axis as a sequence of equal length time intervals which are called time slots.

-Active Flow: If a flow has at least one packet transmitted within a given time slot, then that flow is regarded as being active within that particular time slot.

2.1 Basic Idea

The basic idea of the proposed technique relies on the fact that the incoming and outgoing components of relay flow have to transmit at least some of the packets (non chaff packets) at similar times (within the maximum tolerable delay duration). That is to say, if the time axis is considered as a sequence of time slots, the corresponding incoming and outgoing flows of the relay would be simultaneously active within some of the time slots. This observation is illustrated in Figure 1, where the incoming flow R and outgoing flow F are acting as a relaying flow pair. It is observed that, in order to forward the received information through flow R, flow F is also active in the same time slots as flow R.

In order to capture this correlation of relaying flows, we assign to each flow a random number drawn from a zero-mean known distribution. Then, for each time slot, the random numbers assigned to active **incoming** flows are summed and multiplied to the sum of random numbers assigned to the active **outgoing** flows. Finally the results of each time slot are added together and an overall sum value is obtained. Calculation of this overall sum (S) is summarized below for the flows shown in Figure 1. Note that the letters A, B, C etc. represent the assigned random numbers to the corresponding flows shown Figure 1.

$$\begin{aligned} S &= (R+A)(F+B) + C(D+B) + \cdots \\ &\qquad \cdots + (A+R)F + AD + RF \\ &= RF + RB + AF + AB + CD + \cdots \\ &\qquad \cdots + CB + AF + RF + AD + RF \\ &= 3 \times RF + (RB + AF + AB + \cdots \\ &\qquad \cdots + CD + CB + AF + AD) \\ &= 3 \times RF + sum(\text{Random Numbers}) \end{aligned}$$

It is observed above that, thanks to distributive property of multiplication over addition, this calculation is effectively equivalent to assigning a random number to each active flow pair and then summing them up. Hence, if there were no correlated relaying flows, "S" would be the sum of random numbers coming from a zero-mean known distribution. Consequently "S" itself would be a random number coming from another zero-mean known distribution. However, when there's a relay activity and the random number assigned to the relaying flow pair appears multiple times in the summation and somewhat can be considered as a constant term, which changes the governing distribution of "S". Consequently one can classify a node as being a relay or not based on a simple statistical test applied on "S". The mathematical details of this scheme are presented in Section 2.2.

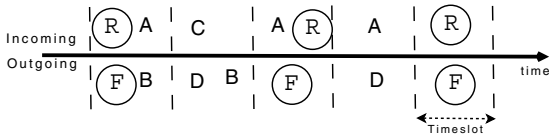


Figure 1. Active flows in a short period of time, where the outgoing flow F relays information coming from the incoming flow R.

At this point, an acute reader would have observed that request-response based protocols (i.e. TCP) pose a problem against the proposed technique. This is because flows which carry requests and corresponding flows which carry responses are often active within the same time slots. Consequently, network nodes which use such protocols would be automatically declared as relay nodes by the scheme we have described above. But there is a simple fix to this problem. Essentially, only one flow of such flow pairs should contribute to the summation process for each time slot and the other flow should simply be ignored. This solution can easily be implemented by checking if a random number has already been accumulated for a flow which has the exact reverse direction (i.e source and destination IP addresses and ports are swapped) of a given flow. If the answer is yes, that flow is simply ignored since its counterpart flow has already been taken into account.

2.2 Problem Formulation and Solution

Our goal is to differentiate between network traffic data which contains relay activity and that which does not. The basic difference between these two situations is the number of time slots in which the same flow pair is simultaneously active. That is because, although uncorrelated flows may share few time slots but it is very unlikely that the number of such time slots is high. On the other hand, for a flow pair relaying information with a maximum tolerable delay constraint, if one flow is active for a given time slot, with high probability the other flow is active in order to relay information before the time constraint elapses. We denote the number of time slots shared by the same flow pair as β, which we will show very closely related to the variance of the sum S ((σ_S^2)) introduced in Section 2.1. Therefore one can distinguish relay nodes from the corresponding (σ_S^2) .

In order to state the relationship between β and σ_S^2, we define I_i and O_j for a given node in the network as the incoming flows the outgoing flows respectively, where $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$. For each incoming flow and each outgoing flow, a random number is generated and assigned to the corresponding flow. Let RI_i and RO_j denote the random numbers assigned to the incoming flow I_i and the outgoing flow O_j respectively. Here RI_i and RO_j are assigned such that they are independently drawn from the probability mass function $P(n)$, which is:

$$P(n) = \begin{cases} \frac{1}{2}, & if\ n = +\gamma \\ \frac{1}{2}, & if\ n = -\gamma \\ 0 & , elsewhere \end{cases}, \quad \text{where } \gamma \in \Re \qquad (1)$$

The reason $P(n)$ is chosen as a bipolar symmetric PMF is that our detection algorithm requires the distribution of $RI_i \times RO_j$ to be bipolar symmetric. This distribution turns out to be bipolar symmetric as long as $P(n)$ itself is chosen as bipolar symmetric.

