On the trade-off between speed and resiliency of Flash worms and similar malcodes

Duc T. Ha and Hung Q. Ngo
Department of Computer Science and Engineering
The State University of New York at Buffalo

Abstract— **Inspired by the Flash worm paper [1], we formulate and investigate the problem of finding a fast and resilient propagation topology and propagation schedule for Flash worms and similar malcodes. Resiliency means a very large proportion of infectable targets are still infected no matter which fraction of targets are not infectable.**

There is an intrinsic tradeoff between speed and resiliency, since resiliency requires transmission redundancy which slows down the malcode. To investigate this problem formally, we need an analytical model. We first show that, under a moderately general analytical model, the problem of optimizing propagation time is NP-hard. This fact justifies the need for a simpler model, which we present next. In this simplified model, we present an optimal propagation topology and schedule, which is then shown by simulation to be even faster than the Flash worm introduced in [1]. Our worm is faster even when the source has much less bandwidth availability. We also show that for every preemptive schedule there exists a non-preemptive schedule which is just as effective. This fact greatly simplifies the optimization problem.

In terms of the aforementioned tradeoff, we give a propagation topology based on extractor graphs which can reduce the infection time linearly while keeping the expected number of infected nodes exponentially close to optimal.

I. Introduction

Recent Internet worms are very speedy and destructive, posing a major problem to the security community [2]–[6]. Consequently, researchers have spent a considerable amount of effort studying worms' mechanics and dynamics, such as modeling the dynamics of worms [7], [8], monitoring and detecting worms [9], developing automated worm containment mechanisms [10]–[12], simulating worm traffic [13], routing worms [14], etc.

Most of the aforementioned works focused on random-scanning worms. To the best of our knowledge, very few studies have investigated hypothetical yet potentially super-fast worm scenarios. Staniford et al. [15] were the first to investigate this kind of hypothetical scenario. They later elaborated on a specific worm instance called "Flash worm" [1], so named since these worms can span the entire susceptible population within an extremely short time frame.

Studying potentially super-fast worms/malcodes is fruitful for a variety of reasons. Firstly, a sense of the doomsday scenario helps us prepare for the worst. Secondly, they can be used to assess the worst case performance of containment defenses. Last but not least, efficient broadcasting is a fundamental communication primitive of many modern network applications (both good and malicious ones), including botnets' control, P2P, and overlay networks. For example, since the control of a botnet is gained through efficient broadcasting [16], both the attacker and the system analysts strive to have the most efficient and resilient strategy. Thus, studying worm propagation helps discover improved solutions to security threats in network environments, for both defense and counter-attack purposes.

There are several major challenges to designing and developing super-fast worms.

Firstly, the victim scanning time must be minimized or even eliminated because heavy scanning traffic makes worms more susceptible to being detected, and scanning traffic potentially self-contend with propagation traffic, resulting in slower propagation speed. Various stealthy scanning techniques can be used as an alternative to amass information for the attacking hour.

Secondly, the collection of victim addresses is quite large. For a population of one million hosts, for example, the IP address list requires roughly 4MB (for IPv4). This much data integrated within each worm instance and transmitted without an efficient distribution scheme will severely impact the speed of propagation.

Thirdly, making the worm resilient to infection failures while maintaining its swift effect is challenging. The list of vulnerable addresses may not be perfect. Some of the nodes in the list might be down or no longer vulnerable. At and during the time of propagation, some intermediate nodes might be patched and the worm instance is removed. Moreover, packets carrying the worm code may be lost, leaving the targets uninfected. If an uninfected target is close to the initial source in the propagation tree, all sub branches in the propagation tree will not be infected. In the absence of more sophisticated and probably time-consuming mechanisms (such as timeouts and retransmissions), one might have to reduce the number of levels in the tree and let the source (usually guaranteed to be infected) infect many targets directly. This burdens the initial source node, slows the worm down, and thus makes it more prone to being detected.

Last but not least, computational and communication resources at the source and targets also greatly affect the worm's speed. The Flash worm described in [1] requires an initial node that can deliver 750Mbs. Compromising a host with that much bandwidth capacity may not always be an option. A natural question that follows is whether the attacker can create the same or similar effects as this Flash worm with more limited communication resources. This paper will answer this question in the affirmative.

Consequently, designing a worm propagation topology and schedule that provides both infection failure resiliency and time efficiency is an important and challenging problem. This paper aims to investigate this problem. Specifically, we will address the following questions:

- Suppose the worm writer has some estimates of a few parameters affecting the worm's speed, such as the average end-to-end delay and the average bandwidth, how would he design the worm transmission topology and schedule to accomplish the task as fast as possible?
- Furthermore, many real-life "glitches" may make some targets uninfectable. For instance, some nodes may be down or have their security holes patched, or worm packets may simply be lost. How can one design a worm which is resilient to these glitches?
- There is an inherent tradeoff between the expected propagation time (*efficiency*) and the expected number of infected targets (*resiliency*). To be more resilient, some redundancy must be introduced. For instance, because node v might fail to infect another node w, we may need to have several "infection paths" from v to w on the propagation topology. Unfortunately, redundancy increases propagation time, hence necessitating the tradeoff. Two related questions we will formally define are: (a) how to design an efficient worm given a resiliency threshold, and (b) how to design a resilient worm given an efficiency threshold.

We will not be able to answer all the questions satisfactorily. However, we believe that our formulation and initial analyses unravel some layers of complexity of the problem and open a door for further exploration.

Perhaps more importantly, the aforementioned tradeoff is not an incidental by-product of the worm propagation problem. Efficient and error-resilient broadcast is fundamental in most network applications [17]–[19]. Hence, results regarding this tradeoff should have applications in other networking areas. On the other hand, while the objectives are similar, the operating constraints are very different between the malcode propagation problem and application-layer broadcast problems.

