On-Demand Link Padding in Traffic Anonymizing

Chen-Mou Cheng, H. T. Kung, Koan-Sin Tan
Harvard University
U.S.A.
doug@eecs.harvard.edu, kung@harvard.edu, freedom@itri.org.tw

Abstract

We consider the problem of using artificially generated cover traffic to hide the network paths used by an application against an adversarial observer who can monitor network traffic and perform traffic analysis. We propose that on-demand link padding be used on top of existing traffic anonymizing infrastructures such as those described in the Onion Routing Project or in the ANON Project. In on-demand link padding, cover traffic is generated dynamically only when real traffic is present. Moreover, the bandwidth usage is economical because we only generate a small amount of cover traffic above real traffic. We describe two types of on-demand link padding schemes and evaluate their performance by simulation. We report our findings based on the simulation results and quantify potential trade-offs in configuring a couple of important parameters. Finally, we provide a strategy for mitigating the potential problem in face of lost or corrupted control packets, to which the traditional recovery solutions based on packet retransmission are inappropriate for performance considerations.

Keywords: Traffic Anonymizing, Link Padding, Network Security.

1 Introduction

We consider the problem of hiding certain meta information of an application traffic flow from an adversarial observer, such as the source and destination network addresses, or the actual network route taken by the flow. Often time, it is difficult to learn exactly what the adversary can or cannot observe, so we assume the worst: an adversary who can observe all network traffic at all time. We start with two basic weapons: cryptography and a set of trusted, privacy-enhancing forwarders, whose principles of operations are described in more details below. First, packets are encrypted with a non-malleable, semantically secure encryption algorithm [4] such that neither by passively inspecting the content of the packets nor by actively injecting arbitrary packets can the adversary learn any information beyond the fact that the two parties are communicating with each other. However, certain information of the packets, such as the source and destination addresses, cannot be hidden using mere cryptography; to hide it, we will use a sequence of trusted, privacy-enhancing forwarders to forward packets, to each of which we only need to reveal the addresses of its two immediate neighbors in the sequence. Furthermore, the packets of the target traffic are "mixed" with those of other traffic to make tracing more difficult to an outside observer. The reader is referred to [5][9] for a discussion on and some concrete examples of such forwarders.

For our adversary, effective mixing in the forwarders is important, for the adversary can monitor all network links and perform sophisticated traffic analysis. Such analysis may correlate packets going into and coming out of a forwarder to determine if two trains of packets belong to a same traffic flow, or use contextual information such as the duration of a conversation to launch intersection attacks [8]. To counteract such analysis, we would like to design a network system in which the characteristics of network traffic is, be it with or without application traffic, indistinguishable to the observer. Link padding [1][8] is a well-known solution to this problem. It is based on the generation of artificial load, i.e., *cover traffic*, on a super set of those links where application traffic, which we will call *real traffic* subsequently, traverses. The hope is that with cover traffic, all links in the superset will exhibit indistinguishable behavior, and as a result, the adversary will not be able to use traffic analysis to find out which subsets of these links actually carry real traffic.

Use of padding traffic may, however, reduce the network's available bandwidth for serving real traffic. Consider the scenario where links may pass through one or more intermediate nodes. For example, a link of an overlay network built on top of IP networks may pass through intermediate IP routers, whereas a link of an IP network may pass through intermediate Ethernet switches. In these cases, padding traffic on the link will increase the load on these intermediate nodes and their adjacent links and thus reduce their bandwidth available for serving other real traffic. Note that padding traffic and real traffic are supposed to be indistinguishable, and thus it would be inappropriate to assume that intermediate nodes could give normal traffic preferential treatments over padding traffic.

To mitigate this bandwidth overhead problem of link padding, in this paper we propose an *on-demand* link padding scheme capable of generating cover traffic only in the presence of real traffic. Furthermore, we make sure that cover traffic can be at a rate just above that of real traffic. In this way, we avoid using traditional "flat" cover traffic [8] that does not dynamically adapt to real traffic. Thus, when real traffic is bursty, the proposed approach can significantly reduce cover traffic's blockage of real traffic at intermediate nodes.

Note that on-demand link padding is not to be confused with on-demand bandwidth reservation for quality of service purposes. The service provided by link padding is path hiding, not bandwidth or latency guarantees as in on-demand bandwidth reservation schemes such as RSVP [11].

The rest of this paper is organized as follows. In Section 2, we give a hypothetical application scenario of on-demand link padding and formulate an abstract model for the on-demand link-padding problem. We then present a general mechanism of implementing link padding in Section 3 and describe two types of on-demand link padding in Section 4. We use simulation to demonstrate the advantages of on-demand link padding over traditional flat link padding in Section 5. We report our findings and discuss potential trade-offs in configuring the important parameters of the proposed on-demand link padding scheme in Section 6. We describe our strategy for dealing with the problem of possible control packet losses in Section 7. We review related work in the literature in Section 8 and conclude this paper in Section 9.

2 Application and Model Problem

To illustrate the need of using cover traffic, we consider a hypothetical application scenario. A company has sent agents to a foreign country X to explore a highly secretive business deal and wants to prevent its competitors from knowing exactly which country it is. At various times the company's agents residing in country X need to communicate to the company's headquarters using, e.g., VoIP applications such as Skype. Beyond hiding the content of communication via message encryption (as Skype currently does with AES), the company's security measure will also include hiding traffic so that its competitors cannot deduce the source of the agents' communication (country X in this example) from traffic monitoring and analysis. To implement traffic hiding, the company recruits some number of decoy nodes, which can fool the competitors by sending and receiving cover traffic that looks like real traffic. Whenever an agent has data ready to send, it will first inform the headquarters the amount of data and the rate at which the data is to be sent, through a low-bandwidth control channel that can be easily hidden. The headquarters will convey, over the control channel again, the received information to all the decoy nodes so that they can transmit proper cover traffic at the same time when the agents send the data. Using this method, traffic from the agents to the headquarters can be hidden in decoy nodes' cover traffic. Hiding communication from the headquarters to the agents can be accomplished similarly by having the headquarters send cover traffic to decoy nodes.