Meanwhile, let $\hat{i}_t$ and $\hat{j}_t$ be the indices of the active incoming flows and the active outgoing flows respectively for a given time slot t. Then for each time slot t, the corresponding random numbers for active incoming flows within t are summed and multiplied to the sum of the random numbers assigned to active outgoing flows. This step is repeated for each time slot and result of each step is accumulated. More formally, the following summation S is calculated for time slots $t = 1, 2, 3, ..., T$:

$$S = \sum_{t=1}^{T} \left[\sum_{i \in \hat{i}_t} RI_i \times \sum_{j \in \hat{j}_t} RO_j \right] \qquad (2)$$

We can rewrite Equation (2) from distributive property as:

$$S = \sum_{t=1}^{T} \left[\sum_{i \in \hat{i}_t} \sum_{j \in \hat{j}_t} RI_i \times RO_j \right] \qquad (3)$$

Notice that S is effectively the summation of a new set of random numbers which are assigned to each active incoming-outgoing flow pair for each time slot. Therefore, Equation (3) can be written as:

$$S = \sum_{a=1}^{A} M_a \qquad (4)$$

where A is the number of terms such that "$A = \sum_{t=1}^{T} \sum_{i \in \hat{i}_t} \sum_{j \in \hat{j}_t} 1$" and "$M_a = (RI_i \times RO_j)$", which are random numbers drawn from probability mass function $\hat{P}(n)$:

$$\hat{P}(n) = \begin{cases} \frac{1}{2} & , n = +\gamma^2 \\ \frac{1}{2} & , n = -\gamma^2 \\ 0 & , elsewhere \end{cases}, \quad \text{where } \gamma \in \Re \qquad (5)$$

Rigorously speaking, the M_a values may not be independent since a single RI_i or a single RO_j may, and probably will, contribute to multiple M_a values. However, for practical purposes M_a values are considered as i.i.d. random variables with probability mass function $\hat{P}(n)$. As a matter of fact, this is not a very crude assumption[1].

[1]Suppose $M_1 = RI_i \times RO_j$ and $M_2 = RI_i \times RO_k$. Since $P(n)$ is bipolar symmetric, knowing RI_i doesn't give any information about $RI_i \times RO_j$. In other words $P(M_1 \mid M_2) = P(M_1)$ and therefore M_1 and M_2 are independent. However, suppose the following values were also given or known: $M_3 = RI_m \times RO_j$ and $M_4 = RI_m \times RO_k$. Then, by using M_3 and M_4 one could determine the polarity relation of RO_j and RO_k such that, if $M_3 = M_4$ then $RO_j = RO_k$ as well and if $M_3 \neq M_4$ then $RO_j \neq RO_k$. This information could later be used to determine M_1 such that if $RO_j = RO_k$ then $M_1 = M_2$ and if $RO_j \neq RO_k$ then $M_1 = -M_2$. So, M_1 is not independent from the collection (M_2, M_3, M_4). However, in order this to happen, flow I_i has to be active in the same time slot with O_j and in another or the same time slot with O_k. Also I_m should be active in the same time slot with O_j and in another or the same time slot with O_k. And all this has to happen in a short period of time before "A" terms are collected. The probability of such event is not very high and therefore independent M_a assumption is quite realistic.

Assuming independent M_a values, if there is no correlation between any pair of incoming and outgoing flows, in other words if there is no relaying flow pairs, then Equation (4) holds and S will simply be interpreted as the sum of independent random variables. On the other hand if y^{th} outgoing flow (RO_y) relays the packets from x^{th} incoming flow (RI_x), both RI_x and RO_y will be active in a number of time slots (β), then the sum in Equation (4) can be written as:

$$S = \beta\,(RI_x \times RO_y) + \sum_{a=1}^{A-\beta} M_a, \quad \text{where } \beta \geq 0 \qquad (6)$$

More generally if there are F such incoming-outgoing flow pairs, namely I_{f_i} and O_{f_o}, where f_i and f_o indicate incoming and outgoing flow indices of f^{th} pair respectively $(f = 1, 2, ..F)$, and each of these pairs are simultaneously active within β_f time slots, then the summation S can be written as:

$$S = \sum_{f=1}^{F} \beta_f M_f + \sum_{a=1}^{A-\beta} M_a \qquad (7)$$

where $M_f = (RI_{f_i} \times RO_{f_o})$, $\beta = \sum_{f=1}^{F} \beta_f$ and $\beta_f \geq 0$ for all f. Notice that Equation (7) reduces to Equation (6) when $F = 1$ indicating a single relaying input-output flow pair and further reduces to Equation (4) when $F = 0$ indicating no relay.