Our main contributions are as follows:

- We first show that, under a moderately general analytical model, the problem of optimizing propagation time is $\mathbf{NP}$-hard. This fact justifies the need for a refined model, which we present next. Later simulation results further validate the refined model.
- We show that, for every preemptive propagation schedule (i.e. the infection processes from one node to its children can interleave in time) there is a non-preemptive schedule (i.e. each transmission is not interrupted until it is finished) which is just as fast. This fact greatly simplifies the optimization problem. It should be noted that this result does not apply to transmission processes with interactive communication between two ends such as the 3-way handshake in TCP.
- In the refined analytical model we present an optimal propagation topology and schedule. We shall show that it is possible to devise a worm propagation topology and schedule with infection time even shorter than the Flash worm described in [1]. Our worm's infection time also seems to scale very well with the number of nodes. Moreover, it is possible to retain the swift effect of the Flash worm of [1] when starting from a root node with much less bandwidth capacity.
- Under uncertainty, i.e. nodes may fail to be infected with some given probabilities, we investigate the tradeoff between the expected infection time and the expected number of infected targets. We derive the optimal expected number of infected nodes along with the corresponding propagation topology. We then give a propagation topology which can reduce the infection time linearly while keeping the expected number of infected nodes exponentially close to optimal.

The rest of this paper is organized as follows. Section II formulates the problem rigorously, and presents preliminary complexity results. Section III presents the design of a new super-fast Flash worm based on a refined analytical model. Section IV addresses the aforementioned tradeoff. Section V discusses some future research problems.

II. Analytical model and complexity results

A. Parameters of our analytical model

Because we want to investigate the tradeoff between speed and resiliency, our analytical model needs several key parameters affecting propagation time (approximate delays, bandwidths), and affecting fault-tolerance (infection failure probabilities).

Consider the situation where there are n hosts, or nodes, and node v_0 is initially infected with the worm. For each node v, let $r_v^{(u)}$ and $r_v^{(d)}$ denote v's up-link and down-link bandwidths, respectively. When $r_v^{(u)} = r_v^{(d)}$, let r_v denote this common value. The maximum effective bandwidth from node v to node w is then $\min\{r_v^{(u)}, r_w^{(d)}\}$.

The capacity of the network core is assumed to be sufficiently large so that nodes can communicate with each other simultaneously up to their available bandwidths. This assumption is justified by several facts: (1) our worm does not generate scanning traffic, which significantly reduces the traffic intensity as a later simulation shows, (2) the total amount of traffic sent by the worm is relatively small compared to the Internet core's capacity, whereas the Internet backbone is often lightly loaded (around 15% to 25% on average) due to over-provisioning [20], (3) one aim of this paper is to design effective worms operating on a vulnerability population with moderate bandwidths (e.g., 1Mbps), and last but not least (4) when some of the worm's packets are lost due to congestion, our resilient propagation topology helps alleviate the problem.

Let L_{vw} denote the propagation delay from node v to node w. Let L denote the average delay. The worm size is denoted by W. This can roughly be understood as the number of bytes of the worm's machine code. A somewhat subtle point to notice is that sophisticated propagation mechanisms might increase W. Most often, though, W should be a constant independent of n. Beside the actual code of size W, the worm must also transmit a fixed number of a bytes per target. Each of these "blocks" of a bytes contains the IP address of the target, and perhaps additional information about the target such as bandwidth.

Lastly, let p be the probability that a randomly chosen target is not infectable due to wrong IP-address, software patched, target down, or firewalled, etc.

B. *Infection topology*

Consider a typical worm infection scenario. Starting from v_0, which keeps a list of addresses and perhaps other information about the targets (bandwidths, delays), the worm selects a subset S of targets to infect. Each node v in S is also delegated set S_v of targets for v to infect on its own. Upon receiving S_v, node v can start infecting nodes in S_v using the same algorithm. In the mean time, v_0 and other nodes in S which were infected before v can also start their infection simultaneously.

We can use a directed acyclic graph (DAG) $G = (V, E)$ to model this process. The vertex set V consists of all nodes, including v_0, which is called the root or the source. There is an edge from v to w if v (after infected) is supposed to infect w. Since we do not need to infect nodes circularly, a DAG is sufficient to model the infection choices. We refer to this DAG as the *infection topology*.

For each node v in G, the set S_v given to v by a parent w is precisely the set of all nodes reachable from v via a directed path in G. (Note that, for distinct nodes u and v, S_u and S_v might be overlapping if we introduce redundancy to cope with infection failures.) In order to give v its list, the parent w must know how to effectively compute (in the piece of code W) the subgraph of G which consists of all nodes that can be reached from w. In principle, the root can presumably pre-compute the entire infection topology for all nodes in the topology. However, giving this pre-computed information to each target requires a large amount of data (of order $\Omega(n^2)$) to be transmitted, which considerably slows down the worm. Hence, we will focus on worms whose code W is capable of computing its own infection sub-topology given only the list of targets.

C. *A remark on topology overlay*

In order to increase resiliency, a sensible strategy is to use multiple, say m independent DAGs, for which nodes closer to the source in one DAG is farther from the source in the other DAGs. This way, if a part of a DAG got "pruned" by failures to infect some nodes close to the source, then with high probability the pruned nodes will be infected via infection paths in the other DAGs. Indeed, [1] used this strategy with m independent m-ary trees. Basically, this strategy sends out m instances of the worm based on the same DAG topology with different rearrangements of vertices. The resiliency is increased at the cost of a (roughly) linear increase in transmission time.

