On the trade-off between the expected number of infected nodes and expected propagation time of malcodes
Technical Report #2007-05,Department of Computer Science and Engineering, SUNY at Buffalo

Duc T. Ha, Hung Q. Ngo

Computer Science and Engineering Department, SUNY Buffalo, Amherst, NY 14260 USA,
{ducha,hungngo}@cse.buffalo.edu

Abstract. We investigate the problem of finding a fast and resilient propagation topology and propagation schedule for worms and similar malcodes, given that the malcode knows a little more information about the targets than just the IP addresses. Resiliency means that the malcode is still able to propagate to a large expected number of targets even when some targets are not infectable.

We first show that, under a moderately general network model, the problem of optimizing propagation time is **NP**-hard. This fact justifies the need for a refinement of the network model, which we present next. In the refined model, when all nodes are infectable we present an optimal propagation topology. 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.

With the presence of uncertainty (some targets may fail to be infected with some given probabilities due to network errors, security patches, etc.), there is a natural trade-off between the expected total infection time and the expected number of infected targets. We investigate this trade-off by deriving the optimal expected number of infected nodes and the corresponding propagation topology. Next, we give a propagation topology and schedule which can reduce the infection time significantly while keeping the expected number of infected nodes exponentially close to optimal.

To the best of our knowledge, this work is the first in the literature to formulate and address the aforementioned trade-off, and also the infection time optimization problem on a formal basis. We expect our problem formulations and solutions to have applications in similar application-level source broadcast problems, such as ones in P2P and overlay networks. Last but not least, our formulations leave open a number of interesting and challenging problems.

1 Introduction

Since the first Internet worm incident in 1988 [10], network worms have evolved considerably. Current worms are very speedy and destructive, posing a major problem to security researchers. The spread of worms is often fast enough to render existing signature-based anti-virus programs and intrusion detection systems useless. The scale

and scope of the worms' potential destructive impact pose serious problems not only for local organizations or even for a single country, but also for the global Internet community (the larger the scale of the Internet and the more popular some computing platforms become, the more costly and more dangerous worms will be!).

In fact, the recent years since 2001 have arguably been the golden age of worms. Multiple varieties of worms have appeared and brought unpleasant surprises to the Internet community, including CodeRed [19], Nimda, Slapper [4], Slammer [17] and Witty [24]. Consequently, researchers have spent a considerable amount of effort to study worms' mechanics and dynamics, such as modeling the dynamics of worms [9, 29], monitoring and detecting worms[31], developing automated worm containment mechanisms [21, 20, 30], simulating worm traffic [14], routing worms [32], 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. [27] were the first to investigate this kind of hypothetical scenario. They later elaborated on a specific worm instance called "Flash worm" [26], so named since these worms can span the entire susceptible population within an extremely short time frame. (Our paper is inspired by [26, 27]!) As a matter of fact, fast worms/malcodes no longer just hide in the theoretical domain. The UDP-based Slammer worm spread at an unprecedented rate: 90% of susceptible population within 10 minutes [18]. Witty infected 110 hosts within 10 seconds. A super-fast worm is now a very real and practical possibility!

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 the 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, both for 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 1 million hosts, for example, the IP address list requires roughly 4MB (for IPv4). This much data if integrated within each worm instance and transmitted without an efficient distribution scheme will severely impact the speed of propagation. Thirdly, making the worm resilient 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 much 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 by Staniford [26] requires an initial node that can deliver 750Mbps. Compromising a host with that much bandwidth may not always be an option. A natural question that follows is whether the attacker can create the same or similar effects as flash worms with limited resources. This paper will answer this question in the affirmative.

Consequently, designing a worm propagation topology and schedule that provides both 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 decent estimates of a few parameters affecting the worm's speed, such as end-to-end delays and up-/down-link bandwidths of the intended targets, 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 have their security holes patched or may simply be down, and worm packets may be lost. If the worm writer has some estimate of the infectability probabilities, how will this knowledge be used to make the worm more resilient to the glitches?
- There is an inherent trade-off 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 some node v may 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 trade-off. 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 solutions unravel some layers of complexity of the problem and open a door for further exploration. Note that, the aforementioned trade-off is not just a by-product of the worm propagation problem. Efficient and error-resilient broadcast is fundamental in most network applications [3, 23, 5]. However, even though the objectives of efficiency and error-resiliency are similar, the operating constraints are very different between our problem and application-layer broadcast problems.