From the above scenario, we formulate a model problem that we will use in this paper to study on-demand link padding. Although it is simple, this model problem captures the essential issues of on-demand link padding.

We consider the situation where the network path between a client C and a server S needs to be hidden. The basic protection strategy is that, whenever C and S communicate, similar cover traffic will be generated over multiple network paths. Even for an adversary which could monitor all network links, it would be difficult to tell which network path C and S use, let alone to locate S.

Motivated by the application scenario given earlier in this section, we use the topology of a balanced tree network of depth d, called a cover tree, for hiding a path between C and S, with C being the root and S a leaf, as depicted in Figure 1. Given a path between C and S, the tree is fixed for a period of time in order to guard against intersection attacks [8]. It is possible to use topologies

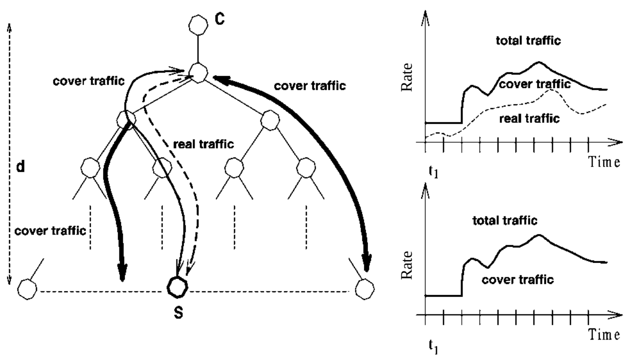

Figure 1 The cover tree for our model problem, where C is the client and S is the server. The diagram on the left illustrates that the total traffic on the path between C and S is the sum of real and cover traffic, while that on any other link is just cover traffic. The two diagrams on the right indicate that at any given time, the total traffic over any link looks the same with or without real traffic

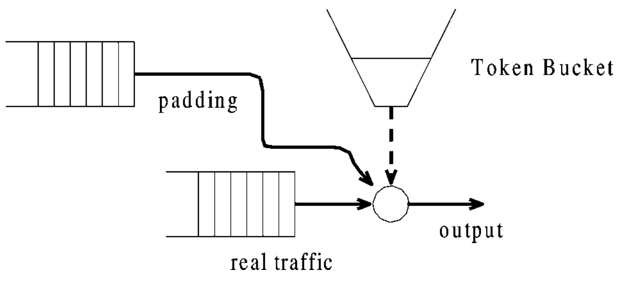

Figure 2 Token bucket based link padding implementation

other than trees; to simplify the presentation, we only use the tree topology to illustrate our on-demand link padding ideas in this paper.

Cover traffic will be generated over all the paths from the root to the leaves, with the property that the *total traffic* over any link in the tree is statistically the same whenever C and S communicate. The total traffic is defined to be the sum of both cover and real traffic.

To achieve this objective, we need to address several issues. First, we need a signaling mechanism that can notify nodes in the tree network to generate cover traffic when real traffic is present. While the total traffic should look statistically indistinguishable across all links, real traffic should not "stick out." Furthermore, the cover traffic should not be excessive to avoid unnecessary load on the network. Finally, participating nodes need to detect whether control messages are lost and act properly so as not to leak useful information to the adversary.

3 Link Padding Operation at a Node

We review a typical token-bucket-based scheme to generate cover traffic. The token bucket receives tokens at a rate corresponding to the target rate of the total traffic, including cover and real traffic. When it receives a token, as depicted in Figure 2, the node will send out a packet from real traffic if the real traffic queue is non-empty. In this case, a token is removed from the token bucket, and a packet is also removed from the real traffic queue.

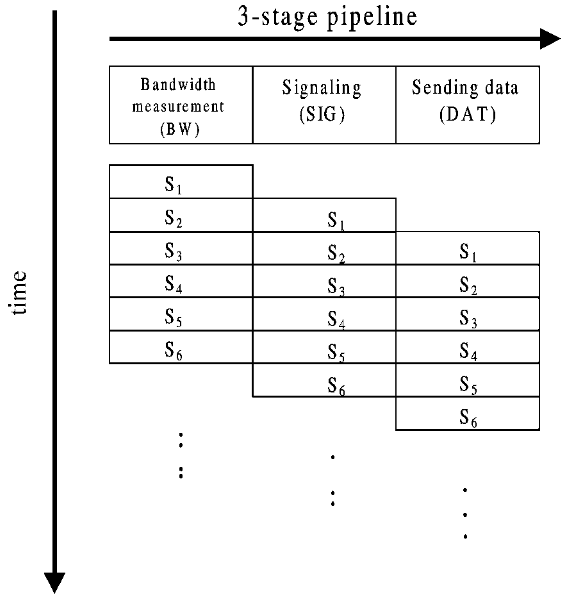

Figure 3 The 3-stage pipeline architecture for data segment processing in on-demand link padding with delay. When segment Si-1 is being sent over the network links, the signal that carries the bandwidth requirement for segment Si is being propagated from traffic source to all the nodes in the network. Meanwhile, the bandwidth requirement of segment Si+1 is being measured while the segment is being transmitting into a delay buffer

Otherwise, the node will send out a padding packet as a part of cover traffic. In this case, only a token is removed from the token bucket.