In this paper, we will focus on constructing **one** DAG which is both time-efficient and resilient. The DAG can then be used as a component in the "overlay" of m DAGs as described above.

D. *Infection schedule*

It is not sufficient for nodes to make infection decisions based on the infection topology alone. The second crucial decision for a node v to make is to come up with an *infection schedule* to infect its children in the topology, i.e. the order in which the children are to be infected. A schedule is non-preemptive if the children are to be infected sequentially, one after another, using the maximum possible bandwidth. A schedule is preemptive if transmissions to different children are overlapping in time.

Preemptive schedules make estimating the total infection time quite difficult. Fortunately, in the case of uniform bandwidths and UDP worms, we do not need to consider preemptive schedules as the following Theorem 1 shows. The theorem helps simplify some later analyses.

Theorem 1: Suppose all bandwidths are equal to r. With UDP worms, for every preemptive schedule from a node to a set of children nodes, there is a non-preemptive schedule in which every target is infected at time no later than that in the preemptive schedule.

Proof: Consider any node w that starts to infect m targets, say $v_1, \dots, v_m$, at time 0 following any preemptive schedule. For each target v_i, let T_i be the time w finishes the transmission to v_i in the preemptive schedule. Also, let W_i be the amount of data transmitted to v_i. Without loss of generality, suppose $T_1 \le T_2 \le \cdots \le T_m$. The time at which v_i is infected is $T_i + L$. Let $f_i(t)$ be the amount of bandwidth used for the transmission to v_i at time t. We then have $W_i = \int_0^{T_i} f_i(t)dt$, and $\sum_{j=1}^m f_j(t) \le r$.

Consider the non-preemptive schedule following the same order $1, 2, \cdots, m$, in which w infects one node at a time using the whole bandwidth capacity. For $1 \le i \le m$, let T'_i be the amount of time until w completes the transmission to v_i. The total amount of time until v_i is infected is $T'_i + L$.

To finish the proof, we want to show that $T'_i + L \le T_i + L$, or simply $T'_i \le T_i$. Setting $T_0 = 0$, we have

$$T'_i = \sum_{j=1}^{i} \frac{W_j}{r} = \sum_{j=1}^{i} \frac{\int_0^{T_j} f_j(t)dt}{r} = \sum_{j=1}^{i} \frac{\int_{T_{j-1}}^{T_j} (\sum_{k=j}^{i} f_k(t))dt}{r}$$

$$\le \sum_{j=1}^{i} \frac{\int_{T_{j-1}}^{T_j} r dt}{r} = T_1 + (T_2 - T_1) + \cdots + (T_i - T_{i-1}) = T_i$$

■

E. *Optimization problems*

Total infection time is the total amount of time during which some worm traffic is still present in the network. The worm aims to infect the largest number of nodes in the fastest possible time. There is an intrinsic tradeoff between these two objectives. The two problems defined below correspond to optimizing one objective while keeping the other as a threshold constraint. For example, we may want to minimize the infection time given that at least 90% of infectable targets are infected; or we may want to maximize the expected number of infected targets within 5 seconds.

Problem 1: (**Minimum Time Malicious Propagation –** MTMP) Given a lowerbound on the expected number of infected nodes, find an infection topology and the corresponding schedules minimizing the expected infection time.

Problem 2: (**Maximum Expansion Malicious Propagation –** MEMP) Given an upperbound on the expected infection time, find an infection topology and the corresponding schedules maximizing the expected number of infected nodes.

This paper focuses on the first problem, leaving the second problem for future research. The analytical model is quite simple, yet MTMP is already $\mathbf{NP}$-hard.

Theorem 2: When the latencies L_{wv} are not uniform, MTMP is $\mathbf{NP}$-hard even for $p = 0$.

Proof: We reduce SET COVER to MTMP. Consider an instance of the decision version of SET COVER where we are given a collection $\mathcal{S}$ of m subsets of a finite universe U of n elements, and a positive integer $k \leq m$. It is $\mathbf{NP}$-hard to decide if there is a set cover of size at most k.

An instance of MTMP is constructed as follows. Set $a = W = c$, where c is an arbitrary integer as long as $\lg c$ is a polynomial in m and n, so that c can be computed in polynomial time. The set of targets is $V = \{v_0\} \cup \mathcal{S} \cup U$, where v_0 is the initially infected node. The up- and down-link bandwidths are as follows.

$$\begin{aligned} r_{v_0} &= r_S^{(d)} = r_1 := c, \quad \forall S \in \mathcal{S} \\ r_S^{(u)} &= r_e = r_2 := 2nc, \quad \forall S \in \mathcal{S}, \forall e \in U. \end{aligned}$$

(Recall that, for any node $v \in V$, $r_v = r$ means $r_v^{(u)} = r_v^{(d)} = r$.) The latencies are:

$$\begin{aligned} L_{v_0 S} &= L_1 := 1, \quad \forall S \in \mathcal{S} \\ L_{Se} &= L_2 := m - k, \quad \forall S \in \mathcal{S}, \forall e \in S \\ L_{vw} &= L := m + n + 2. \text{ for all other pairs of nodes } (v, w). \end{aligned}$$

To complete the proof, we will show that the SET COVER instance has a set cover of size at most k if and only if the MTMP instance constructed above has a propagation topology and schedule with total infection time at most $(m+n+3/2)$.

For the forward direction, suppose there is a sub-collection $\mathcal{C} \subseteq \mathcal{S}$ of at most k members such that $\cup_{S \in \mathcal{C}} = U$. Since $\mathcal{C}$ is a set cover, we can choose arbitrarily for each member $S \in \mathcal{C}$ a subset $T_S \subset S$ such that $\bigcup_{S \in \mathcal{C}} T_S = U$ and the T_S are all disjoint. (This can be done with a straightforward greedy procedure.)

