Dependent Link Padding Algorithms for Low Latency Anonymity Systems

Wei Wang Mehul Motani
Department of Electrical & Computer Engineering
National University of Singapore, Singapore
{wang.wei,motani}@nus.edu.sg

Vikram Srinivasan
Bell Labs Research, India
Bangalore, India
vikramsr@alcatel-lucent.com

ABSTRACT

Low latency anonymity systems are susceptive to traffic analysis attacks. In this paper, we propose a dependent link padding scheme to protect anonymity systems from traffic analysis attacks while providing a strict delay bound. The covering traffic generated by our scheme uses the minimum sending rate to provide full anonymity for a given set of flows. The relationship between user anonymity and the minimum covering traffic rate is then studied via analysis and simulation. When user flows are Poisson processes with the same sending rate, the minimum covering traffic rate to provide full anonymity to m users is $O(\log m)$. For Pareto traffic, we show that the rate of the covering traffic converges to a constant when the number of flows goes to infinity. Finally, we use real Internet trace files to study the behavior of our algorithm when user flows have different rates.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General – Security and protection

General Terms

Security, Algorithms, Performance

Keywords

Anonymity systems, Link padding, Traffic analysis attack

1. INTRODUCTION

With the convergence and emergence of all IP networks, anonymous communication is an extremely important issue which needs to be addressed. The aim of anonymous communication is to allow users to communicate with each other while not revealing the identity of the sender and the receiver to a third party.

The concept of Mix networks was first proposed by Chaum to provide anonymity in mailing systems [1]. To conceal the identity of messages, Mix servers encrypt and pad messages to make them indistinguishable in content and message length. However, we now know that uniform message content and length are insufficient to preserve anonymity, since correlations in the timing of packets arriving and departing from the Mix servers can also be used to discern user identity [2, 3]. One way to remove the correlations of packet transmission times is to collect packets from different users and store them in a pool before flushing them in a batch [1, 4]. However, this pooling process can cause significant delay which may be intolerable for network applications other than the mailing system.

Low-latency anonymity networks, such as Onion Routing [5], Tor [6] and Tarzan [7], try to provide anonymous communication over the Internet with limited packet delay to meet the requirements of interactive web applications. Due to the delay constraints, most low-latency anonymity systems do not intentionally perturb the packet transmission time at the output of an anonymous server. Thus, they are weak against traffic analysis attacks such as packet counting [8] or traffic timing correlation attacks [2]. In traffic analysis attacks, the adversary uses the packet timing information, such as Inter-Packet Delays (IPDs), to identify a particular packet flow. It is shown in [2, 9, 3, 10] that the packet timing carries significant information which cannot be removed even if the system introduce random delays to every packet.

One method to protect the user's anonymity from traffic analysis attacks is via manipulating the sending time of outgoing packets. The server can send packets according to a fixed schedule which is independent of the original packet timing in the incoming flow [11]. Consequently, the packet timing information of the original flow will be fully removed after passing through the anonymity system. To maintain the traffic pattern of the outgoing flow, a dummy packet will be sent by the server when there is no packet to send at the scheduled time. We refer to such link padding algorithms as *independent link padding* schemes in the later discussions. Independent link padding schemes can use either a constant rate padding which sends packets at a constant rate [12] or a random padding which sends packets according to a randomly generated schedule such as the Poisson process [11].

There are two limitations in independent link padding schemes. First, the independently generated schedule imposes additional limitations on the sending rate of servers in the anonymity system. Some packets may experience long delays before they can be sent at the scheduled time. When there is a strict packet delay constraint, these packets will be dropped by the servers [11]. In this case, the through-

CCS'08, October 27–31, 2008, Alexandria, Virginia, USA.

put of the anonymity network will be reduced due to the rate constraints of the independent sending schedule. Second, independent padding schemes need to *know* the average sending rate of the incoming flow to work efficiently. When the padding rate is too high, most of the bandwidth will be wasted on dummy packets. On the other hand, many incoming packets will be dropped when the padding rate is too low. Furthermore, it is hard to adapt the padding rate when the traffic is time-varying. Dynamically changing the padding rate according to the incoming rate could give the adversary a clue to correlate the rate changes in the incoming flow and the outgoing flow.

In this paper, we consider *dependent link padding* algorithms which produce sending schedules based on the packet timing of incoming flows [13]. We show that full anonymity can be achieved via manipulating the outgoing packet timing to make all flows looks similar. Compared to independent link padding algorithms, dependent link padding algorithms provide strict delay bounds with no packet drops and can dynamically adapt to the traffic rate changes in incoming flows. We show that, independent of the packet arrival distribution, our dependent link padding uses the minimum transmission rate to provide maximum anonymity when packets are not dropped. Moreover, we can use our algorithm to investigate the relationship between anonymity and sending rate for different arrival distributions. When flows are independent Poisson processes, the minimum sending rate is $O(\log m)$ for an anonymity system to provide full anonymity for m user flows. For Pareto flows, we show that the sending rate will converge to a constant when the number of flows goes to infinity.

We also propose a heuristic dropping algorithm to restrict the sending rate when the number of user flows is extremely large. Our simulation results show that the packet drop rate of the heuristic algorithm is smaller compared to independent padding schemes.

The rest of this paper is organized as follows. Section 2 summarizes related works in link padding schemes. In section 3 we introduce possible attacks in low latency anonymity systems. We then give the dependent link padding algorithm and show its rate optimality in section 4. The algorithm with heuristic packet dropping is studied and compared with the independent link padding schemes in section 5. Simulation results in simulated Pareto traffic and real-world trace files are shown in section 6. Finally, section 7 concludes the whole paper.

2. RELATED WORK

Low latency anonymity systems have been widely used in anonymous communication over public networks [6, 14, 5, 7]. However, the user anonymity of these systems may be compromised by traffic analysis attacks as shown in [8, 2, 3, 10]. Such traffic analysis attacks can be either passive or active. A passive adversary can monitor the traffic pattern of incoming flows and correlate it with output flows even if the output flow is mixed with other traffic [2]. Researches in [15] show that the sampled data from Internet exchanges can be used for Bayesian traffic analysis. Moreover, passive adversaries can use the end-to-end delay of packets to infer the route taken by the packets [16, 17]. Active adversaries can intentionally introduce certain patterns in user traffic in order to identify user flows at the output of the anonymity system [3, 10]. The flow identity can be encoded in watermarks via slightly changing the packet timing of a flow. Such watermarks cannot be removed by the random delay introduced by the network or the anonymity system [3, 10].

Independent link padding can be used to protect anonymity systems from traffic analysis attacks [11, 12]. Independent link padding schemes uses a predefined schedule to send the user flow. When there are no user packets in the flow, the anonymity system will send dummy packets to pad the link. As most link padding schemes require all output links to send at a constant rate, they are not flexible enough to traffic changes. Moreover, it is shown in [18] that links padded by constant rate schedule are still susceptive to traffic analysis as the variance of packet timing may be correlated to the system loading.