Our main contributions are as follows:

- We first show that, under a moderately general network model, the problem of optimizing propagation time is $\mathbf{NP}$-hard. This fact justifies the need for a refined yet still realistic network model, which we present next. Later simulation results further validate the refined model.

- In the refined network model, when all nodes are infectable, we present an optimal propagation topology and schedule. We shall show that it is possible to devise worm propagation topologies and schedules with infection time even shorter than the Flash worms, provided that the worm designer has decent guesses of a few network parameters of the targets. Moreover, it is also possible to retain the swift effect of Flash worm when starting from a root node with much less bandwidth capacity.
- We also show that for every preemptive propagation schedule (i.e. a node can interrupt the transmission to a target, starting transmission to other targets before resuming the initial transmission), there is a non-preemptive schedule (namely each transmission is not interrupted until it is finished) which is just as good. This fact greatly simplifies the optimization problem. However, this result does not apply to transmission processes with interactive communication between two ends such as the 3-way handshake in TCP. We will only consider UDP malcodes in this paper.
- Under uncertainty, i.e. nodes may fail to be infected with some given probabilities, we investigate the trade-off 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 significantly while keeping the expected number of infected nodes exponentially close to optimal.

The rest of this paper is organized as follows. Section 2 motivates and formulates the problem in a rigorous manner, including the description of our network model. It is shown that the optimization problem on a simple network model is already **NP**-hard, justifying a further refinement of the network model, which is studied in Section 3 in greater depth. Section 4 presents our simulation results. Section 5 discusses some related works and future research directions.

2 Problem formulation

2.1 Network model

In order to address the above questions rigorously, we first need a formal model capturing various pieces of information about the underlying network. The most general network model is the exact topology of the Internet itself, along with precise information about routers, links, bandwidths, delays, connectivity, queue lengths, link/node error probabilities, etc. However, a network model which assumes complete knowledge of the Internet is both unrealistic and not very useful for our purposes.

Practically, although there has been a recent surge of research on measuring many topological properties of the Internet [15, 13, 28, 25], complete information is still out of reach, especially when measured with end-to-end means. More importantly, such a general model makes the problem very complex to the point of not being tractable. To illustrate this point concretely, we will show that our problem defined on a much less general network model is already **NP**-hard. Moreover, the description of an encompassing network model takes so much space that transmitting it would certainly slow down the worm propagation, defeating the purpose.

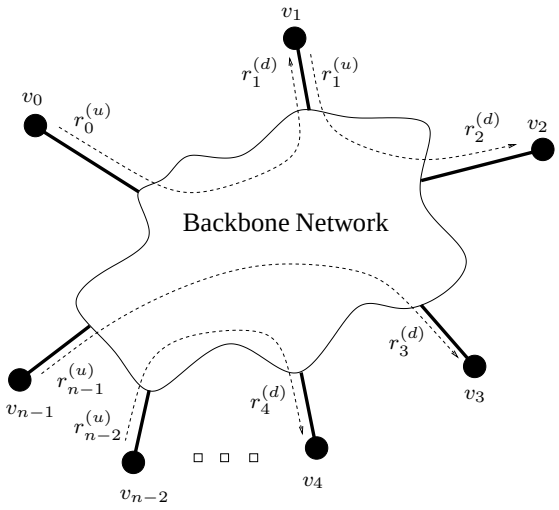


Fig. 1. A formal network model

A good network model has to be *realistic* and *manageable*. It should be *realistic* so that solving our problem on this model gives a good approximation of what actually happens in the real-world. It should be *manageable* (i.e. not too complex to the point of being useless) so that interesting and useful problems can be defined and solved on this model. We present a candidate for such network models in this section.

Consider the situation where there are n hosts, or nodes, and node v_0 is initially infected with the worm. The objective is to devise a propagation schedule and topology for the worm to infect the other nodes subject to constraints to be defined.

For each node v, let $r_v^{(u)}$ and $r_v^{(d)}$ denote v's up-link and down-link bandwidths, respectively. (For instance, v may be connected to the Internet with an ADSL service.) When the up- and down-link bandwidths are equal, we use r_v to denote this bandwidth. The effective bandwidth from node v to node w is then $\min\{r_v^{(u)}, r_w^{(d)}\}$. Figure 1 roughly illustrates our network model.

We assume that node v_0 (i.e. the worm writer) knows other nodes' bandwidths, via some sort of educated guess, pre-infection data gathering, or end-to-end bandwidth measurement. The task of v_0 is to devise the best propagation schedule making use of this knowledge. We will be more specific about our optimization objectives later.