There are various types of link padding. In the traditional type, which we call flat or constant link padding, padding traffic is inserted to a link so that the link usage is constant [8]. This type of link padding is simple but not bandwidth efficient. In contrast, in *on-demand link padding* as proposed in this paper, padding traffic is inserted dynamically only when real traffic is present.

4 On-Demand Link Padding

In this section, we describe our proposed on-demand link padding scheme within the context of our model problem. We use a signaling channel to send out control messages (also called signaling messages) from the root node to the participating decoy nodes at leaves to notify the amount of link padding to use. To ensure that any two communicating parties receive enough covering, we use explicit acknowledgment messages sent from participating decoy nodes to the root node. We assume the control channel is encrypted, and covered possibly by constant link padding. This should not significantly degrade available bandwidth of the underlying network to serve other real traffic because control traffic uses only modest bandwidth.

For simplicity, on-demand padding described here addresses the situation where there is only one client C requesting cover traffic, and hence there is only one single tree rooted at the client. When deployed on a large network, multiple clients can request for cover traffic on multiple trees at the same time. If a node receives multiple requests for cover traffic, the node will try to meet all the requests. If the node is unable to satisfy a request because the total bandwidth need by all incoming requests is larger than the bandwidth reserved for the anonymizing service, then the node rejects the request by sending a negative reply. As depicted in Figure 3, the node sends out replies in the signaling stage to notify each of the requesting clients whether their individual requests can be met or not. Replies or acknowledgements destined to the same client may be combined or piggybacked in data packets to save bandwidth.

The basic concept of on-demand link padding is to add padding traffic based on the bandwidth usage observed from real traffic. This allows padding to be applied only when real traffic is present. Moreover, to minimize its impact to other traffic, the amount of the padding should be limited to a level just sufficient to cover real traffic. We present two types of on-demand link padding.

4.1 On-Demand Link Padding with Delay

The first type, called *on-demand link padding with delay*, works as follows. Consider real traffic from C to S in our model problem of Section 2. We break C's traffic into consecutive data segments, each lasting for one round-trip time (RTT), where RTT is the longest round-trip time among the paths between the root and the leaves in the cover tree network of Figure 1. When a segment departs from C, it will stay in a delay buffer at the output port of C for a time period of 2RTT. During the first RTT, the bandwidth usage of the segment is measured. During the second RTT, a signaling message is sent to all the tree nodes, requesting them to generate padding traffic so that the total traffic over each link will be at a rate equal to the measured bandwidth. At the end of the second RTT, if C has received sufficiently many acknowledgements from the leaf nodes, it sends out the segment at the rate of the measured bandwidth over a period of RTT.

For real traffic from S to C, an approach similar to the one described above is used. There is a delay buffer at the output port of S. After having measured the bandwidth of the current segment, S will send a notification about the measured bandwidth to the root C so that C can signal all tree nodes to generate proper padding traffic. The delay buffer in this case needs to be extended to accommodate the additional delay for S to send the bandwidth notification to C, and for C to send acknowledgements back to S.

As shown in Figure 3, we use a pipeline architecture to process three consecutive segments simultaneously at three pipeline stages. The three stages, each being one RTT long, are for bandwidth measurement, signaling, and data sending.

In both cases of sending traffic from C to S and from S to C, since the bandwidth usage of a segment is measured before the target rate of link padding is set, on-demand link padding with delay can target the total link bandwidth usage to be exactly the same as the measured bandwidth. Therefore, in on-demand link padding with delay, there is no waste of network bandwidth in cover traffic. However, the cost is that the real traffic will be delayed for two or three RTTs.

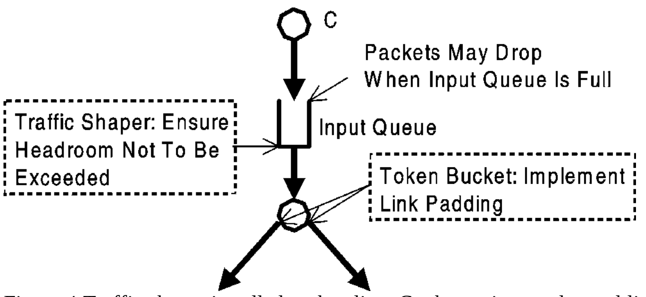
C
Packets May Drop When Input Queue Is Full
Traffic Shaper: Ensure Headroom Not To Be Exceeded
Input Queue
Token Bucket: Implement Link Padding

Figure 4 Traffic shaper installed at the client C when using on-demand link padding with headroom. The shaper will shape the outgoing traffic from C and cap rate increases within the allowed range absorbable by the headroom size. When the traffic rate of C increases, the input queue will build up, and packets will be dropped if the queue fills up. The shaper can be implemented using a token bucket mechanism similar to that depicted in Figure 2

4.2 On-Demand Link Padding with Headroom

The second type of on-demand link padding is *on-demand link padding with headroom.* This type is of interest because it does not incur extra delays. The basic idea is to use measured bandwidth usage of a previously sent segment to predict that of the current one and to add headroom in the allocated bandwidth to allow traffic bursts to pass through. We describe the idea by considering real traffic from C to S in Figure 1. By using a traffic shaper, as shown in Figure 4, we can ensure that any bandwidth increase in the current segment is limited to an allowable range. This means that the current segment can be sent out immediately.