Dependent link padding schemes can be used to overcome these problems in independent link padding schemes. Adaptive padding schemes [19] uses dummy packets to repair statistically unlikely gaps in the flow so that the time "fingerprints" in the traffic can be destroyed. However, adaptive padding do not fully remove the timing correlation between the incoming and outgoing flows. Thus, it is still susceptive to long-term traffic analysis attacks [19]. While submitting this work, we found Venkitasubramaniam *et al.* also proposed a dependent link padding algorithm which is similar to our algorithm [13]. However, their analysis are mostly focused on the average delay of the algorithm while we prove the rate optimality of the algorithm. We also provide more analysis and simulations on the rate-anonymity relationship in dependent link padding systems.

3. SYSTEM MODEL

3.1 Anonymity System with Latency Constraints

We consider an anonymous server which has m incoming flows of $f_i, i = 1, \ldots, m$ and n outgoing flows of $o_i, i = 1, \ldots, n$, as in Fig. 1.

We assume that the adversary can monitor all packets in the incoming and outgoing flows of the anonymous server. However, the adversary is not able to correlate the flow by packet content or packet length since the packet contents are encrypted and packets all have uniform length. A passive adversary is allowed to record any traffic timing information in order to infer the relationship between incoming flows and outgoing flows. We also allow active adversaries to manipulate the packet timing of the incoming flows.

To measure the anonymity of the system, we adopt the information theoretic definition used in [20, 21]. The anonymity of outgoing flow o_i is defined as the information entropy of the identity of the corresponding incoming flow. When all the m incoming flows have the same probability to be the source of o_i, the anonymity will be $\log m$, which is the maximum anonymity that can be achieved.

The anonymous server aims to achieve maximum anonymity for all flows by manipulating the sending time of outgoing flows. In other words, the packet timing of incoming and outgoing flows should not leak any information about the relationship between incoming and outgoing flows.

As we wish to support interactive web applications, there are some additional requirements on the forwarding protocols of the server. First, the delay incurred by the server should be strictly smaller than Δ for all packets. Also, all packets in the incoming flow must be sent out in the same outgoing flow. We also prefer that the anonymous server do

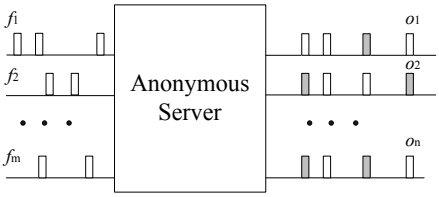

Figure 1: An anonymous server with m incoming flows and n output flows.

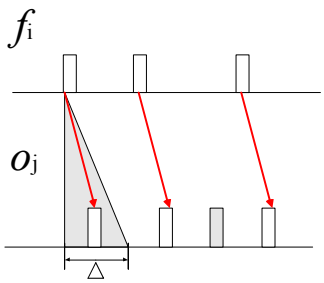

Figure 2: Matching packets in incoming and outgoing flows, grey packets are dummy packets introduced by the anonymity system.

not drop message packets from incoming flows. In this case, when an incoming flow f_i is sent to outgoing flow o_j, every packet in the flow f_i should appear within time Δ in the flow o_j. We will relax this requirement in later discussions and allow the anonymous server drop a small fraction of the incoming packets.

3.2 Matching Attack

In the matching attack, the adversary tries to find out the relationship between the incoming flow and outgoing flow by matching the packets in different flows.

Consider an incoming flow f_i which is sent to o_j by the anonymous server. It is easy to see that there must exist a way to match every packet in f_i exclusively to one packet in o_j while the delay between the two matched packets should be within $[0, \Delta]$, see Fig. 2. If we cannot find such a matching between f_i and o_j, then f_i cannot be the source of o_j when the anonymous server is not dropping any packets. We define an output schedule o_j to be a *matched schedule* for f_i, if we can find a way to match every packet in f_i exclusively to one packet in o_j within delay of Δ.

Hall's matching theorem [22] gives the necessary and sufficient condition for o_j to be a matched schedule for f_i as follows.

An outgoing flow o_j is a matched schedule for f_i if and only if

$$N_{f_i}(t') - N_{f_i}(t) \leq N_{o_j}(t' + \Delta) - N_{o_j}(t) \quad (1)$$

for any time interval of $[t, t'], t' \geq t$, where $N_{f_i}(t)$ is the total number of packets arrived at flow f_i until time t.

In other words, the number of packets arrived during $[t, t']$ at flow f_i must be smaller than or equal to the number of packets sent by flow o_j during $[t, t' + \Delta]$, for all t and t'.

The Bounded Greedy Match (BGM) algorithm [23] provides an efficient way to find out whether o_j is a matched schedule for f_i or not [24]. This algorithm tries to match packets in f_i to the first unmatched packet in o_j. The outgoing link o_j can no longer be a matched schedule for f_i once a packet in f_i cannot be matched in BGM. BGM is efficient in ruling out flows that are not matched. It takes only a small number of packets to find an unmatched packet when o_j and f_i are independent Poisson arrival processes [23].

If the adversary finds out that a outgoing flow o_j is not a matched schedule for a incoming flow f_i, the size of candidate outgoing flow set that f_i can use will be reduced and user anonymity will be compromised accordingly. Therefore, if we wish to provide full anonymity of $\log m$ to users, o_j should be a matched schedule for all input flows.

Dependent Link Padding Algorithm
Parameters: Packet arrival time t_{ij} for all flows $f_i \in \mathcal{F}$
Output: A matched schedule $S(\mathcal{F})$ for all flows $f_i \in \mathcal{F}$
01: Take a new packet P_{ij} according to the arrival sequence.
02: **if** there is an unused token with $t_s \geq t_{ij}$ for f_i
03: Schedule P_{ij} at t_s
04: Mark the token as used for f_i
05: **else**
06: Add a new token at $t'_s = t_{ij} + \Delta$ in $S(\mathcal{F})$, which can be used by all flows in $\mathcal{F}$
07: Schedule P_{ij} at time t'_s and mark the token as used for f_i.
08: **endif**
09: Go to step 01 until no more packet arrives.

Figure 3: Dependent Link Padding Algorithm for a given flow set $\mathcal{F}$.

4. DEPENDENT PADDING ALGORITHM

To protect user flows from a matching attack, we introduce the Dependent Link Padding (DLP) algorithm in this section. The DLP algorithm uses the packet arrival timing information to generate a matched schedule with the minimum sending rate for a given set of incoming flows.

4.1 DLP Algorithm

Consider a set of incoming flows $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$. The DLP algorithm takes the packet timing of flows in $\mathcal{F}$ as the input and outputs a schedule $S(\mathcal{F})$ which is a matched schedule for all flows in $\mathcal{F}$.

We denote the jth packet in flow f_i as P_{ij} and its arrival time as t_{ij}. The *schedule* $S(\mathcal{F})$ contains a sequence of sending time $t_s, s = 1, 2, \ldots$ at which the server sends either one message packet or one dummy packet on the outgoing link. In the rest of this paper, we call each sending opportunity in $S(\mathcal{F})$ a *token*. We define the sending rate for $S(\mathcal{F})$ as

$$R_{S(\mathcal{F})} = \lim_{s \to \infty} \frac{s}{t_s}. \quad (2)$$

When a new packet in f_i arrives, the DLP algorithm first checks whether the packet can be sent with an existing token at time t_s which satisfies $t_{ij} \leq t_s \leq t_{ij} + \Delta$. If such a token exists, P_{ij} will be scheduled at t_s. Otherwise, we will add a new token at time $t'_s = t_{ij} + \Delta$ and send P_{ij} at t'_s. The new token is marked as unused for other flows. If a packet P_{kl} in another flow f_l arrives after P_{ij} while before $t_{ij} + \Delta$, it can be scheduled at t'_s. The detail of the DLP algorithm is shown in Fig. 3.