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 two facts: (i) 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 [22], and (ii) when some of the worm's packets are lost, our resilient propagation topology helps alleviate the problem. Note that our worm does not generate scanning traffic. This significantly reduces the traffic intensity as compared to random-scanning worms. In

fact, using the propagation topologies presented in the next sections, we have calculated the total worm traffic at any given time in the network to be at most 500MBs or even less than 10MBs.

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 nodes. Each of these "blocks" of a bytes contains the address of a distinct target node (4 bytes for IPv4 and 16 bytes for IPv6), and perhaps additional information about distinct target nodes such as the up- and down-link bandwidths and/or end-to-end delays from the node to other nodes in the network. Unlike W, a depends on n theoretically. For example, to address n nodes we will need at least $\lg n$ bits. However, in practice we can safely assume that a (a few bytes) is much less than W (a few hundred bytes). Lastly, let p be the probability that a randomly chosen target is not infectable. Nodes are assumed to be infectable with independent probabilities.

To clarify the use of the above parameters, let us consider a typical worm infection scenario. Starting from v_0, which keeps a list of addresses and other information about the targets (bandwidths, delays), the worm picks out a subset of S target nodes to infect, along with a transmission schedule for infecting these nodes. Furthermore, v_0 will give each node v in S a set S_v of targets for v to infect on its own. This way, upon receiving the list 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. The process is completed when the last target node is infected. *Infection time* is the amount of time some worm traffic is still present in the network.

There are several natural optimization objectives which affect the way nodes choose subsets of targets to infect and subsequently the infection schedule. For example, we may want to minimize the infection time subject to the constraint that at least 90% of targets are infected. With the presence of uncertainty (i.e. $p \neq 0$), we may want to maximize the expected number of infected targets given a threshold on the expected infection time. Each infected node v which is delegated with a subset S_v of targets will use the same algorithm as the root v_0 for further infection. Thus, if we intend to make use of bandwidth and/or delay knowledge, this information will have to be passed to v along with the addresses of nodes in S_v. This explain why the parameter a contains not only the size of a target's address, but also some additional bytes representing further information about the target.

2.2 Rigorous problem definitions

We can use a directed acyclic graph (DAG) $G = (V, E)$ to model the way nodes choose subsets of targets to infect. The vertex set V consists of all target nodes, including v_0. There is an edge from v to w if v (after infected) is supposed to infect w. Since we do not need to infect an already infected node, a DAG is sufficient to model the infection choices. We refer to this DAG as the *infection topology*. Obviously, v_0 has in-degree 0; we thus refer to v_0 as the "root" of the infection topology.

Consider a node v in the G. Let S_v be the set of all nodes reachable from node v via a directed path. Clearly, S_v is the subset of targets that v was delegated the infection task to. (Note that, for distinct nodes v and w, S_v and S_w might be overlapping if we introduce redundancy to cope with infection failures.) Now, consider a parent node w of v, namely $(w, v) \in E(G)$. Node w is expected to infect v and give v the list S_v. Thus, each node 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. The subgraph will contain information about S_v for all "children" v of w.

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 nodes in S_v (i.e. in which order the children nodes must be infected). If nodes can regulate their transmission rates to a certain degree, the schedule can even be preemptive.

Note that, even though the root can presumably pre-compute the entire infection topology and transmission schedule for all nodes in the topology, 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 only look at worms whose code W is capable of computing its own infection topology and schedule given only the list of targets.

At this point, we are able to formally define our problems. The worm aims to infect the largest number of nodes in the fastest possible time. These two objectives form an intrinsic trade-off. The two problems defined below correspond to optimizing one objective while the other remains as a threshold constraint.

Problem 1 (Minimum Time Malicious Propagation – MTMP) *Given a lower-bound threshold on the expected number of nodes to be infected, find a propagation topology and the corresponding propagation schedule minimizing the expected infection time.*

Problem 2 (Maximum Expansion Malicious Propagation – MEMP) *Given an upper-bound threshold on the expected infection time, find a propagation topology and the corresponding propagation schedule maximizing the expected number of infected nodes.*

In this paper, we will focus on the first problem and leave the second problem open for future research. The general MTMP problem when the latencies are not uniform is **NP**-hard as shown in Theorem 1 (see Appendix A.1 for a proof). This result justifies a further refinement of the model.