The value S is a random variable in both Equations (4) and (7) and variance of S (σ_S^2) can be used to identify relay activity. In order to show this, let $\sigma^2_{S\_noRelay}$ represents the variance of S where there's no relay activity. We can write $\sigma^2_{S\_noRelay} = \gamma^4 A$ from Equation (4), since M_a values are i.i.d random variables drawn from probability distribution given in Equation(5) whose variance is γ^4. On the other hand, $\sigma^2_{S\_withRelay}$, which represents the variance of S under relay traffic, can be similarly calculated from Equation (7) as:

$$\begin{aligned} \sigma^2_{S\_withRelay} &= \textstyle\sum_{f=1}^{F} (\beta_f)^2 \gamma^4 + (A - \beta)\gamma^4 \\ &= \gamma^4 \left(\textstyle\sum_{f=1}^{F} (\beta_f)^2 - \beta \right) + \gamma^4 A \\ &= \gamma^4 \left(\textstyle\sum_{f=1}^{F} (\beta_f)^2 - \beta \right) + \sigma^2_{S\_noRelay} \end{aligned} \qquad (8)$$

Since $\beta = \sum_{f=1}^{F} \beta_f$, "$\sigma^2_{S\_withRelay} > \sigma^2_{S\_noRelay}$" is satisfied so long as $\sum_{f=1}^{F} (\beta_f)^2 > \sum_{f=1}^{F} \beta_f$, which holds for

$\beta_f > 1$. Therefore one can identify relay nodes by looking at σ_S^2 values as long as the relay flows are simultaneously active for more than one time slot. This is a purely theoretic constraint and in practice a relay has to be active for sufficiently enough number of time slots in order to be detected. Fortunately, most of the relay scenarios have to satisfy this constraint in order to serve their purpose. Another interesting observation is that the above constraint is independent of γ. Therefore, the system performance doesn't change for different values of γ. Therefore, in the experiments this value is simply set to $\gamma = 1$.

In practice, estimating σ_S^2 is the first task to be performed. Then the system can declare a node as being a relay if the estimated σ_S^2 is sufficiently larger than the anticipated σ_S^2 if the node is not relaying at all. For this purpose, the system computes a number of S values simultaneously but independently in order to estimate the σ_S^2 value.

2.3 Detection Algorithm and the Analysis

In this section we first discuss the proposed algorithms which compute S values in order to estimate σ_S^2 and consequently performs relay detection. Following that, space and time requirements of these algorithms are discusses.

The S values are calculated by the algorithm, $Calculate\_S$ listed below, which basically performs the operation defined in Equation (2). The algorithm takes the parameter A, which is introduced in Equation (4) and indicates the number of terms added together. Here I and O indicate the list of incoming and outgoing flows respectively. Also the function "*Reverse()*" returns the flow identification which has the complete reverse direction of a given flow. The algorithm then checks if this reverse flow has been included in the summation previously in order to deal with the problem of request-response based protocols mentioned earlier in Section 2.1.

Notice that $Calculate\_S$ computes a single S value. In order to accurately estimate σ_S^2, a number of S values have to be collected. This can be done by executing multiple $Calculate\_S$ instances simultaneously. The algorithm, $Estimate\_\sigma_S^2$ listed below, implements the estimation procedure by employing the algorithm $Calculate\_S$. The input parameter T indicates the number of simultaneous $Calculate\_S$ executions.

It should be noted that $Estimate\_\sigma_S^2$ outputs a single σ_S^2 value whenever A terms are collected. The time elapsed until A terms are collected totally depends on the input parameters and the characteristics of the underlying network traffic. In a typical scenario the parameters can be chosen so that the elapsed time to collect A terms is around few seconds. Therefore, each estimated σ_S^2 value would correspond to a few seconds of network traffic. For continuous operation, $Estimate\_\sigma_S^2$ should be executed repeatedly.

The final decision is given based on the difference between estimated σ_S^2 values and the anticipated σ_S^2 value when the node is not relaying at all. This value can be written more formally as:

$$\Psi = \sigma_S^2 - \gamma^4 A \tag{9}$$

Algorithm 1 $Calculate\_S(A, I = \{I_1, ., I_m\}, O = \{O_1., O_n\})$

```
S ← 0
noOfTerms ← 0
timeSlot ← currentTimeSlot
while noOfTerms ≤ A do
    incomingSum ← 0
    actInFl ← 0
    for all active incoming flow I_i within timeSlot do
        if RO_i for Reverse(I_i) is already used timeSlot then
            continue;  //In order to avoid request-response problem ignore this flow
        end if
        if RI_i is not assigned to I_i then
            assign RI_i randomly  [as in Eq(1)]
        end if
        incomingSum ← incomingSum + RI_i
        actInFl ← actInFl + 1
    end for
    outgoingSum ← 0
    actOutFl ← 0
    for all active incoming flow O_j within timeSlot do
        if RI_j for Reverse(O_j) is already used in timeSlot then
            continue;  //In order to avoid request-response problem ignore this flow
        end if
        if RO_j is not assigned to O_j then
            assign RO_j randomly  [as in Eq(1)]
        end if
        outgoingSum ← outgoingSum + RO_i
        actOutFl ← actOutFl + 1
    end for
    noOfTerms ← noOfTerms + actInFl × actOutFl
    timeSlot ← nextTimeSlot
    S ← S + incomingSum × outgoingSum
end while
output S
```

In the experiments, a recent few Ψ values are incorporated in the decision process such that if the sum of "d" most recently calculated Ψ values exceeds a threshold "th" then the corresponding host is declared as performing relay activity within the corresponding time slice.