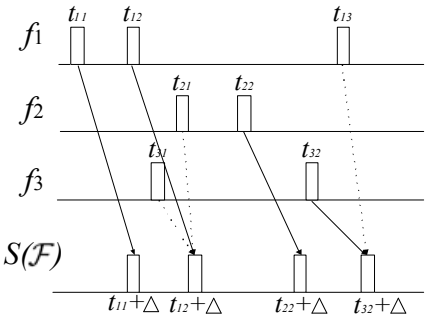

Figure 4: Using DLP to generate $S(\mathcal{F})$.

It is easy to see that every packet P_{ij} in f_i will be sent in $S(\mathcal{F})$ at some time t_s which satisfies $t_{ij} \leq t_s \leq t_{ij} + \Delta$, see Fig. 4. This holds true for all $f_i \in \mathcal{F}$, so $S(\mathcal{F})$ is a matched schedule for all incoming flows in $\mathcal{F}$. The algorithm is an online algorithm as the scheduled time t_s can be determined by packet arrival times before $t_s - \Delta$.

For each token, the associated operation in DLP is to put it in $|\mathcal{F}|$ FIFO queues when it is generated and remove it from the queue when it is used. So, the total number of operation used for a token is $O(|\mathcal{F}|)$ where $|\mathcal{F}|$ is the number of incoming flows in $\mathcal{F}$. The amortized computational cost for each token is therefore $O(|\mathcal{F}|)$.

4.2 Using the Matched Schedule

A simple way to use the matched schedule in the anonymity system is to send packets according to $S(\mathcal{F})$ at all outgoing links, where $\mathcal{F}$ includes all incoming flows. When sending f_i though outgoing link o_j, the server will send a message packet from f_i at the scheduled time t_s or send a dummy packet when the token at t_s is not used. In this way, all outgoing links will have exactly the same packet timing which is a matched schedule for all incoming links. Therefore, any incoming flow will have equal probability to be sent through any outgoing link and maximum anonymity is achieved, see Fig. 1.

In constant link padding schemes, the anonymous servers needs to constantly send packets over every output link to hide traffic information such as traffic pattern, start and ending time of a flow, *etc.* When using the DLP algorithm, the server will send no packets when there are no incoming flows. As more incoming flows are relayed by the server, the server can gradually increase the sending rate at all outgoing links and adapt to the rate changes in incoming flows. The DLP algorithm is efficient to conceal the start and ending times of flows, as all the n output links "start" transmitting at the same time.

The matched schedule can also be used to achieve k-anonymity by grouping incoming flows to groups of size k. The system can use the matched schedule of k incoming flows on k different outgoing links to provide k-anonymity for incoming flows in the same group. When flows with similar traffic patterns are grouped together, the matched schedule may send fewer dummy packets for each flow and the efficiency of the padding scheme can be improved.

The DLP algorithm is not only useful to protect the system from the matching attack. It can also make all outgoing traffic have the same traffic pattern so that the traffic correlation [2] or watermarking attacks [3, 10] cannot compromise user privacy by comparing traffic patterns of incoming and outgoing traffic.

In practice, it may not be possible to send out one packet at each outgoing link at exactly the same time due to the limitations on bandwidth or the processing capacity of the server. In this case, we can send the packets scheduled at the same time in a sequence with random or deterministic order. This would also make all outgoing flows have statistically the same packet timings.

4.3 Optimality of DLP

THEOREM 1. *The schedule $S(\mathcal{F})$ generated by DLP uses the minimum number of tokens among all matched schedules of the incoming flow set of $\mathcal{F}$.*

PROOF. We use induction to prove the optimality of DLP algorithm. First, denote the packets in the union of all incoming flows according to their arrival time as $P^{(1)}, P^{(2)}, \ldots$. For example, we have $P^{(1)} = P_{11}$, $P^{(2)} = P_{12}$ and $P^{(3)} = P_{31}$ in Fig. 4. Therefore, we have $t^{(1)} \leq t^{(2)} \leq t^{(3)} \ldots$. It is easy to see that we only need to consider the number of tokens used by the algorithm at the time just after a packet arrives. Between two packet arrivals, the minimum number of tokens used in the algorithm will remain the same.

The DLP algorithm is optimal at time $t^{(1)}$ when $P^{(1)}$ arrives. This is because it only schedules one token at $t^{(1)} + \Delta$ and any matched schedule for the flow set $\mathcal{F}$ must schedule at least one token when $P^{(1)}$ arrives.

Suppose that our algorithm is optimal at time $t^{(k)}$ for all $k \leq N-1$ and our algorithm uses T tokens at time $t^{(N-1)}$. When the packet $P^{(N)}$ arrives, we have three different cases:

Case 1: DLP finds an unused token for packet $P^{(N)}$

In this case, the number of tokens used by DLP is not increased. As DLP is optimal at time $t^{(N-1)}$, the algorithm remains optimal for $t^{(N)}$ as the the number of token used by any schedule is non-decreasing when a new packet arrives.

Case 2: DLP cannot find an unused token for packet $P^{(N)}$ and $t^{(N)} > t^{(N-1)} + \Delta$

In this case, DLP algorithm uses $T+1$ tokens at time $t^{(N)}$. Now consider the number of tokens used by other schedules. If a schedule have scheduled tokens after $t^{(N-1)} + \Delta$ at time $t^{(N-1)}$, then the token cannot carry any packet arrived before $P^{(N-1)}$ (including $P^{(N-1)}$) due to the delay constraint of Δ. Therefore, the token scheduled after $t^{(N-1)} + \Delta$ is redundant at time $t^{(N-1)}$ and such schedule algorithm cannot be optimal at time $t^{(N-1)}$. As the optimal schedule uses T tokens at $t^{(N-1)}$, a suboptimal schedule must use at least $T+1$ tokens at $t^{(N-1)}$ and at time $t^{(N)}$.

As a schedule which is optimal at $t^{(N-1)}$ cannot schedule tokens after $t^{(N-1)} + \Delta$ at time $t^{(N-1)}$ and packet $P^{(N)}$ can only be sent after $t^{(N)} > t^{(N-1)} + \Delta$, these optimal schedules need to add at least one more token at time $t^{(N)}$. As the minimum number of tokens used at time $t^{(N-1)}$ is T, the minimum number of tokens used at time $t^{(N)}$ is $T+1$ which equals to the number of tokens used in DLP algorithm.

Case 3: DLP cannot find an unused token for packet $P^{(N)}$ and $t^{(N)} \leq t^{(N-1)} + \Delta$

Without loss of generality, we assume packet $P^{(N)}$ belongs to input flow f_i and $P^{(N)} = P_{ij}$.