Theorem 1 *When the latencies L_{wv} are not uniform,* MTMP *is* **NP***-hard even for $p = 0$*

2.3 A propagation topology revisited

In this section we revisit the 3-level tree infection scheme of a Flash Worm [26] in order to illustrate the ideas and problems discussed above. Figure 2 depicts the *infection topology* of this scheme, which is a directed 3-level 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 (in total, infect nodes from v_{m+1} to v_{n-1}). The *infection schedule*

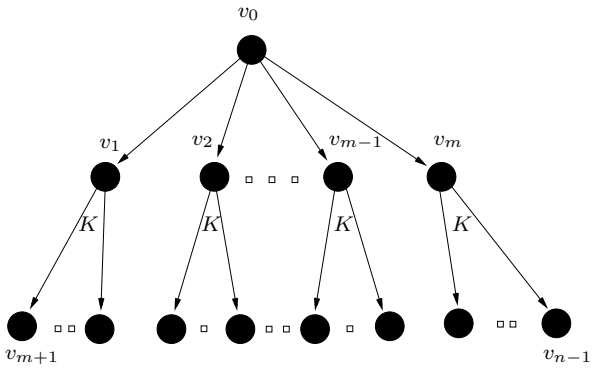


Fig. 2. A 3-level tree infection topology of a Flash Worm

was not provided in the original paper, but for our purpose, we can assume a parent node sends the infecting packets to its children in sequential order.

In this example, to illustrate the use of our network model we assume the source has the same bandwidth as the other nodes, and the up- and down-link bandwidths are the same for all nodes, and are equal to r. Finally, as in the original example, the pairwise end-to-end delays are the same, denoted by L. We can now compute the minimum propagation time for this topology. Note that $n = m(K+1)+1$ or $m = \frac{(n-1)}{(K+1)}$. The total infection time is then

$$t_1 = \frac{m(W+aK)}{r} + L + \frac{KW}{r} + L \geq \frac{(n-1)a}{r} + 2L - \frac{W}{r} + 2\frac{\sqrt{W(W-a)(n-1)}}{r}. \quad (1)$$

3 A refined network model: uniform bandwidth and latency

Since the general problem is hard, in this section we further refine the model by using the average bandwidth r as up- and down-link bandwidth, and the average latency L as the pair-wise latency.

In the example considered in Section 2.3, we implicitly assumed that a non-preemptive propagation schedule was used, namely each node infects each of its targets one at a time and sequentially. It turns out that for UDP-worms we do not need to consider preemptive schedules as Theorem 2 shows (a proof is in Appendix A.2). The theorem allows us to restrict our search for optimal schedules to the space of non-preemptive ones only.

Theorem 2 *For every preemptive schedule from a node to a set of targets, there is a non-preemptive schedule in which every target is infected at time no later than in the preemptive schedule.*

We next consider two scenarios: (a) all target nodes are infectable, and (b) some nodes could fail to be infected. Recall that we use p to denote the probability that a node fails to be infected, and that nodes fail to be infected independently.

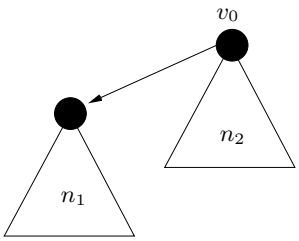


Fig. 3. Illustration of the tree topology for the $p = 0$ case.

3.1 The case of absolute certainty ($p = 0$)

When there is no error, any topology in which all targets are reachable from the root would be sufficient to infect the entire population. Thus we can focus on minimizing the infection time. If only some percentage of targets need to be infected, then we can further reduce the total infection time using the same technique developed here. Also note that, as there is no error each node needs to be infected only once. Therefore tree topologies (arborescence) are sufficient.

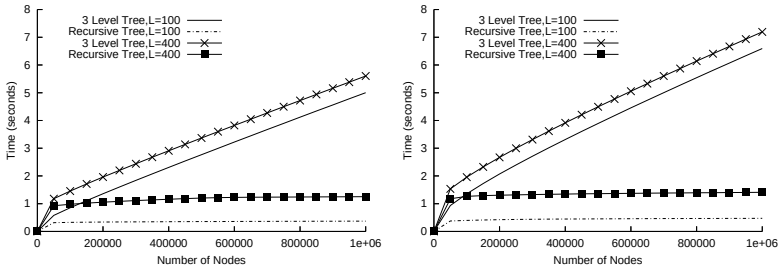
Thanks to Theorem 2, we only need to consider non-preemptive schedules. Every propagation scheme following a tree topology of n nodes can thus be viewed as a combination of two sub-trees of size n_1 and n_2. Note that n_1 and n_2 also count the root nodes, thus $n = n_1 + n_2$, as shown in Figure 3. Let $T(n)$ be the minimum total infection time for n nodes. Then $T(n)$ can be recursively computed in a straightforward manner as follows.