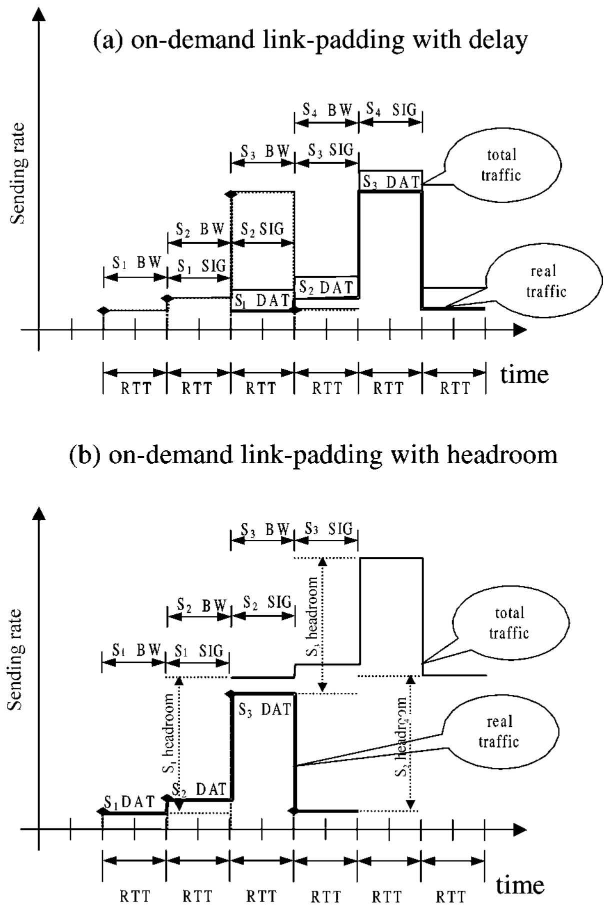

Figure 5 Sending rate as a function of time. Both (a) and (b) show the pipeline depicted in Figure 3. The upper diagram (a) shows the 2RTT delay. For example, S_1 is sent using the bandwidth measured during S_1 BW and the signal delivered during S_1 SIG. The lower diagram (b) shows that with sufficient headroom we can send segments without delay. For example, S_3 is sent using the bandwidth measured during S_1 BW and the signal delivered during S_1 SIG. Since the sum of the measured bandwidth and headroom is larger than the bandwidth required by S_3, we can send it without delay

Table 1 Simulation Configuration

Simulation time	600 sec.
Link bandwidth	10 Mbps
Link propagation delay	16.7 ms
Round-trip propagation delay	100 ms
Packet size	1000 bytes

However, this zero-delay link padding method suffers from a drawback. That is, large headroom is needed to accommodate real traffic that may or may not have large bandwidth fluctuations between consecutive segments. Use of large headroom while there is not corresponding large real traffic results in wasteful use of network bandwidth in link padding.

To alleviate this link efficiency problem, we use the traffic shaper with an input queue, as depicted in Figure 4, to absorb bandwidth fluctuations between consecutive segments, thereby reducing the need of large headroom. The queueing required in the traffic shaper, however, could introduce delay. Thus when using on-demand link padding with headroom, we can trade off queueing delay at the traffic shaper for small headroom.

4.3 Illustration

We use Figure 5 to illustrate the two types of on-demand link padding described above. In Figure 5(a) we show that the 2RTT delay is required by on-demand link padding with delay. For example, the first segment S_1 is sent using the bandwidth measured during S_1 BW and the signal delivered during S_1 SIG. Figure 5(b) shows that using on-demand link padding with headroom, we can send segments immediately. For example, segment S_3 is sent using the bandwidth measured during S_1 BW and the signal delivered during S_1 SIG. As long as the sum of the measured bandwidth and headroom is larger than the bandwidth required by S_3, we can send S_3 without delay, while ensuring that the traffic of S_3 will not stick out above the total traffic seen on other paths.

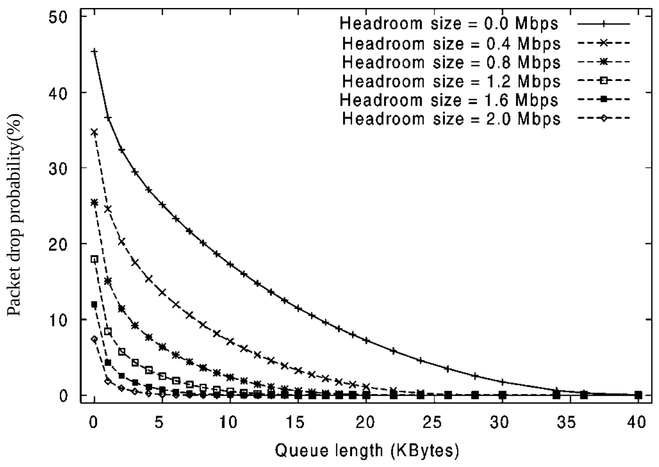

Figure 6 The effect of queue size and headroom size on packet drop probabilities. A Pareto traffic source is used with the average rate being 0.876 Mbps. The bandwidthdelay product is about 6 KBytes. The figure shows that the packet drop probabilities can be decreased by either increasing the queue size or the headroom size

5 Simulation

We report the results obtained from *ns-2* [3] simulations in this section. These simulation results assume on-demand link padding with headroom.

5.1 Simulation Setup

The topology of the simulated network is a complete binary tree of Figure 1, with the client C sitting at the root node. We investigate the case where the depth of the tree is four (d = 4), i.e., the client wishes to send data to a server three hops away at a leaf node. The parameters are summarized in Table 1.

We assume that the client is an open-loop traffic source that generates constant-sized packets at a time-varying rate. The packet inter-arrival times are drawn from Pareto distributions with shape parameters between 1.05 and 1.15 [10]. Recall that the probability density function of Pareto distribution is $f(x) = \beta \alpha^{\beta} x^{-\beta-1}$ where the “shape” parameter β controls the shape of the distribution. Pareto distribution is widely used in modeling the heavy-tailed behavior of Internet traffic [7]. Note that when the shape parameter of Pareto distribution is between 1 and 2, it has finite mean but infinite variance.