If DLP sends no dummy packets for flow f_i, i.e., every

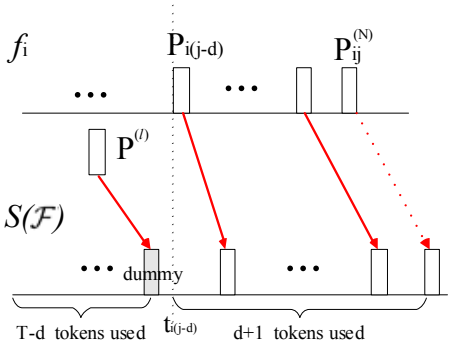

Figure 5: Matching packets for case 3.

token in the schedule is matched to one packet in f_i, then f_i has at least T packets at time $t^{(N-1)}$. As flow f_i has $T+1$ packets at time $t^{(N)}$, the minimum number of token used for any schedule at $t^{(N)}$ should be $T+1$, which is equal to the number of tokens used in DLP.

If DLP sends at least one dummy packet for f_i, we can find a packet $P_{i(j-d)}$ in f_i which is matched to the token just after the latest dummy token, see Fig. 4. The dummy token must have a time $t_s < t_{i(j-d)}$, otherwise DLP will match $P_{i(j-d)}$ to the dummy token. Since the dummy token has $t_s < t_{i(j-d)}$, it must be scheduled when a packet, say $P^{(l)}$, arrives at $t^{(l)} \leq t_{i(j-d)} - \Delta$.

Now we have divided the time of $[0, t^{(N)}]$ to two non-overlapping parts of $[0, t^{(l)}]$ and $[t_{i(j-d)}, t^{(N)}]$. It is easy to see that the number of tokens used by DLP is $T-d$ at time $t^{(l)}$, since d tokens are matched to $P_{i(j-d)}, P_{i(j-d+1)}, \ldots, P_{i(j-1)}$ which arrives in $[t_{i(j-d)}, t^{(N-1)}]$ and no dummy packet is send for f_i in this period. As the DLP algorithm is optimal at time $t^{(l)}$, any schedule should use at least $T-d$ tokens at time $t^{(l)}$. Similar to the arguments in case 2, if a token is scheduled after time $t^{(l)}+\Delta$, it will be considered as redundant at time $t^{(l)}$. So, all the $T-d$ tokens should have $t_s \leq t^{(l)} + \Delta < t_{i(j-d)}$ and they cannot be used for packets arrived after or at $t_{i(j-d)}$.

There are $d+1$ packets $P_{i(j-d)}, P_{i(j-d+1)}, \ldots, P_{ij}$ arrives in flow f_i during $[t_{i(j-d)}, t^{(N)}]$. Using Hall's matching theorem, any schedule should have at least $d+1$ tokens in the duration of $[t_{i(j-d)}, t^{(N)}+\Delta]$. In other words, at least $d+1$ tokens should be scheduled after $t_{i(j-d)}$ at $t^{(N)}$. As DLP do not have dummy tokens for f_i after $t_{i(j-d)}$, DLP uses exactly $d+1$ tokens during $[t_{i(j-d)}, t^{(N)}+\Delta]$.

Therefore, the necessary number of token for $[0, t^{(l)}]$ and $[t_{i(j-d)}, t^{(N)}]$ is $T-d$ and $d+1$ accordingly. Also, the tokens in the one of the two time period cannot be used to send packet in the other period. So, any schedule should at least use $T+1$ tokens at time $t^{(N)}$.

In summary, DLP uses T tokens at time $t^{(N-1)}$. It remains optimal at time $t^{(N)}$ if an unused token can be found. Otherwise, DLP will use $T+1$ transmissions at time $t^{(N)}$, while all other schedules should also use at least $T+1$ transmissions in this case. So, if DLP algorithm is optimal at time $t^{(k)}, k \leq N-1$, it remains optimal at time $t^{(N)}$. Along with the optimality at $t^{(1)}$, DLP is optimal over all schedules at any time instance. □

4.4 Sending Rates for Poisson Traffic

The schedule generated by DLP will use a higher outgoing rate than individual incoming flow rates to protect flow anonymity. When the outgoing link uses the matching schedule to "carry" the incoming flow f_i, dummy packets will be sent on outgoing flows for which the corresponding incoming flows have no packets. As shown in theorem 1, the DLP algorithm sends the least number of dummy packets among all algorithms which can provide full anonymity with a strict delay bound. Therefore, the sending rate of $S(\mathcal{F})$ can be used to study the relationship between the sending rate of the covering traffic and the anonymity provided by the covering traffic.

In this section, we provide analysis in the scenario where there are m independent incoming flows and each of them has a Poisson arrival rate of λ. More general traffic models will be studied via simulation in section 6.

THEOREM 2. *For a flow set $\mathcal{F}$ which contains m flow with independent Poisson arrival rates of λ, the schedule generated by DLP will have an average sending rate $R_{S(\mathcal{F})}$ of $O(\log m)$ as $m \to \infty$.*

PROOF. First, we divide time in to slots with equal length of Δ. In this case, the lth time slot will be the time period of $[(l-1)\Delta, l\Delta]$.

Suppose the maximum number of packets arriving in a single incoming flow during the $(l-1)$th time slot $[(l-2)\Delta, (l-1)\Delta]$ is k. Consider a new schedule which use the same schedule as DLP during $[0, (l-2)\Delta]$ and schedules k tokens at time $(l-1)\Delta$. For any packet P_{ij} arriving in the $(l-1)$th time slot, we have $t_{ij} \leq (l-1)\Delta \leq t_{ij}+\Delta$. This means the k tokens can be used for any packets arriving in the $(l-1)$th time slot. Therefore, the new schedule is a matched schedule for all flows during $[0, (l-1)\Delta]$. By theorem 1, DLP is optimal at time $(l-1)\Delta$ so that it will schedule at most k tokens for packets arrived during $[(l-2)\Delta, (l-1)\Delta]$. By the DLP algorithm, a token with $t_s \in [(l-1)\Delta, l\Delta]$ can only be scheduled when a packet arrives at some incoming flow during $[(l-2)\Delta, (l-1)\Delta]$. So, the DLP algorithm will at most use k tokens in the lth time slot $[(l-1)\Delta, l\Delta]$.

As all flows have independent Poisson arrivals, the number of packets arriving during $[(l-1)\Delta, l\Delta]$ for a single flow will be a Poisson random variable with mean $\lambda\Delta$. Similar to the well-know Balls in Bins problem [25, 26], we can show that the maximum of m independent Poisson variables is $O(\log m)$ with high probability, when $\lambda\Delta$ is constant. Therefore, the number of tokens scheduled by DLP in $[(l-1)\Delta, l\Delta]$ will be at most $O(\log m)$. Averaging over all time slots, the rate used by DLP is at most $O(\log m)$. □

Theorem 2 shows the asymptotic sending rate of DLP when the number of incoming flow is large. Interestingly, the sending rate $R_{S(\mathcal{F})}$ is closely related to the anonymity provided by the schedule. As the flow set $\mathcal{F}$ contains more incoming flows, the schedule generated by DLP provides a better anonymity of $\log m$ when $|\mathcal{F}| = m$. To achieve the increased anonymity, the sending rate of the schedule is increased as $O(\log m)$ accordingly. Fig. 6 shows the experimental sending rate curve with different values of m. We see that the sending rate is actually increasing at a speed of $O(\log m)$.