Consider the propagation topology $G = (V, E)$ defined as follows. The root v_0 will infect all nodes in $\mathcal{S}$, i.e. $(v_0, S) \in E$, for all $S \in \mathcal{S}$. Each node S in the cover $\mathcal{C}$ infects the nodes $e \in T_S$, namely $(S, e) \in E$, for all $S \in \mathcal{C}$ and $e \in T_S$. Now, the transmission schedule for the root v_0 is such that v_0 infects all nodes in $\mathcal{C}$ first in any order, and then all other nodes in $\mathcal{S} - \mathcal{C}$. Then, for each $S \in \mathcal{C}$, S infects nodes in T_S in any order. The time it takes for the last node in $\mathcal{S}$ to be infected is $T_1 = (na + mW)/r_1 + L_1 = n + m + 1$. The last node S in $\mathcal{C}$ will receive the worm W and its data (for nodes in T_S) at time at most $(na + kW)/r_1 + L_1 = n + k + 1$. Up on receiving the worm, each node $S \in \mathcal{C}$ will infect nodes in T_S, which takes time at most $|T_S|W/r_2 + L_2 \leq nW/r_2 + L_2 = 1/2 + m - k$. Because these infections happen as soon as each node S receives its worm, the last node in U receiving the worm at time at most $T_2 = (n + k + 1) + (1/2 + m - k) = m + n + 3/2$. Thus, the total infection time is at most $\max\{T_1, T_2\} = m + n + 3/2$, as desired.

Conversely, suppose there is a propagation topology $G = (V, E)$ and some transmission scheduling such that the total infection time is at most $m + n + 3/2$. Note that $(v_0, e) \notin E$ for all $e \in U$, because the latency $L_{v_0,e}$ is $m + n + 2 > m + n + 3/2$. For the same reason, $(S_1, S_2) \notin E$ for any $S_1, S_2 \in \mathcal{S}$; $(e, S) \notin E$ for any $e \in U$ and $S \in \mathcal{S}$; and if $e \notin S$, then $(S, e) \notin E$. Consequently, the only possible edges of G are of the form (v_0, S) for $S \in \mathcal{S}$, and (S, e) for $e \in S$. Now, let $T_S = \{e \mid (S, e) \in E\}$ be the set of out-neighbors of S in G. Let $\mathcal{C} = \{S \mid T_S \neq \emptyset\}$ be the set of S with non-zero out-degrees. It is clear that $\mathcal{C}$ is a set cover of the original SET COVER instance, otherwise not all nodes in U are infected. We show that $\mathcal{C}$ has at most k members. Suppose $\mathcal{C}$ has at least $k + 1$ members, then the last member S of $\mathcal{C}$ receiving the worm at time at least $T_1 = (na + (k + 1)W)/r_1 + L_1 = n + k + 2$. This last member will have to infect nodes in T_S (there is at least one node in this set), which takes time at least $T_2 = W/r_2 + L_2 = 1/(2n) + m - k > m - k$. Consequently, the total infection time is at least $T_1 + T_2 > n + m + 3/2$. ■

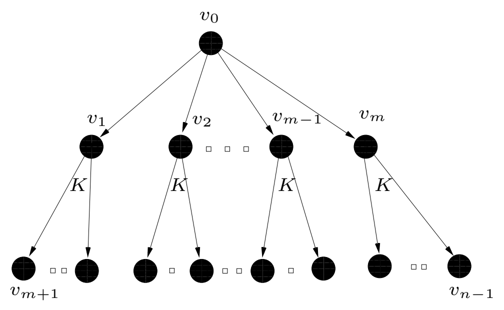


Fig. 1. Infection topology of a Flash worm in [1]

F. An example

Next, let us next revisit the infection scheme of the non-resilient UDP Flash Worm presented in Section 2 of [1] to illustrate the model discussed above. Fig. 1 depicts the infection topology of this scheme, which is a tree whose root is the source v_0. The source first infects m intermediate nodes (from v_1 to v_m), each of which continues to infect K other nodes. The infection schedule was not provided in the original paper. For simplicity, we assume all nodes have the same up- and down-link bandwidths of r. Thanks to Theorem 1, we can assume a parent node infects its children in any sequential order. Finally, as in [1], the pairwise latencies are all the same, denoted by L. Noting that $n = m(K+1)+1$, the propagation

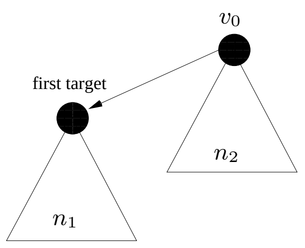


Fig. 2. The Recursive Tree topology, $p = 0$ case.

time for this topology is

$$\begin{aligned} t_1 &= \frac{m(W+aK)}{r} + L + \frac{KW}{r} + L \qquad (1) \\ &\geq \frac{(n-1)a}{r} + 2L - \frac{W}{r} + 2\frac{\sqrt{W(W-a)(n-1)}}{r} \end{aligned}$$

III. AN EVEN FASTER FLASH WORM

When the latencies are not uniform, the general MTMP problem is **NP**-hard by Theorem 2. This result justifies a further refinement of the model where we use the average bandwidth r as up- and down-link bandwidth, and the average latency L as the pair-wise latency. This section demonstrates that this simpler model already allows us to design a faster Flash worm which requires much less bandwidth at the source.

We focus on minimizing the infection time assuming no infection failure, deferring the failure-prone case to Section IV. In this case, any topology in which all targets are reachable from the root would infect the entire population. Moreover, each node needs to be infected only once, implying that a tree is sufficient. Thanks to Theorem 1, we only need to consider non-preemptive schedules.