Before we begin to analyze the algorithm, it should be emphasized that the parameters A and T are constant values which are in the order of few hundreds and they do not vary with the input size. As for the investigation of time requirements, the algorithm $Calculate\_S$ loops over the active incoming flows and active outgoing flows separately. It repeats these loops a constant number of times until A terms are collected for summation. Therefore Calculate$S$ is $O(m+n)$ time algorithm where m and n are the number of incoming and outgoing flows respectively of the node being analyzed. Practically speaking, the algorithm may actually loop fewer times than $m+n$ since only a fraction of incoming and outgoing flows are active for a given time slot.

For the second algorithm, $Estimate\_\sigma_S^2$ calls

$Calculate\_S$ exactly T times which is indeed a constant parameter. Therefore, $Estimate\_\sigma_S^2$ too runs in $O(m+n)$ time. However, the time requirements of the whole system actually depends on how many times the algorithm $Estimate\_\sigma_S^2$ is executed. But again for a given fixed time interval, $Estimate\_\sigma_S^2$ is called repeatedly for a constant number of times. Therefore for a given fixed time interval, which is typically the duration of a typical relay activity, the decision is given in linear time.

The space required by the algorithm, on the other hand, is mainly the table which keeps assigned random values of the incoming and outgoing flows. Hence, it can be concluded that the algorithm requires linear space as well.

Algorithm 2 Estimate_$\sigma_S^2(A, T, I, O)$

$\sigma_S^2 \leftarrow 0$
for $i = 1$ to T **do**
  $S_i \leftarrow Calculate\_S(A, I, O)$
  $\sigma_S^2 \leftarrow \sigma_S^2 + \frac{(S_i)^2}{T}$
end for
output σ_S^2

3 Experiments and Results

In order to verify the practical efficacy of the proposed scheme, the algorithms discussed in Section 2.3 were implemented and executed for real network traffic. This section presents the experimental setup and their results in order to demonstrate the performance of the proposed scheme.

3.1 Experimental Setup

The ultimate goal of the proposed scheme is to identify network nodes that perform relay activity, or in other words, "*relay nodes*". Aside from relay traffic, in almost every case, relay nodes also receive and transmit legitimate non-relay traffic. Therefore, in our experiments, traffic for relay nodes was constructed such that pure relay traffic is blended into non-relay host traffic. For the non-relay traffic, two different types of traffic data were captured from real network. The first type was the traffic of our university's web server which basically consists of http flows. The second type was captured from a mail server, which provides mail client connections and ssh/telnet interactive sessions. Both traffic data were captured on a typical day for a few hours. The Web server's traffic had average packet rate of 70 packets/second and average bit rate of 416 kbps whereas mail server's traffic had higher average packer rate of 76 packets/second but lower average bit rate of 257 kbps.

The relay traffic on the other hand was artificially generated where the packet inter-arrival times were determined by the following Gaussian mixture model:

$$P(i) = p\mathcal{N}(\mu_{short}, \sigma_{short}^2) + (1-p)\mathcal{N}(\mu_{long}, \sigma_{long}^2) \quad (10)$$

where, $\mathcal{N}(\mu, \sigma^2)$ indicates normal distribution with mean μ and variance σ^2.

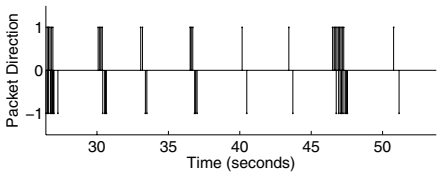
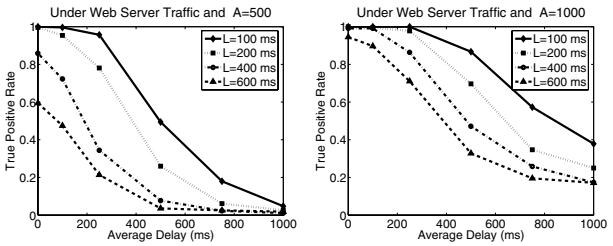


Figure 2. A typical relay traffic generated by model in Equation (10).

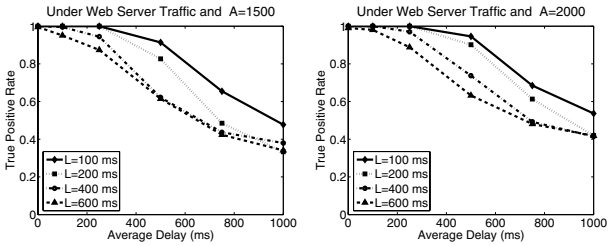
This model generates bursty traffic such that the packets are mostly sent back to back without waiting too much (with probability p) and a pause period occurs once in a while (with probability $1-p$). The reason this model was used is that bursty traffic captures the behavior of most applications more accurately. Notice that this model generates only the incoming portion of the relay traffic. Each received packet has to be forwarded in order to obtain a complete relay activity. Rather than forwarding packets immediately, a certain amount of delay was introduced for each packet in order to represent the packet processing time and/or intentional adversarial delays. For each packet, delay values were chosen randomly from a normal distribution with μ_{delay} and σ_{delay}^2. About 30 second portion of an example relay traffic generated by this model is given in Figure 2, which shows the incoming packets, inter-arrival times, and introduced delay between incoming and outgoing packets. The parameters used for the relay traffic in this figure are $p = 0.8$, $\mu_{short} = 100$ milliseconds, $\sigma_{short} = 10$, $\mu_{long} = 3000$ milliseconds and $\sigma_{long} = 500$. The introduced delay parameters are $\mu_{delay} = 400$ milliseconds and $\sigma_{delay} = 50$. Also in this figure, positive bars indicate incoming packets where negative ones indicate corresponding forwarded (relayed) packets.