The sending rate $R_{S(\mathcal{F})}$ also depends on the arrival rate λ of incoming flows and the delay constraint Δ. As these two

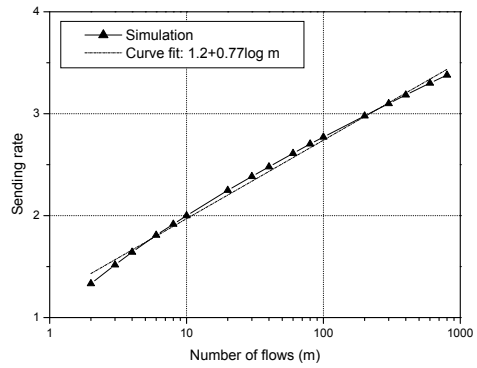

Figure 6: Relationship between the sending rate and the number of flows ($\lambda = 1$, $\Delta = 1$).

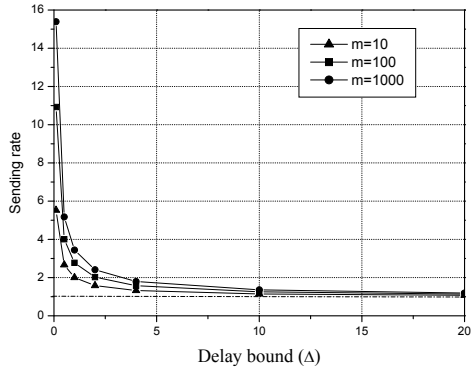

Figure 7: Relationship between the sending rate and the delay bound Δ ($\lambda = 1$).

parameters both depend on the time unit chosen by us, it is enough to only consider the value of $\lambda\Delta$, the average number of packets arriving at a flow during the delay bound. The rate used by DLP can be obtained by properly scaling the result with λ. For example, if the DLP algorithm takes two Poisson flows with rate of λ, we will have $R_{S(\mathcal{F})} = \frac{2(\lambda\Delta+1)\lambda}{2\lambda\Delta+1}$ as shown in [13].

Fig. 7 shows the sending rate of DLP with different Δ when λ is normalized to one. We see that the rate used by DLP converges to 1 when the delay constraint increases. In other words, DLP does not introduce too many dummy packets when the system has a large $\lambda\Delta$ value. This is because as $\lambda\Delta$ increases, the number of packets arriving in one slot for every flow will be within $[(1-\varepsilon)\lambda\Delta, (1+\varepsilon)\lambda\Delta]$ with high probability for arbitrarily small ε by the law of large numbers [27].

4.5 Multi-hop DLP

Many anonymity systems uses anonymity networks instead of a single anonymous server. User traffic may go through several servers to achieve a better anonymity. The DLP algorithm can also work for anonymity networks with more than one server.

Consider an anonymous server which can take both original user flows f_i and traffic from other servers which are already scheduled by DLP as $S(\mathcal{F}_i)$. Since the DLP algorithm will merely delay the packets by Δ without changing the traffic pattern when it takes only one flow as the input, an original user flow f_i can also be treated as a DLP schedule with a flow set $\mathcal{F}_i$ which has only one element of f_i. Therefore, we only need to consider the server which takes $S(\mathcal{F}_i)$ as the incoming flows.

Suppose the anonymous server has incoming flow set of $\mathcal{S} = \{S(\mathcal{F}_1), \ldots, S(\mathcal{F}_m)\}$ and generates a DLP schedule of $S(\mathcal{S})$. In this case, the anonymous server only has the packet timing of the schedule $S(\mathcal{F}_i)$ generated by anonymous servers in previous hops while not knowing the original user flows in $\mathcal{F}_i$. A natural question to ask is whether the lack of information of original flows will deteriorate the performance of the DLP algorithm. We compare the performance of multi-hop DLP system with a single hop server which takes incoming flow set of $\mathcal{F} = \bigcup_{i=1}^{m} \mathcal{F}_i$. The single hop server knows the packet timing of all the original user flows and generates DLP schedule of $S(\mathcal{F})$.

As it is difficult to directly compare the rate of $S(\mathcal{S})$ and $S(\mathcal{F})$, we define a schedule $S^{\Delta}(\mathcal{F})$ which a delayed version of the DLP schedule $S(\mathcal{F})$ where the delay is Δ. We can see that $R_{S^{\Delta}(\mathcal{F})} = R_{S(\mathcal{F})}$ and we have the following lemma.

LEMMA 1. *$S^{\Delta}(\mathcal{F})$ is a matched schedule for any $S(\mathcal{F}_i)$ generated by DLP if the flow set $\mathcal{F}_i \subseteq \mathcal{F}$.*

PROOF. Suppose $S^{\Delta}(\mathcal{F})$ is not a matched schedule for $S(\mathcal{F}_i)$. By Hall's matching theorem, there must exist a time period $[t, t']$ where the number of tokens used by $S(\mathcal{F}_i)$, which is $N_{S(\mathcal{F}_i)}(t') - N_{S(\mathcal{F}_i)}(t)$, is more than the number of tokens used by $S^{\Delta}(\mathcal{F})$ during $[t, t' + \Delta]$. Consequently, the number of tokens used by $S(\mathcal{F})$ during $[t - \Delta, t']$ will be smaller than $N_{S(\mathcal{F}_i)}(t') - N_{S(\mathcal{F}_i)}(t)$ as $S^{\Delta}(\mathcal{F})$ is a delayed version of $S(\mathcal{F})$.

As $S(\mathcal{F})$ is a matched schedule for all $f_j \in \mathcal{F}$, any packet in $f_j \in \mathcal{F}_i$ arrived during $[t - \Delta, t' - \Delta]$ can find an exclusive match to the tokens in $S(\mathcal{F})$ during $[t - \Delta, t']$. Therefore, if we remove all tokens scheduled by $S(\mathcal{F}_i)$ during $[t, t']$ and replace them with tokens in $S(\mathcal{F})$ on the period $[t - \Delta, t']$, the number of tokens used by $S(\mathcal{F}_i)$ at time $t' - \Delta$ will be reduced while we still can find a matching for all packets in flow f_j arrived before and at $t' - \Delta$. This is a contradiction to the proof of Theorem 1, where we show that $S(\mathcal{F}_i)$ should be optimal at time $t' - \Delta$. □

By lemma 1, we see that $S^{\Delta}(\mathcal{F})$ is a matched schedule for all the incoming flows in $\mathcal{S} = \{S(\mathcal{F}_1), \ldots, S(\mathcal{F}_m)\}$. Thus, the performance of DLP can be bounded by the following theorem.