Suppose the source first infects the root of a sub-tree of size n_1 (see Figure 2). The tree on n nodes can be viewed as a combination of two sub-trees of size n_1 and $n_2 = n - n_1$. Let $T(n)$ be the minimum infection time for n nodes. Then $T(n)$ can be recursively computed:

$$\begin{aligned} T(n) &= \min_{n_1 \leq n-1} \left\{ \frac{W + a(n_1 - 1)}{r} + \max\{T(n_1) + L, T(n_2)\} \right\} \\ &= \min_{n_1 \leq \lfloor n/2 \rfloor} \left\{ \frac{W + a(n_1 - 1)}{r} + \max\{T(n_1) + L, T(n_2)\} \right\} \qquad (2) \end{aligned}$$

$T(n)$ can be computed in $O(n^2)$-time. A newly infected node v does not need to recompute the value n_1 for its sub-tree (of size $|S_v|$). The optimal choices of n_1 for different values of n can be pre-computed and then be transmitted along with the address list. This strategy adds a fixed number of bytes (≤ 4) to be transmitted per target, and thus can be included in the block of a bytes per target.

Figures 3(a) and 3(b) compare the infection times of this *recursive tree topology* with the Flash Worm topology as the population size varies. Two values for W are observed – 404 and 1200 bytes – corresponding to actual sizes of Slammer and Witty [21], [22]. As seen in the figure, our worm infection time scales very well with the population size. We also simulated the worm traffic generated during the propagation process. Figure 3(c) summarizes the total traffic for the recursive topology with 4 average values of L. For this simulation, we set $n = 1$ million nodes. As can be seen from the figure, the total traffic has a peak value of 400Mbps, independent of the latencies. This number is much smaller than the traffic generated by some actual worms such as Slammer (165 Gbs), and also smaller than that of the Flash Worm in [1]. For this amount of total traffic the worm is less likely to cause any significant instability in the network core, validating our assumption for the analytical model.

The time difference between the two topologies is reduced as L increases. When L is large, the optimal tree based on (2) becomes shallower, thus performs more like the Flash Worm topology. Intuitively, when the propagation time is too large the source can actually send all worm packets to all targets within the propagation delay of the first packet. The trend can be explained with a simple analysis. At one extreme, when $L > \frac{W(n-1)}{r}$ equation (2) sets $n_1 = 1$ for any n, yielding a star topology. At the other extreme, when $L = 0$ or very close to zero, the optimal topology is a highly unbalanced tree, as shown by the following proposition, whose inductive proof is omitted.

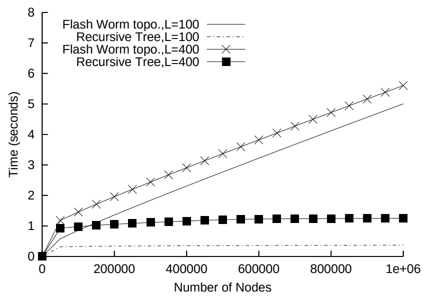
Proposition 3: When $L \to 0$, we have

$$T(n) = \lceil \log n \rceil \left(\frac{W-a}{r} \right) + (n-1)\frac{a}{r} \qquad (3)$$

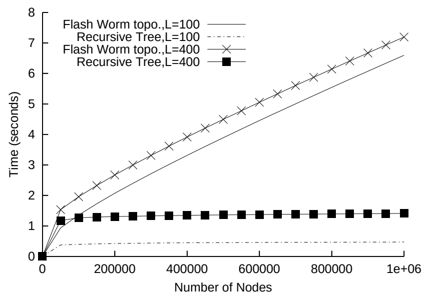
which is attained at $n_1 = \lfloor \frac{n}{2} \rfloor$.

To cope with the obstacle of **NP**-hardness, we refined our analytical model by assuming uniform bandwidths and latencies. Does the above analytical comparison result holds in practice? In the bandwidth case, if the uniform bandwidth is taken to a lowerbound of all actual bandwidths, then the analytical infection time computed from the model is a worst-case infection time bound. The uniform latency assumption, however, might be too strong.

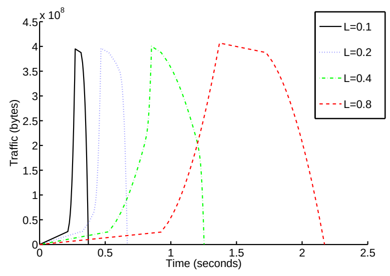
To address this doubt, we simulated the propagation of the worms in a network with varied latencies. These latencies were generated in accordance with the empirical latency distribution in the Skitter data set [23]. We generated 100 sets of latencies, corresponding to 100 network configurations. For each network configuration, we computed the recursive topology based on (2), setting L equal to the average Internet latency, which is about 201ms. Due to enormous time consumption, we were only able to run the simulation with $n = 100,000$ nodes instead of 1 million nodes. We kept $r = 1$ Mbps as before. We then simulated and compare the Flash Worm and the Recursive Tree topologies on the same network. Figure 4(a) plots the ratio of the simulated infection time of the Flash Worm topology over that of the Recursive Tree topology over 100 simulations with several values of latency variances. Firstly, it can be seen that the time-improvement ratio is roughly close to that of the same analytical data point (at $n = 100,000$) shown in Figure 3. Secondly, as expected the smaller the latency variance, the better the time-improvement ratio. This means that our worm will work well if the target population does not have many clusters of near-by nodes.