When the client C generates a packet, the packet is delivered to the input queue, as shown in Figure 4, where it might be queued for later transmission. The queue is of fixed size, and when it is full, the newly generated packet will be dropped. We use packet drop probabilities as a metric to measure performance of different link padding methods.

5.2 Simulation Result

We consider three factors that affect performance of link padding methods: queue size, amount of cover traffic, and source traffic characteristics.

First, we investigate the effect of queue size and headroom size. Queue and headroom can absorb traffic increases. As we can see in Figure 6, increasing either the queue size or the headroom size decreases packet loss probabilities. The applications traffic is generated using Pareto inter arrival times of shape 1.1 at an average rate of 0.876 Mbps.

Next, we compare the performance of flat link padding and on-demand link padding with headroom.

In Figure 7, we show that the effects of different queue sizes. For achieving the same packet drop probability, on-demand padding uses less cover traffic than flat padding, when the queue size is reasonably large. The average data traffic rate is 0.735 Mbps, generated by the same Pareto traffic source as used in the previous simulation.

Finally, we show how traffic characteristics play an important role in packet drop probabilities. As Pareto distribution of shape 1.05, it requires more cover traffic than less bursty traffic sources, such as those of shape 1.1 in Figure 6 and Figure 7. The average data traffic rates are 1.019, 0.735, and 0.663 Mbps for shape 1.05, 1.1, and 1.15, respectively.

6 Discussion

We use queue and headroom to absorb traffic bursts, which results in a trade-off between delay and link utilization, as mentioned earlier in Section 4.2. As we will explain below, using queue or headroom alone is insufficient to achieve a low packet loss rate. This is because the marginal effect of increasing either queue size or headroom size decreases if we increase either alone.

The diminishing marginal effect of increasing headroom size alone can be illustrated using the diagram in Figure 9. For simplicity, we ignore the discrete steps in our control mechanism and regard the resulting on-demand link padding system as evolving continuously over time. The product of headroom size and signaling delay determines the maximum ramp-up speed of the application traffic. If the traffic ramps up too fast so that it generates more excess packets than what the queue can hold, then those surplus packets will be dropped. This is independent of traffic rate; it can happen even if the traffic rate is low. Flat link padding, on the other hand, suffers from packet losses only when there are persistent bursts whose accumulative deficit in bandwidth exceeds what the queue can hold, such as region GHJK in Figure 9. In the simulation, when the queue size is small, increasing cover traffic beyond a certain limit would benefit flat link padding more than on-demand link padding. This results in the crossover of the curves in Figure 7 and Figure 8. From the above illustration, we can see that the reason for such a slowdown is that when we increase cover traffic volume, the probability of having persistent bursts decreases faster than the probability of having long ramp-up.

7 Handling Signaling Packet Loss

Packet losses due to network buffer overflows and other reasons are inevitable in a large network such as the Internet. When the signaling packet that tells a participating node at the leaves of the cover tree how much padding traffic to insert in the following round fails to reach its destination, that particular node will not be able to provide covering for the root node; even if it decides to, the adversary would easily distinguish the node, for it will not know the exact amount of cover traffic to send and hence cannot pretend to be the real source of the traffic.

As we have stated earlier in Section 4, in order to ensure that the root node will receive enough covering, we use explicit acknowledgment messages sent from leaf nodes to the root node of a cover tree. The root node will stop sending if it does not receive enough acknowledgment

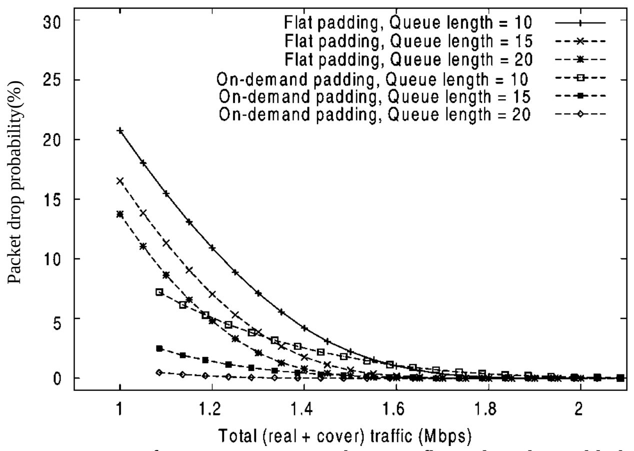

Figure 7 Performance comparison between flat and on-demand link padding. For the same queue size, flat padding uses more cover traffic in order to achieve the same packet drop probability. However, if the queue is not large enough, adding more headroom to on-demand padding is less effective, compared with adding more bandwidth to flat padding. This phenomenon of crossover curves is explained in Section 6

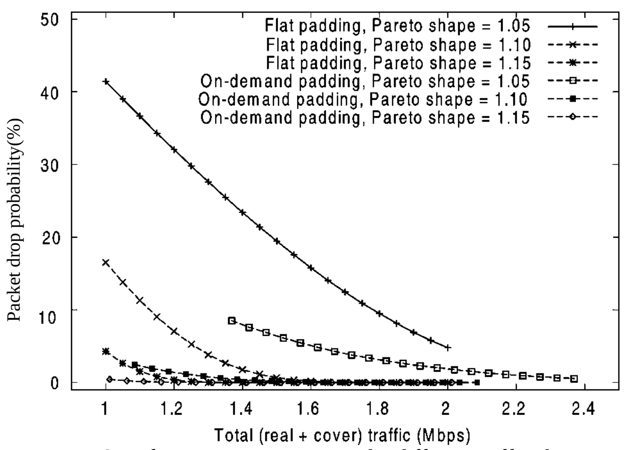

Figure 8 Performance comparison under different traffic characteristics. The shape parameter in Pareto distribution controls burstiness of traffic rate: traffic rate with smaller shape parameter is more bursty than those with larger shape parameters. Flat padding is more prone to increasing burstiness in the traffic rate than on-demand padding, as is evident in the larger gaps between curves representing flat link padding of different traffic characteristics

messages. Reasons for the root node not receiving acknowledgment messages from some of the leaf nodes include, but not limit to, that the acknowledgment messages sent by leaf nodes may get dropped in the network, that these leaf nodes did not receive signaling messages in the first place, and that these leaf nodes may simply go down during the course of the communication. Such events can pose a security threat to the anonymizing service provided by the system.