Given network traffic data, the system was required to decide if there is relay activity or not. The decision is given based on the Ψ value described in Section 2.3. As discussed in that section, the system calculates a Ψ value each time A terms are collected. In our experiments, the decision in favor of the presence of a relay activity was made if the sum of the most recent two Ψ values exceeded the threshold value $th = 1000$. The numbers 2 and 1000 are selected experimentally and they are tuned to detect shorter relay activities which last only for a few hundred packets. However, it should be noted that further reducing th value would enable detecting even shorter relay activity but would incur the cost of increased false positive rates. On the other hand, large th values could reduce false positive rates to arbitrarily small values but the system can detect only long enough relay activity.

In order to measure detection performance, the generated relay traffic was blended into real network traffic and the overall traffic was fed to the system for analysis. If the system was able to detect the relay activity by the time all

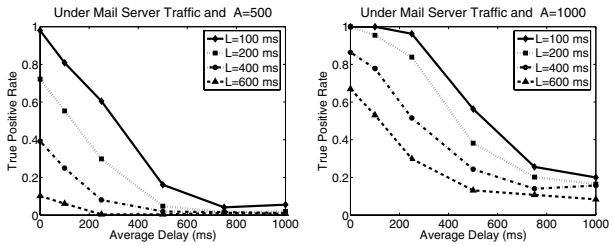


(a) A=500 (b) A=1000

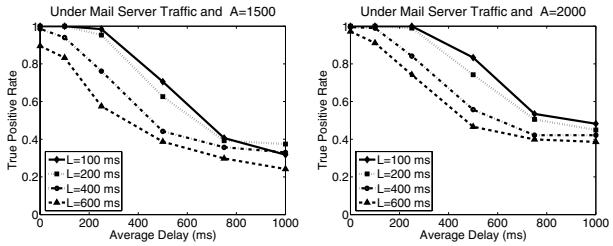


(c) A=1500 (d) A=2000

Figure 3. True positive rate vs. average delay curves under Web Server traffic.



(a) A=500 (b) A=1000



(c) A=1500 (d) A=2000

Figure 4. True positive rate vs. average delay curves under Mail Server traffic.

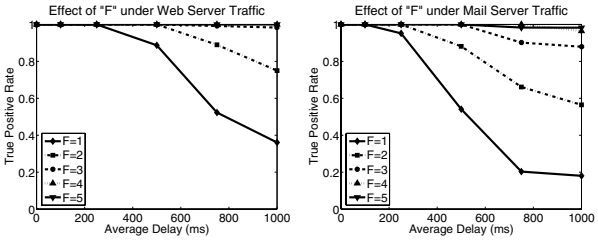
relay packets are forwarded, then the number of true positives was incremented. This process was repeated several times and at the end, the ratio of the true positives to the number of all generated relay traffic was declared as the true positive rate. The same experiment was then repeated once more but this time without blending the relay traffic into the real traffic. Hence, if the system declares a relay activity then it means a false positive and the false positive counter was incremented. Finally the ratio of the number of false positives to the number of all generated but not blended relay traffic gives the false positive rate.

3.2 Results

In this section the results of the experiments described in the previous section are discussed. The experiments were conducted using various time slot lengths (L), and various (A) values which indicates the number of terms incorporated as discussed in Section 2.3 in Equation (4). The performance was also investigated for various average delay values that relay packets encounter. In the first set of experiments we used only one pair of relaying flows (i.e. $F = 1$ in Equation (7)) and relay traffic consisted of 200 incoming packets and the corresponding 200 outgoing relayed packets. The inter-arrival times between incoming relay packets were generated according to the model in Equation (10) where $p = 0.8$, $\mu_{short} = 100$ milliseconds, $\sigma_{short} = 10$, $\mu_{long} = 3000$ milliseconds and $\sigma_{long} = 500$. A delay value was drawn from $\mathcal{N}(\mu_{delay}, \sigma^2_{delay})$ between each incoming packet and the corresponding forwarded packet. Here μ_{delay} was changed throughout the experiments but $\sigma_{delay} = 50$ was kept fixed. Finally the resulting relay traffic, which typically lasts for about 150-160 seconds, was blended into the captured real traffic and true positive rate was calculated.