THEOREM 3. *If the incoming flow set is $\mathcal{S} = \{S(\mathcal{F}_1), S(\mathcal{F}_2), \ldots, S(\mathcal{F}_m)\}$, the DLP algorithm will generate a schedule $S(\mathcal{S})$ which has no more than $|S(\mathcal{F})|$ tokens, where $S(\mathcal{F})$ is the schedule generated by DLP on the flow set $\mathcal{F} = \bigcup_{i=1}^{m} \mathcal{F}_i$.*

PROOF. By Theorem 1, the schedule $S(\mathcal{S})$ uses minimum number of tokens among all schedules which are matched schedule for $S(\mathcal{F}_i), i = 1, \ldots, m$. As shown by lemma 1, $S^{\Delta}(\mathcal{F})$ is a matched schedule for all $S(\mathcal{F}_i)$. Therefore the number of tokens used by $S(\mathcal{S})$ is no more than $|S^{\Delta}(\mathcal{F})|$ which is equal to $|S(\mathcal{F})|$. □

Theorem 3 shows that when the DLP is executed on several cascading anonymous servers, the final schedule will use an average sending rate no larger than treating all anonymous servers as a single one. Note that the cost we pay here is an increase in the delay bound. The overall delay bound will be increased to $k\Delta$ when the flow is passed through k anonymous servers. If a single anonymity sever is used for all incoming flows and have a delay constraint of $k\Delta$, the resulting schedule could be more efficient than the cascading case.

This property of the DLP algorithm gives more flexibility when organizing the anonymity networks. Consider an anonymity network which has k first hop servers each taking m incoming flows. The first hop server can provide anonymity of $\log m$ by using the DLP schedule. When the number of m is too small to meet the anonymity requirements, we can use a second hop server which take the flows come from the k first hop servers and use DLP to schedule them again. The resulting schedule will provide an increased anonymity of $\log(km)$, since the outgoing link of the second hop server could come from any incoming flow from any first hop server. By Theorem 3, the rate use by the second hop server will be smaller or equal to the case that it actually knows the incoming packet timing of all flows in every first hop server. Therefore, when all incoming flows have independent Poisson arrivals, the rate used by the second hop server will be $O(\log(km))$ instead of $O(\log k \log m)$. Such anonymity network organization can be extended to cases where there are more than two hops. Also, the anonymous servers can mix flows with different rate and mix original flows with DLP schedules.

5. COMPARISON WITH INDEPENDENT PADDING ALGORITHMS

In this section, we compare the DLP algorithm to independent link padding algorithms, such as constant rate link padding [12] or exponential link padding [28]. In these algorithms, the system generates an independent sending schedule and sends packets at the scheduled time. Like dependent link padding, the system also sends dummy packets when there are no message packets to be sent. Note that different outgoing links in independent link padding schemes should also have the same sending rate. Otherwise, the adversary can use the traffic rate to correlate the flow of different links.

For independent link padding schemes, the probability that some packet can not meet the delay bound is non-zero when the incoming flow is a Poisson process. This is because the probability that a Poisson process has more than N arrivals in a given time period is non-zero for arbitrarily large N. Some independent padding algorithms drop all packets that cannot meet the delay constraint [28]. In this case, the drop rate is an important metric to measure the efficiency of the algorithm.

The DLP algorithm does not drop packets with Poisson traffic, while the rate it uses grows unbounded as $O(\log m)$ when the number of incoming flow m increases. Therefore, it is unfair to directly compare the DLP algorithm with independent padding algorithms. Here we provide a DLP heuristic algorithm which drops some token to reduce the sending rate of the generated schedule. Similar to independent link padding algorithm, this heuristic algorithm uses a schedule with limited rate for arbitrarily large number of flows.

DLP Heuristic Algorithm
Parameters: Packet arrival time t_{ij} for all flows $f_i \in \mathcal{F}$
Utility threshold U.
Output: A sending schedule with utility of at least U
01: Put new packet P_{ij} into a FIFO queue for the flow f_i
02: Repeat step 01 until there is a packet P has been in the queue for Δ time units
03: **if** more than $U|\mathcal{F}|$ queues are non-empty
04: Add a new token and send one packet for each flow immediately
05: **else**
06: Drop the packet P.
07: **endif**
08: Go to step 01 until no more packet arrives.

Figure 8: Dependent Link Padding Algorithm for a given flow set $\mathcal{F}$.

5.1 DLP Heuristic

Define token utility for a token as $u = \frac{d}{|\mathcal{F}|}$, where d is the number of message packets sent by the token and $|\mathcal{F}|$ is the size of the incoming flow set of DLP algorithm. It is easy to see that $\frac{1}{|\mathcal{F}|} \leq u \leq 1$, as each token scheduled by DLP should at least send one message packet and it can at most send one packet for each flow.

The DLP heuristic algorithm checks the utility of every token before using it. A token is used only if its utility is larger than a given threshold U. If a token is not used due to low utility, all packets scheduled at this token will be rescheduled or dropped if its delay bound cannot be met. Fig. 8 shows the details of the DLP heuristic algorithm.

The DLP heuristic algorithm requires tokens to have utility of at least U. If the average arrival rate for each incoming flow is λ, then the sending rate for the DLP heuristic algorithm will be at most $\frac{\lambda}{U}$ for any number of incoming flows.

5.2 Packet Drop Rates

For a constant padding scheme with sending rate of R_c, the interval between consecutive packets at the outgoing link is $1/R_c$. When we have $\Delta = K/R_c$ for some integer K, the scheduling scheme can be treated as an M/D/1/K queue. The drop rate can be solved via queuing theory [29]. For exponential link padding, the drop rate is given in [11].

Fig. 9 compares the packet drop rate of different padding algorithms. For DLP heuristic algorithm, we change the utility threshold U from $\frac{1}{|\mathcal{F}|}$ to 1 to adjust the sending rates of the schedule. We see that the DLP heuristic is better than constant rate padding when the number of flows is small. However, the drop rate converges to that of constant rate padding when the number of flows is large. This is because the DLP heuristic algorithm can reduce the number of tokens used when all flows are sending at a low rate. However, such a scenario rarely happens when the number of flows is large. Therefore, the behavior of the DLP heuristic tends to be similar to the constant padding algorithm when the number of flows is large.

The DLP heuristic algorithm can provide protection against availability attacks, where the adversary tries to use a small number of flows to consume all the bandwidth of the the server. The original DLP algorithm is susceptive to such at-

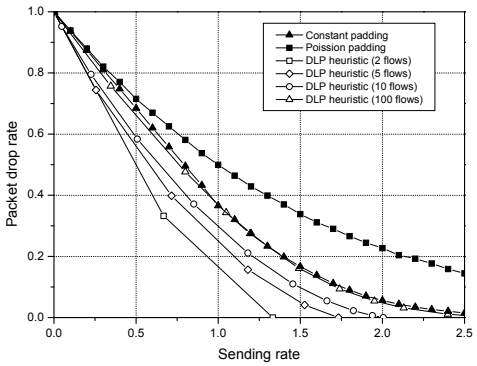

Figure 9: Packet drop rate for different algorithms ($\lambda = 1, \Delta = 1$).

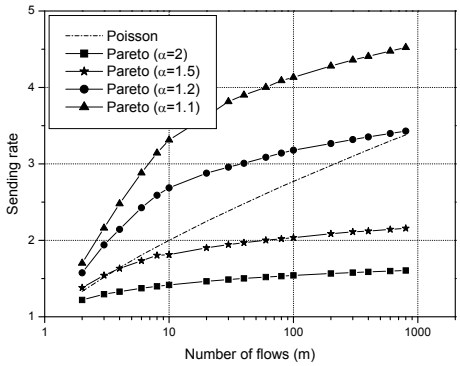

Figure 10: Sending rate of DLP for Pareto traffic when the average rate of incoming flows are normalized to 1 (Δ=1).

tacks as a single flow with a high rate or a small number of bursty flows could make the server send at a high rate on all output links. With the dropping mechanism in DLP heuristic, most packets of these attacking flows will be dropped given that the number of attacking flows is much smaller than that of normal user flows.