For example, consider the scenario in which a server sits at a leaf node of a cover tree, so that a client at the root node can request services from the server without knowing the address of the server [5]. An adversary who can monitor output links of all leaf nodes may be able to infer useful information by continually making requests to the server and checking received replies. If, during certain period of time, the adversary receives replies from the server and observes that a small set of leaf nodes are not participating, then he can infer that these nodes cannot be the real server. In the rest of this section, we analyze the success probability of such attacks and describe schemes that can tolerate losses of signaling packets.

7.1 Model Problem

Consider the situation where the adversary is able to act like a legitimate user by sending requests to a server S. In this case the adversary wants to find out the address of the server S. The system recruits a set of n nodes, called *camouflaging nodes*, to provide cover for S (for example, in Figure 1 the camouflaging nodes are the leaf nodes other than S). Time is segmented into runs, and before the beginning of each run, a signaling packet is sent to each of the n camouflaging nodes and S, in which the amount of padding is disclosed. We denote by X an arbitrary camouflaging node: if X fails to receive the signaling packet for a particular run, it will not send any padding traffic in that run. Similarly, if S does not receive the signaling packet, it will not send any traffic in that run. We assume that in each run, a camouflaging node X or server node S fails to receive its signaling packet with some small probability p such as 0.01, and whether or not a node receives its signaling packets is independent of any other node.

We further assume that the adversary is able to monitor traffic on links connecting to all camouflaging nodes and server S; however, due to link encryption and link padding, it can only tell the total amount of traffic within each run that flows through these links. In each run, we denote the set of all nodes that send the correct amount of padding traffic as a “participant group” (PG) and the complement as a “non-participant group” (NPG). Nodes in PG are called participating nodes or simply participants for that run. As a result, by monitoring the links, the adversary can identify whether a node is in PG or in NPG.

Since the adversary is acting as a legitimate client, it will be able to tell whether the server S has sent out true replies or not in any run. We denote a run to be a *successful run* if the adversary receives a true reply message from server S; otherwise it is an *unsuccessful run*. We summarize our model problem in Figure 10.

7.2 Convergence Time for Attacks that Exploit Successful Runs

We first investigate the case where the adversary exploits information obtained from successful runs. Since S must be in PG in a successful run, those nodes in NPG cannot be S. Only those nodes that are in all PGs of all successful runs can be a candidate for S. We denote by AS(s) the set of nodes that still cannot be pruned after s successful runs. Then the expected size of this set, |AS(s)|, is

$$E[\,|AS(s)|\,] = n(1-p)^S.$$

In the above calculation, we have used the fact that |AS(s)| is a binomially distributed random variable with size n and probability $(1-p)^s$. Recall that a binomial random variable with size $\hat{n}$ and probability $\hat{p}$ can be interpreted as, say, the number of heads in tossing $\hat{n}$ biased coin, where the outcome of each coin toss is head with probability $\hat{p}$ independently of that of any other coin toss. We can view the event that a camouflaging node X appears in all PG of s successful runs as a “head” event in a coin toss with the probability of a head being $(1-p)^s$, which we assumed to be independent of that of any other node. Consequently, |AS(s)| is binomially distributed with size $\hat{n} = n$ and probability $\hat{p} = (1-p)^s$.

We give an illustrative scenario where the adversary wants to be 99.9% sure that a particular node is S, i.e., it wants |AS(s)| = 0 with probability of 0.999, after s runs. For p = 0.01 and n = 1000, the adversary will need to observe for roughly s = 1400 runs, where s is the solution to the equation:

$$(1-(1-p)^s)^n = 0.999$$

Recall that S has the same probability p of being in PG. Thus the expected number of runs for each successful run is 1/(1-p), which is about 1.01 for p = 0.01. Since 1414/1.01 is approximately 1400, the expected total number of runs for the adversary to identify S with 99.9% of probability is about 1414 runs.

7.3 Convergence Time for Attacks that Exploit Unsuccessful Runs

Intersection attacks can also be based on unsuccessful runs, assuming that server S always sends out true replies when it receives signaling packets. Under this assumption, those nodes which do send out cover traffic in an unsuccessful run cannot be S; only those nodes which are in all NPGs of all unsuccessful runs can be S. We denote the set of nodes that still cannot be pruned after *r* unsuccessful runs as AS(*r*). Then the size of this set, |AS(*r*)|, is also a binomially distributed random variable with size *n* and probability p^r. Thus, the expected number of camouflaging nodes in AS(*r*) is np^r after *r* runs.

Attacks that exploit unsuccessful runs are more effective than those that exploit successful runs when *p* is small. That is, an adversary might be able to get |AS(*r*)| = 0 more quickly than |AS(*s*)| = 0. For example, for *p* = 0.01 and *n* = 1000, we have $r \approx 3$ by solving the following equation:

$$(1-p^r)^n = 0.999$$

Note that the expected number of runs for S to appear in the NPG of one of these runs is 1/*p*, which is 100. Thus the expected number of runs for the adversary to identify S with 99.9% of probability is only about *r**100 = 300, much smaller compared with the number of 1414 runs in Section 7.2. In this case, attacks exploiting unsuccessful runs are more effective than those exploiting successful runs.