(a) L = 100 ms, 400 ms, W = 404 Bytes

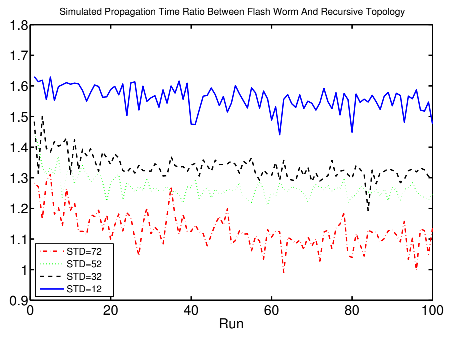


(b) L = 100 ms, 400 ms, W = 1200 Bytes

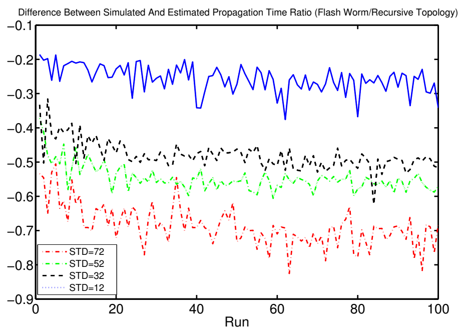


(c) Traffic during propagation, 1M hosts

Fig. 3. Analytical infection times of Flash Worm and Recursive Tree topologies with varying n

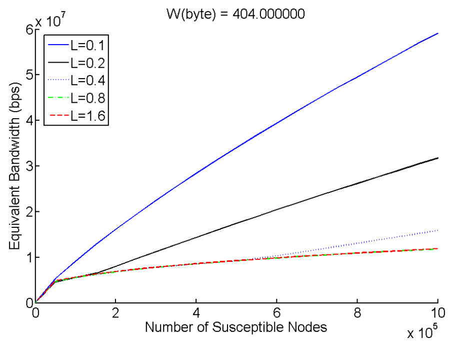


(a) Infection time of Flash worm in [1] over our worm (time improvement ratio)

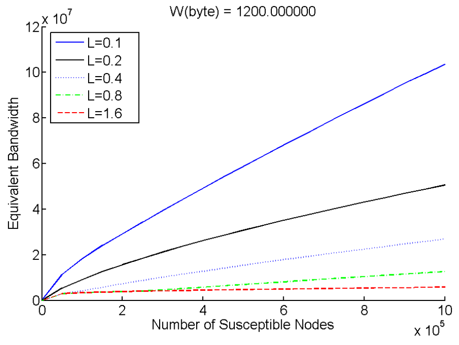


(b) Difference between simulated and analytical time improvement ratios

Fig. 4. The simulated infection times of Flash Worm and Recursive Tree topologies with $n = 100,000$



(a) $W = 404$ bytes



(b) $W = 1200$ bytes

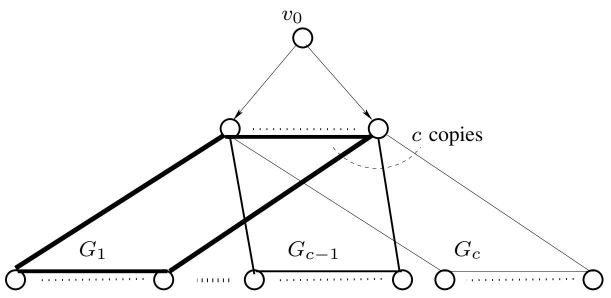
Fig. 5. Minimum source bandwidth for the Flash Worm topology to achieve the same effect as our worm

Furthermore, to see the effect of the latency variance, Figure 4(b) shows the difference between the analytical time-improvement ratio and the simulation time-improvement ratio. The analytical time-improvement ratio is a slight overestimate for small variances, and gets worse as the variance increases. However, it is still well within a small constant time the real time-improvement ratio.

Another advantage of the recursive topology is that it can retain a similar time efficiency as the Flash Worm of [1] even when starting from a root node with much less bandwidth capacity. Figures 5(a) and (b) show the minimum bandwidth at the root of the Flash Worm in [1] required in order for the Flash worm to propagate as fast as our worm, whose starting node has only 1Mbps capacity. The figures plot this required root's bandwidth as a function of the total number of nodes n for two empirical values of W. We also varied L to see the effect of latencies on the efficiency of the Flash worm and our worm. As can be seen, the result is consistent with our previous analysis. The required root's bandwidth for Flash worms stays at peak for small values of L and reduces gradually as L increases. In particular, at $L = 0.1s$, the Flash worm needs a significantly larger bandwidth of 60 Mbps (compared to the uniform bandwidth 1 Mbps using our recursive topology). This number even grows to more than 100 Mbps when W increases, as shown in Figure 5(b).

IV. The Tradeoff Between Speed and Resiliency

For each infection topology G on n nodes, let $N(G)$ and $T(G)$ be the expected number of infected nodes and the



The G_i are all copies of the same graph H

Fig. 6. The Resilient Topology R

expected total infection time, respectively. For any particular G, increasing $N(G)$ forces $T(G)$ to increase also. This is the tradeoff we investigate. Define

$$\begin{aligned} N(n) &= \max\{N(G) \mid G \text{ is an infection topology on } n \text{ nodes}\}, \\ T(n) &= \min\{T(G) \mid G \text{ is an infection topology on } n \text{ nodes}\}. \end{aligned}$$

The following proposition follows straightforwardly.

Lemma 4: Let S be a star on n nodes rooted at v_0. Then, S is the topology which maximizes $N(G)$. In particular, $N(S) = N(n) = 1 + (n-1)(1-p)$.