6. SIMULATION RESULTS

6.1 Pareto Traffic

In section 4.4, we showed that the sending rate of the DLP schedule increases as $\log m$ when the incoming flows are Poisson processes. However, it has been shown that most Internet traffic is better modeled as a self-similar process rather than the Poisson process [30]. In this section, we consider incoming flows with inter-packet-duration following a Pareto distribution

$$P\{t_{ij} - t_{i(j-1)} \leq x\} = \begin{cases} 0 & x < \tau \\ 1 - (x/\tau)^{-\alpha} & x \geq \tau \end{cases} . \qquad (3)$$

Fig. 10 shows the sending rate of the DLP schedule with different number of Pareto flows of the same average rate.

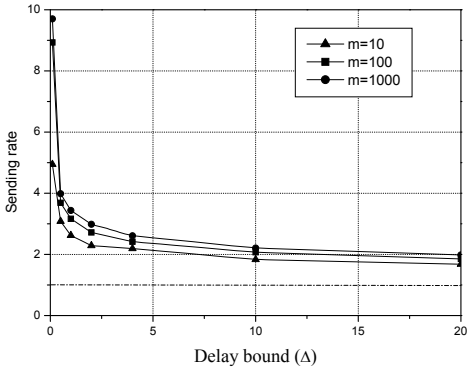

Figure 11: Sending rate of DLP for Pareto traffic with different delay bound.

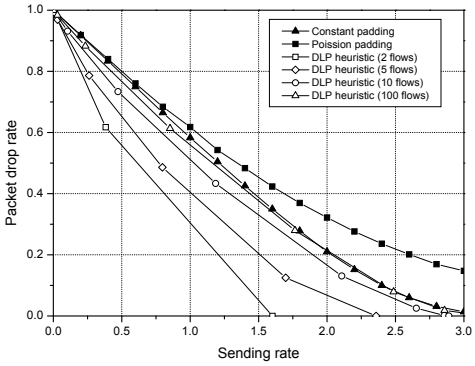

Figure 12: Packet drop rate for Pareto traffic under different algorithms when the average rate of incoming flows are normalized to 1 ($\Delta = 1$).

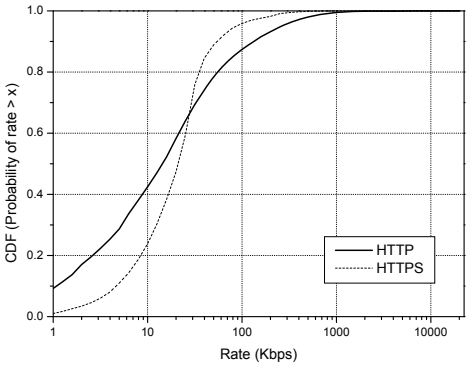

Figure 13: Rate distribution of http and https flows.

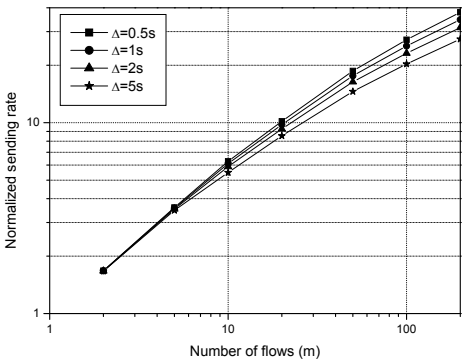

Figure 14: DLP sending rate for http flows under different delay constrains ($\Delta = 1$).

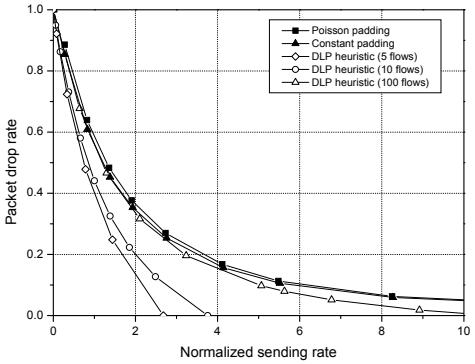

Figure 16: Packet drop rate for https traffic under different algorithms.

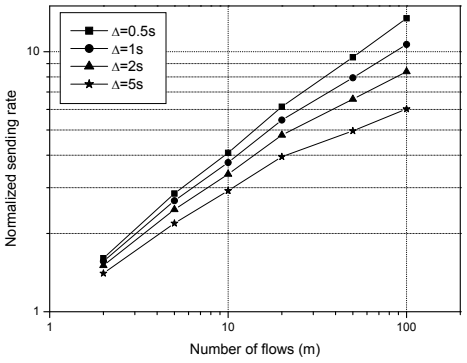

Figure 15: DLP sending rate for https flows under different delay constrains.

Unlike the Poisson traffic, we see that the sending rate actually increases at a speed much slower than $O(\log m)$. For Pareto traffic, the sending rate of DLP is upper bounded by $1/\tau$ as the inter-packet-duration is at least τ. Note that the average inter-packet-duration for Pareto traffic is $\frac{\alpha\tau}{\alpha-1}$ when $\alpha > 1$. Therefore, the rate of DLP schedule is at most $\frac{\alpha}{\alpha-1}$ times the rate of the original flow when the number of incoming flows goes to infinity. From Fig. 10, we also see that the sending rate of DLP increases as the traffic become more bursty when α decreases.

Fig. 11 shows how the sending rate decrease with the delay bound Δ when traffic is Pareto. The sending rate decreases similarly as in Fig. 7. However, the sending rate converges to 1 at a much slower speed for Pareto traffic than Poisson traffic. Fig. 12 shows the packet drop rate of DLP heuristic and independent link padding schemes. We see that the DLP heuristic is better than constant padding and Poisson padding schemes when the traffic is Pareto.

6.2 Simulation on Network Traces

The average sending rate of different flows in real networks can vary drastically. To cover the difference in the traffic patterns, the link padding scheme need to send at a high rate which is larger than the maximum rate of all flows. We use the trace file from the NLANR Auckland-VIII data set [31] in our simulation to study the performance of DLP in real networks. The data contains packet header traces of a network which has more than 10K concurrent flows. All packets have GPS synchronized timestamps.

In the simulation, we randomly pickup m flows from the trace as the incoming flows of a virtual anonymity system. The virtual anonymous server uses the DLP algorithm to generate the matched schedule for these flows. As the real flows are quite short which last around 10-100 seconds, we pick up a new flow when a old flow terminates so that the number of incoming flows stays at m. We also segment all packets to a normalized length of 512 bytes as in many anonymity systems [6].