7.4 Adding S Randomness: First Attempt to Defend Against Attacks that Exploit Unsuccessful Runs

The intersection attack mentioned in the previous section depends on the assumption that server S will always send out true replies when it receives signaling packets. The consequence is that, when the adversary receives no true reply, it can deduce that S did not receive its signaling packet, i.e., S is in NPG. To defend against this attack, we can add randomness to the behavior of S, namely, we will allow S not to send true replies with certain probability *q* even when it does receive signaling packets. That is, when S receives a signaling packet, with probability *q* it will pretend to be a camouflaging node by

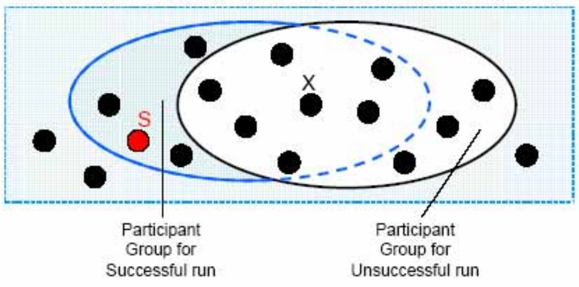
X
S
Participant Group for Successful run
Participant Group for Unsuccessful run

Figure 10 A model problem for analyzing intersection attacks. Depending on whether or not a node receives the signaling packet in a run, it can be in the participant group (PG) or non-participant group (NPG). The adversary classifies each run to be either successful or unsuccessful, depending on whether it has received true replies from the server. The server is denoted by S and any of the camouflaging nodes by X. Note that in a successful run, S must be in PG

sending out pure padding traffic instead of true replies. From the adversary's point of view, this is an unsuccessful run since it does not receive any true reply. Note that in this case S is in PG because it does send out something, so the adversary can no longer use the rule that S must be in NGP for every unsuccessful run.

Let *u* and *v* denote the probability of S or X in PG of an unsuccessful run, respectively. Then, $u = (1-p)\,q$, whereas $v = p\,(1-p) + u\,(1-p)$.

The value of *v* consists of two terms: the first corresponds to the case when S does not receive its signaling packet but X does, while the second one corresponds to the case when S pretends to be a camouflaging node and X happens to receive its signaling packet. We can rewrite *v* as

$$v = p(1-p-u) + u = p(1-p)(1-q) + u \geq u.$$

This means, unless *q* is equal to 1, in which case S will not be able to send any true reply, the probability of S in PG of an unsuccessful run is always less than that of X. Using this fact, the adversary will be able to tell, with high probability, whether or not a particular node is S, when the number of runs is sufficiently large for necessary statistical tests.

7.5 Adding Both S and X Randomness to Defend Against Attacks that Exploit Unsuccessful Runs

To correct the problem that S is more probable to appear in NPG of an unsuccessful run than X, we increase the probability of X appearing in NPG from *p* to *ap*, where $p \leq ap \leq 1$ This can be achieved by introducing randomness: when X receives a signaling message, it will send out padding traffic with probability $h = (1-ap)/(1-p)$ and keep silent otherwise. For example, for *a* = 2 and *p* = 0.01, we have *h* approximately equal to 0.99, so that when X receives a signaling packet, with probability 0.99, it sends out padding traffic, and with probability 0.01, it sends nothing. Therefore, the probabilities of X in PG and NPG are always 1 - *ap* and *ap*, respectively, in either successful or unsuccessful runs.

Next, we compute these probabilities for S during unsuccessful runs. The probability that S falls in PG of an unsuccessful run is $(1-p)q$, whereas the probability that S falls in NPG of an unsuccessful run is simply *p*. Because the two cases mentioned above are the only possible ways for unsuccessful runs to happen, we denote the probability of unsuccessful run by η, where $\eta = (1-p)\,q + p$. From the adversary's perspective, given the fact that it is an unsuccessful run, the conditional probabilities of S being in PG and NPG are $((1-p)+q)/\eta$ and p/η, respectively.

We adjust *a* and *q* such that the probabilities of S being in PG or NPG are equal to those of X being in PG or NPG, respectively. This implies

$$a = \frac{1}{\eta} = \frac{1}{(1-p)q+p} \cong \frac{1}{q},$$

when *p* is small. In this case, there is no way an adversary can tell which node is S based on information obtained during unsuccessful runs.

7.6 Trade-offs and Optimal Strategies

From the above formula for *a* and *q*, we note that for small *p*, the product of *a* and *q* remains constant, so increasing the value of one will require decreasing the value of the other. However, one may wish to decrease both values simultaneously for the following reasons. As we have seen in Section 7.2, the expected size of the anonymity set |AS(s)| is $n(1 - ap)^s$, and as we decrease *a*, E[|AS(s)|] becomes larger so that the adversary will have to spend more time pruning; hence it is more difficult for the adversary to identify S based on information obtained from successful runs. On the other hand, as we decrease *q*, S will spend less time on sending pure padding traffic rather than true replies in order to confuse the adversary; thus S achieves higher resource utilization. As a result, depending on the security requirements and resource budget, one may need to choose a pair of suitable parameters *a* and *q* to strike a balance between security and efficiency.