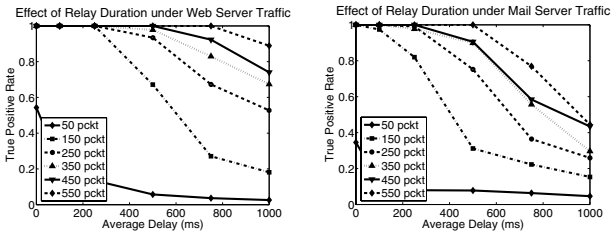
Figure 3 and Figure 4 plot the true positive rates vs. average delay (μ_{delay}) of the relay traffic where the relay traffic is blended into web server traffic and mail server traffic respectively. The experiment was repeated for parameter values $A = 500, 1000, 1500, 2000$ and $L = 100, 200, 400, 600$ milliseconds. Each of the four sub-figures corresponds to a different A value whereas each sub-figure contains four different curves each represents a different L value.

In all experiments, regardless of the A and L values, it is observed that the true positive rate decreases as the average delay of the relay is increased. This behavior is expected since increased delay reduces the probability of relaying flows being active simultaneously in the same time slot. In other words, as the delay increases, it is more likely that the time slot during which the incoming packet is received, elapses before the node forwards that packet. However, another interesting behavior was also observed in all experiments. Even though the average delay value was way greater than the length of the time slots (L), some of the relay activities were still successfully detected. For instance when the time slot length (L) was 100 milliseconds and the average delay was 600 milliseconds, there is no way that the incoming packets are relayed within the same time slot. Therefore, at first sight it can be said that the system cannot detect these relays although it certainly can as shown in Figures 3 and 4. The reason for this behavior is that sometimes relay packets corresponding to previous incoming packets are transmitted within the same time slot that a new packet arrives. Therefore, although the same content isn't being relayed at the same time slot, both incoming and outgoing



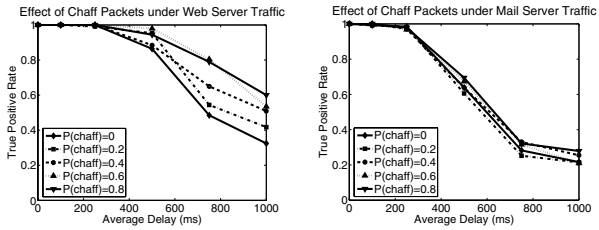
(a) Under Web Server Traffic (b) Under Mail Server Traffic

Figure 5. True positive rate vs. average delay curves for different "F" values.



(a) Under Web Server Traffic (b) Under Mail Server Traffic

Figure 6. True positive rate vs. average delay curves for different number of relayed packets.



(a) Under Web Server Traffic (b) Under Mail Server Traffic

Figure 7. True positive rate vs. average delay curves for different chaff packet probabilities.

relay flows still may simultaneously be active. This event contributes to the increase in calculated Ψ value and enables the system to detect some of these relays.

It is also observed that in all experiments, detection performance decreases as the length of the time slots increases. This is little bit counterintuitive since one may expect to capture relay activity more accurately with longer time slots. However, in reality longer time slots means more flows (relay or non-relay) are active within the same time slot and therefore more random number terms added to the overall sum as in the algorithm $Calculate\_S$. Therefore, the system collects "A" terms more rapidly, and hence is forced to make a decision earlier. This sometimes prevents relay flows from injecting sufficient number of packets in order to be detected before the system makes decision. Consequently some of the relay activities are left undetected with larger L values.

Table 1. False positive rates

	UNDER WEB SERVER TRAFFIC			
	$\mathbf{L = 100}$	$\mathbf{L = 200}$	$\mathbf{L = 400}$	$\mathbf{L = 600}$
$\mathbf{A = 500}$	0.0088	0.0085	0.0020	0.0020
$\mathbf{A = 1000}$	0.1257	0.0904	0.0441	0.0553
$\mathbf{A = 1500}$	0.1890	0.2022	0.1610	0.2024
$\mathbf{A = 2000}$	0.2842	0.3077	0.2353	0.2554
	UNDER MAIL SERVER TRAFFIC			
	$\mathbf{L = 100}$	$\mathbf{L = 200}$	$\mathbf{L = 400}$	$\mathbf{L = 600}$
$\mathbf{A = 500}$	0.0352	0.0067	0.0066	0.0043
$\mathbf{A = 1000}$	0.1337	0.0807	0.0702	0.0920
$\mathbf{A = 1500}$	0.1886	0.1839	0.1990	0.1827
$\mathbf{A = 2000}$	0.2873	0.2557	0.2615	0.2657

On the other hand, increasing the "A" value certainly delays the time that the system has to make a decision. Therefore, relay flows will have enough time to inject sufficient number of packets to be detected. This behavior can be observed as an increased detection performance as we go from Figure 3(a) to Figure 3(d) and from Figure 4(a) to Figure 4(d). However, this increase in the detection performance comes at the cost of increased false positive rates as can be seen in Table 1. This is primarily due to the fact that the number of time slots which any two arbitrary flows are simultaneously active within, increases as A is increased. This is a simple fact from probability theory that if the size of the universal set increases, the number of event occurrences increases as well, as long as the frequency of that event remains constant.