Knowing the maximum expectation $N(n)$, we can use it as a benchmark to characterize the tradeoff between $N(G)$ and $T(G)$ for any infection topology G. Note that, while $N(S)$ is optimal $T(S) = (n-1)W/r + L$ is not. A natural problem is to find a topology G for which $N(G)$ is very close to the optimal $N(S) = N(n)$, say more than a factor f of $N(S)$, while $T(G)$ is much smaller than $T(S)$. An obvious strategy is to choose any subset of $n' < n$ nodes and apply topology S on n' nodes, where $n' \approx fn$. This way, the sacrifice in $N(n)$ is linear and the payoff in $T(n)$ is also linear. Fortunately, there is a much better strategy than this simple-minded approach. We will show that, to achieve a linear payoff in $T(n)$, we can still keep $N(n)$ exponentially close to optimal! The result, in a sense, is the best one can hope for. (The converse is also desirable, where a linear sacrifice in $N(n)$ gives an exponential payoff in $T(n)$. This problem is open!)

Our infection topology named the *resilient topology* R is illustrated in Figure 6. For $i = 1, \dots, c$, let $G_i = (X_i \cup Y_i; E_i)$ be copies of the same bipartite graphs H. (H to be specified.) To construct R, we "glue" together the X_i-side of $G_1, \dots, G_c$, i.e. identify vertices in the X_i in any one-to-one manner, as shown in Figure 6. Let X denote the glued X_i, and $Y = Y_1 \cup \cdots \cup Y_c$. The source v_0 has an edge to each vertex in X. In total, the vertex set of R is $\{v_0\} \cup X \cup Y$. Different choices of the "seed component" H lead to different degrees of resiliency, as explained in the following theorems.

In the following theorem, we consider a very simple case to illustrate the idea and the complexity of analyzing the expected infection time. The proof is quite tedious and thus omitted due to space limit.

Theorem 5: Let H (i.e. the G_i) be any k-regular simple bipartite graph. Let $l = \left\lceil \frac{kc + Lr/W}{kca/W + 1} \right\rceil$, we have

$$N(R) = (n-1)(1-p)\left(1 - \frac{c}{1+c}p^k\right) + 1. \tag{4}$$

$$T(R) = \frac{W + kca}{r}\left[x(1 - p^l + p^{2l})\right] + Lp^l + \frac{W + kca}{r}\left[lp^l + \frac{p(p^l - 1)}{1-p}\right](1 - p^l) + (1 - p^l)^2\left(\frac{kcW}{r} + 2L\right). \tag{5}$$

$$\lim_{n\to\infty} \frac{T(R)}{T(S)} = (1 - p^l + p^{2l})\left(\frac{1}{1+c} + \frac{kca/W}{1+c}\right). \tag{6}$$

In particular, for sufficiently large n, topology R has the expected number of infected nodes exponentially close to optimality, yet reduces the expected infection time by a linear factor of $\frac{1 + c(ka/W)}{1+c}$.

The limit (6) tells us roughly how far away from $T(S)$ our infection time is. The limit ratio is simpler to work with than the actual ratio, which is dependent on n. Moreover, when n is in the order of tens of thousands, the limit ratio accurately depicts the actual ratio. (We omit the plot due to space limit.)

We next illustrate how this theorem can be applied. To reduce the infection time for this topology, we want the limit (6) to be as small as possible, subject to some desired threshold in terms of the expected number of infected nodes. For instance, to guarantee that $N(R) \geq fN(n)$, then we choose k and c to minimize the limit (6), subject to the condition that

$$(n-1)(1-p)(1 - \frac{c}{c+1}p^k) + 1 \geq f[(n-1)(1-p) + 1],$$

which is equivalent to

$$k \geq \frac{-\ln\left[(1-q)(c+1)/c(\frac{1}{(n-1)(1-p)} + 1)\right]}{\ln(1/p)} \tag{7}$$

This can be done in a variety of ways, one of which is to choose a relatively large c (thus reducing the infection time), then choose k to satisfy constraint (7). This lower bound for k is relatively small. For most values of c the lower bound for k is at most 4, even when f is large (90% or more).

A stronger notion of resiliency

While $N(R)$ within a fraction f of the optimal is a good notion of resiliency (the same notion as that in [1]), it is still a weak guarantee in the following sense. The ratio between $N(R)/N(n)$ is large when the expectation $N(R)$ is large. This expectation is a weighted-average in accordance with the failure distribution of the targets imposed by p. This means that many node failure combinations still yield a **smaller** number of infected nodes than $N(R)$. It would be nice to have a stronger guarantee than that.

In what follows, we will show that, with the right choice of H, for any given $\epsilon > 0$ the number of infected nodes is within a fraction $(1-\epsilon)$ of the number of infectable nodes with probability exponentially close to 1.

A (w, ϵ)-*extractor* is a bipartite graph $H = (L \cup R, E)$ with left part $|L| = x \geq w$ and right part $|R| = y$, such that every subset of at least w left vertices has at least $(1-\epsilon)y$

neighbors on the right. It is known that, for any ϵ and any $w \leq x$, there exist (w, ϵ)-extractors in which all left vertices have degree $d = \Theta(\log x)$, $y = \Theta(wd)$, and the distribution of right degrees are close to uniform, i.e. of degree $xd/y = \Theta(x/w)$ [24].

Theorem 6: Fix any constant $\epsilon > 0$. Let $x = n/(c+1)$ for any chosen constant c as before. Let $\alpha > 0$ be any constant such that $\alpha < 1 - p$. Set $w = \alpha x$, and let H be the (w, ϵ)-extractor as described above. Then, with probability at least $1 - \exp(-\Theta(n))$ (i.e. exponentially close to 1), topology R will be able to infect a $(1 - \epsilon)$-fraction of all nodes that are infectable.