In real networks, the average sending rate for different users varies drastically. Take the http protocol as an example. The average sending rate of http flows is 25.6Kbps in the trace. However, there are more than 10% flows having an average sending rate larger than 100Kbps as shown in Fig. 13. The maximum sending rate for http flows is more than 10,000Kbps. This makes link padding algorithms less efficient as the padding rate must be larger than the maximum sending rate among all incoming flows. Another problem of real traffic is the burstyness of the http flows. Most http flows exchange data in short bursts which are separated by long silent intervals. In order to cover these bursts, the link padding algorithm needs to send out packets in all links whenever a single flow has a packet. This makes the sending rate of the link padding algorithms extremely high.

Fig. 14 shows the sending rate of the DLP schedule for http flows under different delay constraints. The sending rate in Fig. 14 is normalized by the average rate of all incoming flows. We see that the rate of DLP schedule is increasing as a power law as shown in the log-log plot. When the number of flows is 100, the rate of DLP schedule is more than 20 times larger than the rate of original incoming flows. Moreover, increasing the delay bound from 0.5 second to 5 seconds only slightly reduces the rate of DLP schedule. This may be due to the rate differences in the http flows as shown in Fig 13.

To study how rate variation affects the performance of DLP, we choose https flows which have less variant flow rates. The sending rates for https flows are shown in fig. 15. The normalized rate of DLP is reduced by more than two times for https flows and increasing the delay bound causes significant reductions in the DLP rate.

The different performance on http and https flows suggests that it would be more efficient to group flows with similar sending rate before applying the DLP algorithm. One way to do this is by grouping flows by different applications, such as VoIP, file downloading and web surfing. In each category, the users will have similar traffic rates so that the DLP can be executed in a efficient way. For example, VoIP flows, such as Skype flows, normally have regular traffic patterns so that DLP will not introduce much dummy packets when all flows in the group are Skype flows. However, we may still need to regulate the sending rates which can be used by users for some versatile protocols such as http.

Fig. 16 compares the drop rate of the DLP heuristic with constant and Poisson padding algorithms. We see that the DLP heuristic is better than constant and Poisson padding when the number of flows is small. When the number of flows is large, the drop rate is nearly the same. The difference between constant and Poisson padding is smaller in trace driven simulations. This is because the delay bound used in the simulation is much larger than the average inter-packet-duration for the padding traffic. Therefore, Poisson padding generates a large number of tokens in the duration of delay bound compared to previous simulations. In this case, Poisson padding will have a small probability to have an extremely small number of tokens in the duration of delay bound and the performance is then close to constant padding.

7. CONCLUSION

We studied dependent link padding algorithms in this paper. We showed that the dependent link padding algorithm uses minimum rate to provide full anonymity when there are strict delay constraints on the anonymity systems. However, the rate of the optimal padding algorithm increases very fast when the number of user flow increases. This suggests that it is hard to cover the traffic timing information of flows with long durations by padding the links. Therefore, we may aim to find better covering schemes for flows which last only for a short time to make sure that they cannot be identified within their lifetime.

8. ACKNOWLEDGMENTS

This work was supported in part by the Temasek Defence Systems Institute under grant numbers R-263-000-395-592, R-263-000-396-422, & R-263-000-396-592.

9. REFERENCES

[1] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–88, 1981.

[2] G. Danezis, "The traffic analysis of continuous-time mixes," in *Proceedings of Privacy Enhancing Technologies Workshop (PET)*, 2004.

[3] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer VoIP calls on the internet," in *Proceedings of ACM CCS*, 2005.

[4] U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman, "IETF draft: Mixmaster protocol version 2," http://www.ietf.org/internet-drafts/draft-sassaman-mixmaster-03.txt, 2004.

[5] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous connections and onion routing," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, 1998.

[6] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *Proceedings of the 13th conference on USENIX Security Symposium*, 2004.

[7] M. Freedman and R. Morris, "Tarzan: a Peer-to-Peer Anonymizing Network Layer," in *Proceedings of ACM CCS*, 2002.

[8] A. Serjantov and P. Sewell, "Passive Attack Analysis for Connection-Based Anonymity Systems," in *Proceedings of European Symposium on Research in Computer Security*, 2003.

[9] V. Anantharam and S. Verdu, "Bits through queues," *IEEE Trans. on Information Theory*, vol. 42, no. 1, pp. 4–18, 1996.

[10] X. Wang, S. Chen, and S. Jajodia, "Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems," in *IEEE Symposium on Security and Privacy*, 2007.

[11] P. Venkitasubramaniam, T. He, and L. Tong, "Relay secrecy in wireless networks with eavesdroppers," in *Proceedings of Allerton Conference on Communication, Control and Computing*, 2006.

[12] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "ISDN-Mixes: Untraceable Communication with Very Small Bandwidth Overhead," in *Proceedings of GI/ITG-Conference Communication in Distributed Systems*, 1991.

[13] P. Venkitasubramaniam and L. Tong, "Anonymous Networking for Minimum Latency in Multihop Networks," in *IEEE Symposium on Security and Privacy*, 2008.

[14] P. Boucher, A. Shostack, and I. Goldberg, "Freedom systems 2.0 architecture," White paper, Zero Knowledge Systems, Inc., December 2000.

[15] S. J. Murdoch and P. Zielinski, "Sampled traffic analysis by internet-exchange-level adversaries," in *Proceedings of Privacy Enhancing Technologies Workshop (PET)*, 2007.

[16] S. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *IEEE Symposium on Security and Privacy*, 2005.

[17] N. Hopper, E. Y. Vasserman, and E. Chan-Tin, "How Much Anonymity does Network Latency Leak?" in *Proceedings of ACM CCS*, 2007.

[18] X. Fu, B. Graham, R. Bettati, and W. Zhao, "On effectiveness of link padding for statistical traffic analysis attacks," in *Proceedings of IEEE ICDCS*, 2003.

[19] V. Shmatikov and M. Wang, "Timing analysis in low-latency mix networks: attacks and defenses?" in *Proceedings of ESORICS*, 2006.

[20] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proceedings of Privacy Enhancing Technologies Workshop (PET)*, 2002.

[21] C. Diaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Proceedings of Privacy Enhancing Technologies Workshop (PET)*, 2002.

[22] B. Bollobás, *Modern Graph Theory*. Springer, 1998.

[23] A. Blum, D. Song, and S. Venkataraman, "Detection of interactive stepping stones: Algorithms and confidence bounds," in *Proceedings of International Symposium on Recent Advances In Intrusion Detection*, 2004.

[24] T. He and L. Tong, "Detecting information flows: Improving chaff tolerance by joint detection," in *Proceedings of Annual Conference Information Sciences and Systems (CISS)*, 2007.

[25] G. H. Gonnet, "Expected Length of the Longest Probe Sequence in Hash Code Searching," *Journal of the ACM*, vol. 28, no. 2, pp. 289–304, 1981.

[26] M. D. Mitzenmacher, "The power of two choices in randomized load balancing," PhD Thesis, University of Califonia at Berkeley 1996.

[27] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*. 4th Ed. McGraw Hill, 2002.

[28] P. Venkitasubramaniam, T. He, and L. Tong, "Anonymous networking amidst eavesdroppers," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2770–2784, 2008.

[29] J. W. Cohen, *The Single Server Queue*. North-Holland, 1982.

[30] M. E. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, 1997.

[31] N. L. for Applied Network Research., "Auckland-viii data set," http://pma.nlanr.net/Special/auck8.html, 2003.