When we look at the results presented in Figure 3, Figure 4, Table 1, we observe that some settings of the parameters lead to quite unsatisfactory results. For instance, in the case when $A = 500$ and $L = 600$, the true positive rate is too low. Therefore, the parameters A and L should not be set to these values although they are included in the figures to demonstrate the effect of changing parameters. Similarly when $A = 2000$, even though true positive rates are significantly higher, this setting shouldn't be used due to high false positive rates. However, if the proposed algorithm were to be used as an initial step for relay flow detection as discussed earlier, setting $A = 2000$ may not be a bad idea. Because, in that case the nodes that are flagged by the proposed algorithm would be further analyzed by a relay flow detection algorithm in order to identify relay flows. High false positive rates do increase the number of flagged nodes but this computational inefficiency may still be better as compared to the case where the proposed scheme is not used at all as the initial step. In the rest of the experiments, we set our algorithm parameters as $A = 1000$ and $L = 100$ as they lead to reasonable true positive and false positive rates. However, for real deployments, some care will need to be taken before setting algorithm parameters. Finally, it

was observed for all parameter values that the performance for the mail server traffic was slightly worse than the performance for web server traffic. The reason is that the mail server traffic had higher average packet rates than the web server traffic. Similar to the previous observations, the more the packet rate, the traffic obtains the more active flows fall within a time slot and hence the earlier the "A" terms are collected. Therefore, the system has to make a decision earlier under heavier traffic and consequently it may miss some relay activities. In order to minimize this effect, the system parameters should be carefully chosen according to the expected traffic characteristics. In the experiments identical parameter settings are used for both web server and mail server traffic for comparison purposes.

3.2.1 Effect of Multiple Flow Pairs

In the experiments described above, a single pair of flows perform the relaying activity. However, it is possible that a relay node could host multiple relay activities simultaneously. The number of such relay activities is denoted by "F" in Section 2.2. In this section the performance of the system was investigated for different values of F. In order to demonstrate the performance, for a specific set of parameters such as $A = 1000$ and $L = 100$, the system is required to detect the presence of a relay activity when there are multiple relaying flow pairs. As shown in Figure 5, the true positive rate increases drastically as the number of relaying flow pairs increases. In fact the relay activity is detected almost with 100% accuracy when there are more than four relaying flow pairs. This result is not surprising because higher values of F also increases the variance (σ_S^2) as observed in Equation (8).

3.2.2 Effect Relay Duration

In section 2.2, it was mentioned that theoretically relay activity is detected if corresponding flows are simultaneously active within more than one time slot. It was also mentioned that this was a purely theoretical conclusion and in practice sufficiently many packets have to be relayed in order to be detected. This section investigates the detection performance for various relay durations and gives insight about the minimum detectable relay duration. Here the duration of a relay is measured as the number of relayed packets. In the experiment, where the parameters are chosen as $A = 1000$ and $L = 100$, relay packets were generated by the same model described in Section 3.1. The true positive rate vs. average delay curves for different number of relayed packets are presented in Figure 6. As expected, it is observed that the detection performance increases as the number of relayed packets increases. This is because, as the number of relayed packets increases the number of time slots, within which the corresponding flows are active, also increases. This exactly corresponds to the increase in β_f values mentioned in Equation (7) and consequently increase in σ_S^2 and Ψ.

On the other hand, it is also observed in Figure 6 that the proposed scheme has low accuracy in detecting relays that last for less than about 50-60 packets for the given underlying traffic characteristics. Here it should be noted that the underlying web server traffic and mail server traffic used in the experiments have average packet rates of 70 packets/sec and 76 packets/sec respectively. Since the average packet rate of the mail server traffic is higher, when the curves for mail server and web server traffic are compared, it is expected that, the minimum detectable relay duration gets lower and lower as the non-relay traffic of the node gets lighter.

3.2.3 Effect of Chaff Packets

In order to disturb the correlation structure between relaying incoming and outgoing flows, adversaries often blend chaff packets into the relay stream. This enables an adversary to relay information without being detected by flow-correlation based relay detection algorithms. The chaff packets usually carry no useful information. They can be placed in the incoming flow and not relayed through the outgoing flow. They can also be generated by the relay node and placed in the outgoing flow. In both cases the purpose is to generate a packet in a flow which has no counterpart in the corresponding flow. However, regardless of the chaff packets, some of the packets (actual relay packets) still have to be relayed within a certain time period in order the relay node to serve its purpose. Those packets will still make the incoming and outgoing flows simultaneously active within a number of time slots. Hence, the proposed scheme will be able to detect the relay activity. Therefore it can be said that chaff packets have virtually no effect on the proposed method. This property is clearly observed in the experiments whose results are presented in Figure 7. In the experiments, the system parameters are set to $A = 1000$ and $L = 100$. After each relayed packet a chaff packet is generated with probability $P(\text{chaff}) = 0, 0.2, 0.4, 0.6,$ and 0.8. These chaff packets take place only in incoming flow and are not relayed through the outgoing flow. The inter-arrival times for these chaff packets are determined by the same model which determines the inter-arrival times of regular incoming relay packets as described in Section 3.1. It is observed that none of these $P(\text{chaff})$ values have decreased the detection performance at all. On the contrary, as the number of chaff packets is increased, the detection performance is slightly improved especially for larger average delay values. This is because these extra chaff packets in the incoming flow sometimes coincides with other packets in the outgoing flow and therefore increase the number of time slots that both flows are simultaneously active.