We also note that it is not necessary for the probabilities of S in PG and X in PG of an unsuccessful run to be equal, because the adversary can always identify S using information obtained from successful runs. Therefore one can relax the condition on *a* and *q* such that it takes approximately the same number of total runs for the adversary to identify S via either successful runs or unsuccessful runs. We note that the problem for the adversary to identify S using information obtained from unsuccessful runs is a type of statistical hypothesis testing, in which the adversary has to accept or reject the hypothesis that a set of sample outcomes are drawn from the probability distribution that S is in PG (or equally effectively that S is in NPG). Given any target confidence level for the adversary, one can apply the Chernoff bounds (or resort to similar arguments) to obtain constraints on the probabilities of S in PG and X in PG (and hence those of S in NPG and X in NPG) such that with high probability, the adversary cannot identify S more quickly via unsuccessful runs than successful runs.

8 Related Work

Chaum proposes to construct a synchronous, untraceable email system called MIX [1]. MIX nodes perform anonymizing tasks by collecting, reordering, and distributing messages in a synchronous manner. An outside adversary can only obtain the information that a participant may be communicating with one or more parties involved in a MIX network in that particular round but cannot be sure with which party or parties the participant is communicating. Our entire tree can be thought of as a real-time MIX node, with the root communicating with one of the leaves.

Many extensions to the MIX network that aim to provide real-time traffic transport have been proposed, among which the Onion Routing Project [9] is the most widely known. In Onion routing, the initiator of communication first constructs an Onion, in which the entire path to be taken by subsequent packets is specified. The Onion is then forwarded, a secure virtual circuit is set up, and the application packets are exchanged without link padding. We note that our proposed on-demand link padding schemes can be used to help Onion routing achieve better anonymity protection against a more powerful adversary model.

The purpose of the NymIP [6] proposal aims to define and deploy standardized protocols for pseudonymity and anonymity at the IP layer. ANON [5] and Onion Routing [9] represent some implementation efforts in IP anonymizing. All these projects emphasized the importance of link padding.

9 Conclusion

We have investigated the problem of using on-demand link padding to hide the network paths used by applications. We propose to generate cover traffic dynamically on demand, only when real traffic is present. We describe the control mechanisms and report performance comparisons with flat link padding using *ns-2* simulation. The results show that, for open-loop traffic sources, on-demand link padding experiences fewer packet loss than flat link padding when both methods use comparable link bandwidth. Furthermore, we have considered the situation where control messages could be lost. We describe how nodes should behave under such circumstances in order to prevent useful information from leaking to the adversary.

We note that the problem considered in Section 7.5 can be placed in a more general setting of anonymizing infrastructure to obtain security parameters in defense against generic intersection attacks. It is typical that one side of the situation cannot be changed, e.g., S has to be in PG during successful runs in our model problem, so the participant nodes have to cooperate in other situations to reduce the probability of S being identified.

Acknowledgment

This work was supported in part by DARPA through AFRL/IFKD under contract F33615-01-C-1983.

Reference

[1] Chaum, D., "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, Vol. 24, Feb. 1981, pp. 84-88.

[2] Crovella, M. and A. Bestavros, "Self-similarity in world wide web traffic: Evidence and possible causes," in *Proc. SIGMETRICS'96*, Philadelphia, Pennsylvania, May 1996.

[3] Breslau, L., D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," *IEEE Computer*, Vol. 33, No. 5, May 2000, pp. 59-67.

[4] Dolev, O., C. Dwork, and M. Naor, "Non-malleable cryptography," in *Proc. ACM STOC'91*, New Orleans, Louisiana, May 1991.

[5] Kung, H. T., C-M Cheng, K-S Tan, and S. Bradner, "Design and Analysis of an IP-Layer Anonymizing Infrastructure," The Third DARPA Information Survivability Conference and Exposition (DISCEX 3) , April 2003, pp. 62-75.

[6] NymIP Effort, http://www.nymip.org/

[7] Paxson, V. and S. Floyd, "Wide-area traffic: The failure of poisson modeling," *IEEE/ACM Trans. Networking*, Vol. 3, No. 3, Jun. 1995, pp. 226-244.

[8] Raymond, J.-F., "Traffic analysis: Protocols, attacks, design issues and open problems," *Designing Privacy Enhancing Technologies: Proceedings, International Workshop on Design Issues in Anonymity and Unobservability*, LNCS, Vol. 2009, Springer-Verlag, 2001, pp. 10-29.

[9] Reed, M., P. Syverson, and D. Goldschlag, "Anonymous connections and onion routing," *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 4, May 1998, pp.482-494.

[10] von Seggern, D., CRC *Standard Curves and Surfaces*, CRC Press, 1993, p. 252.

[11] Zhang, L., S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource reservation protocol," *IEEE Network*, Sep. 1993.

Biographies

Chen-Mou Cheng is currently a Ph.D. student at Harvard University. His research interests include network security and privacy, sensor networks, and wireless communications. Cheng received his S.M. and S.B., both in Electrical Engineering, from National Taiwan University, Taiwan.

H. T. Kung is currently William H. Gates Professor of Computer Science and Electrical Engineering at Harvard University. Since 1999, he has co-chaired a joint Ph.D. program with the Harvard Business School on information, technology and management. Prior to joining Harvard, he served on the faculty of Carnegie Mellon University from 1974 to 1992. Kung has pursued a variety of interests over his career, including complexity theory, database theory, systolic arrays, VLSI design, parallel computing, computer networks, and network security. He received his Ph.D. from Carnegie Mellon in 1974 and B.S. from National Tsing Hua University in Taiwan in 1968.

Koan-Sin Tan is a researcher at the Advanced Technology Center in Industrial Technology Research Institute, Taiwan. His research interests include congestion control, network security, embedded software, and sensor networks. Tan received his Ph.D. in Information Management from National Chiao-Tung University, Taiwan, in 2003.