Proof: [**Sketch**] The probability that at least αx vertices is X are infected is exponentially close to 1. This can be obtained with a typical Hoeffding-Chernoff inequality to bound the tail of the binomial distribution. From these αx vertices, at least $(1 - \epsilon)$-fraction of vertices in Y will be reached. ■
Note that, the degree of vertices in Y is $\Theta(x/w) = \Theta(1/\alpha)$, which is a constant. Hence, the linear time improvement factor over topology S still holds!

V. Discussions and Future Works

We have not specified how to estimate bandwidths and delays. Recent works on end-to-end Internet measurements point to several answers to this question [25], [26].

As a solution to our problem, propagation topology R is a decent topology in the sense that it sacrifices the expected number of infected targets a little bit, while it improves the expected infection time relatively well. However, there is more room for improvement. The main reason we chose a shallow topology to analyze is the feasibility of its analysis. Taking the lesson from the new design in Section III, we should look at more unbalanced design using the same kind of seed component composition, where the topology is deeper in one branch than the other.

The key idea behind a resilient propagation topology is that the graph should be "expanding" to allow for concurrent propagation. The obvious choice would seem to be some sort of expanders [27], which are graphs with very high connectivity and relatively low diameters, thus reducing propagation time while keeping a high level of resiliency. The bipartite extractors are one kind of expanding graphs. The extension of this idea to use general expanders is open for further research.

We have shown that the problem MTMP is **NP**-hard. Another open problem is thus to devise a good approximation algorithm for this problem. If the approximation ratio is sufficiently good, the difference between the optimal solution (say, a few milliseconds), and the approximated solution (say, a few more milliseconds) is practically insignificant. It is important to also study how current containment policies such as that in [11] can thwart these infection schemes. Last but not least, the second problem we formulated – MEMP – has not been addressed at all.

References

[1] S. Staniford, D. Moore, V. Paxson, and N. Weaver, "The top speed of flash worms," in *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*. New York, NY, USA: ACM Press, 2004, pp. 33–42.

[2] D. Moore, C. Shannon, and J. Brown, "Code-red: a case study on the spread and victims of an internet worm," in *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. New York, NY, USA: ACM Press, 2002, pp. 273–284.

[3] I. Arce and E. Levy, "An analysis of the slapper worm," *IEEE Security and Privacy*, vol. vol. 1, no. 1, pp. 82–87, 2003.

[4] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," *IEEE Security and Privacy*, vol. vol. 1, no. 4, pp. 33–39, 2003.

[5] C. Shannon and D. Moore, "The spread of the witty worm," *IEEE Security and Privacy Magazine*, vol. 2, no. 4, pp. 46–50, 2004.

[6] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "The spread of the sapphire/slammer worm," http://www.caida.org/publications/papers/2003/sapphire/sapphire.html, 2003.

[7] Z. Chen, L. Gao, and K. Kwiat, "Modeling the spread of active worms," in *INFOCOM 2003*, vol. 3, Mar 30 - Apr 3 2003, pp. 1890–1900.

[8] C. C. Zou, W. Gong, and D. Towsley, "Code red worm propagation modeling and analysis," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 138–147.

[9] C. C. Zou, W. Gong, D. Towsley, and L. Gao, "The monitoring and early detection of internet worms," *IEEE/ACM Trans. Netw.*, vol. 13, no. 5, pp. 961–974, 2005.

[10] D. Nojiri, J. Rowe, and K. Levitt, "Cooperative response strategies for large scale attack mitigation," in *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, vol. 1, 2003, pp. 293–302.

[11] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Internet quarantine: requirements for containing self-propagating code," in *INFOCOM 2003*, vol. 3, 2003, pp. 1901–1910.

[12] C. C. Zou, W. Gong, and D. Towsley, "Worm propagation modeling and analysis under dynamic quarantine defense," in *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, 2003, pp. 51–60.

[13] M. Liljenstam, D. M. Nicol, V. H. Berk, and R. S. Gray, "Simulating realistic network worm traffic for worm warning system design and testing," in *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*. New York, NY, USA: ACM Press, 2003, pp. 24–33.

[14] C. C. Zou, D. Towsley, W. Gong, and S. Cai, "Routing worm: A fast, selective attack worm based on ip address information," in *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 199–206.

[15] S. Staniford, V. Paxson, and N. Weaver, "How to own the internet in your spare time," in *Proceedings of the 11th USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2002, pp. 149–167.

[16] B. McCarty, "Botnets: big and bigger," in *IEEE Security and Privacy Magazine*, vol. 1, no. 4, 2003, pp. 87–90.

[17] "http://www.icir.org/yoid/."

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *SIGCOMM 2001*, 2001, pp. 161–172.

[19] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *SIGCOMM 2002*. New York, NY, USA: ACM Press, 2002, pp. 205–217.

[20] A. Odlyzko, "Data networks are lightly utilized, and will stay that way," *Review of Network Economics*, vol. 2, no. 3, pp. 210–237, September 2003.

[21] http://www.f-secure.com/v-descs/mssqlm.shtml.

[22] http://www.f-secure.com/v-descs/witty.shtml.

[23] CAIDA, "Skitter datasets," http://www.caida.org/tools/measurement/skitter/.

[24] C.-J. Lu, O. Reingold, S. Vadhan, and A. Wigderson, "Extractors: optimal up to constant factors," in *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*. New York: ACM, 2003, pp. 602–611 (electronic).

[25] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz, "Towards an accurate as-level traceroute tool," in *SIGCOMM 2003*. New York, NY, USA: ACM Press, 2003, pp. 365–378.

[26] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, 2004.

[27] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc. (N.S.)*, vol. 43, no. 4, pp. 439–561 (electronic), 2006.