4 Conclusion

Due to their potential harmful effects, identifying relay nodes in the network can improve security policy enforcement. In this work, the delay constrained relay node detection problem is investigated. A statistical solution, which has linear time and space complexity, is proposed. The proposed algorithm is lightweight and simple, therefore it is scalable and can be used in large scale implementations which may require real time detection.

For some applications identifying relay nodes may be sufficient. If an application requires flow level relay identification, one of the existing relay flow detection techniques can be subsequently applied to the relay nodes which have been flagged by the proposed method. The contribution of this work is then in terms of computational complexity, as quadratic time relay flow detection algorithms now have to be executed only for flagged nodes rather than every node in the network.

Experimental results show that the proposed scheme is robust against various possible adversarial or non-adversarial modifications on the underlying network traffic. In summary, the experiments reveal that the proposed scheme withstands some extent of packet delays which could be introduced due to packet processing time or for adversarial purposes. Also the algorithm is shown to be able to detect relay activity even if the flows contain chaff packets intended to defeat relay detection systems.

There are few limitations of the proposed scheme. First of all relay nodes should be delay-constrained. That is to say, if incoming packets were buffered long enough before they are forwarded, the algorithm would not be able to detect. Also it is assumed that the flows are relatively sparse such that they are active for some of the time slots and inactive for others. Otherwise, if a flow were continuously active within all observed time slots, then it would appear as a relaying flow since would be simultaneously active with all the other flows. In the current setup of the proposed scheme, if a node contains such flow, then it will be detected as a relay node.

There is a lot of work that still needs to be done. As part of our future work effort, we plan to focus on methods which can increase the detection performance such that higher true positives rates and lower false positive rates can be achieved. One possible way to achieve this could be using a different alignment of time slots for each of the simultaneous S calculations (i.e. time slot boundaries are not aligned for each simultaneous $Calculate\_S$ execution). This could enable some of the calculated S values to catch relay activity which the others might have missed. Also an adaptive selection of algorithm parameters, which adjust the parameters according to encountered traffic characteristics, might be very useful especially when the traffic characteristics tend to change over time. This could be done by sensing and analyzing the ongoing traffic and reacting accordingly. Finally, we also plan to deploy our algorithm in a large scale implementation such that, in order to experimentally verify its ability to monitor thousands of network nodes and detect relay activity in real time.

References

[1] D. S. A. Blum and S. Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In *Conference of Recent Advance in Intrusion Detection (RAID), Sophia Antipolis, French Riviera, France*, September 2004.

[2] D. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford. Multiscale stepping-stone detection: Detecting pairs of jittered interactive streams by exploiting maximum tolerable delay. In *Fifth International Symposium on Recent Advances in Intrusion Detection, Lecture Notes in Computer Science 2516, New York, Springer*, 2002.

[3] A. Gerber, J. Houle, H. Nguyen, M. Roughan, and S. Sen. P2p the gorilla in the cable. In *National Cable and Telecommunications Association (NCTA) 2003 National Show, Chicago, IL*, June 2003.

[4] T. He and L. Tong. A signal processing perspective of stepping-stone detection. In *Proc. of IEEE CISS'06, Princeton, NJ*, 2006.

[5] T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy. Transport layer identification of p2p traffic. In *Proc. 4th ACM SIGCOMM Conf. on Internet Measurement, Taormina, Sicily, Italy,*, October 2004.

[6] R. Meent and A. Pras. Assessing unknown network traffic. In *CTIT Technical Report 04-11, University of Twente, Netherlands*, February 2004.

[7] S. Ohzahata, Y. Hagiwara, M. Terada, and K. Kawashima. Aa traffic identification method and evaluations for a pure p2p application. In *Lecture Notes in Computer Science*, volume 3431, 2005.

[8] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of p2p traffic using application signatures. In *Proc. 13th Int. Conf. on World Wide Web, NY*, 2004.

[9] S. Staniford-Chen and L. Heberlein. Holding intruders accountable on the internet. In *Proc. IEEE Symposium on Security and Privacy, Oakland, CA*, page 3949, May 1995.

[10] K. Suh, D. Figueiredo, J. Kurose, and D. Towsley. Characterizing and detecting skype-relayed traffic. In *Proc. of Infocom*, 2006.

[11] X. Wang and D. S. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 20–29, 2003.

[12] K. Yoda and H. Etoh. Finding a connection chain for tracing intruders. In *F. Guppens, Y. Deswarte, D. Gollamann, and M. Waidner, editors, 6th European Symposisum on Research in Computer Security - ESORICS 2000 LNCS -1985, Toulouse, France*, October 2000.

[13] Y. Zhang and V. Paxson. Detecting stepping stones. In *Proceedings of the 9th USENIX Security Symposium*, page 171184, August 2000.