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

Here we use the fact that $T(n)$ is monotonically increasing, thus $T(n - n_1) \ge T(n_1)$ when $n_1 \le \lfloor \frac{n}{2} \rfloor$. It is easy to see that $T(n)$ can be computed within $O(n^2)$ time. Once the optimal value of n_1 is determined, the infecting node can forward the information about $n_1 - 1$ nodes in the first sub-tree to the first target and continue to infect the residual $n_2 - 1$ nodes analogously.

If every newly infected node recomputes its sub-tree topology, then this would pose several disadvantages to the infection process. Firstly, the computation makes the worm more prone to being detected due to prolonged resource consumption. Secondly, significant delay is added to the infection process, especially when n is large. One solution is to compute this information off-line before the infection process. The optimal values of n_1 corresponding to different values of n can then be transmitted along with the topology information. This strategy adds a fixed number of bytes to be transmitted per target, and thus can be included in the block of a bytes per target.

We now compare the infection time of this optimal topology with the 3-level tree topology in Section 2.3. We consider multiple variations of W and L to observe the



(a) L = 100 ms, 400 ms, W = 404 Bytes (b) L = 100 ms, 400 ms, W = 1200 Bytes

Fig. 4. The infection times of the 3-level topology and the recursive topology (topo. A)

difference. In particular, we use two values of W: 404 and 1200 bytes, corresponding to actual packet sizes of Slammer and Witty worms [1, 2]. Figure 4 shows the propagation time of the two topologies. As is demonstrated in the figure, the difference between two topologies is reduced as L increases. The reason for this is that when L is large, the optimal tree becomes shorter, making it closer to the 3-level tree topology. However, the 3-level tree is not optimal when L is sufficiently large (when L is very large, the optimal topology is a 2-level tree). On the other hand, the recursive equation (2) is guaranteed to yield the optimal topology.

The optimal value of n_1 according to (2) depends a lot on the value of L. At one extreme, when $L > \frac{W(n-1)}{r}$, we have $n_1 = 1$ for all n, yielding a two level tree topology whose root is the source. The intuition is that, when the propagation time is too large the source can actually send all worm packets to all targets within the time span for the first target to be infected. At the other extreme, when $L = 0$ or very close to zero, the optimal topology is a very unbalanced tree, as shown by the following result, whose proof is in Appendix A.3.

Theorem 3 *When $L = 0$, the optimal value of $T(n)$, attained at $n_1 = \lfloor \frac{n}{2} \rfloor$, is*

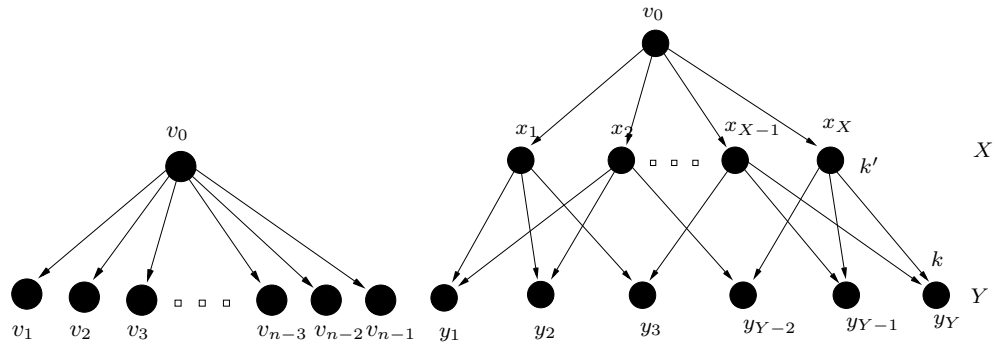
$$T(n) = \lceil \log n \rceil \left(\frac{W - a}{r} \right) + (n-1)\frac{a}{r}. \tag{3}$$

3.2 The case under uncertainty ($0 < p < 1$)

For each infection topology G on n nodes, let $\mathrm{E}_N^G(n)$ and $\mathrm{E}_T^G(n)$ be the expected number of infected nodes and the expected total infection time, respectively. Somewhat abusing notation we define

$$\mathrm{E}_N(n) = \max\{\mathrm{E}_N^G(n) \mid G \text{ is an infection topology on } n \text{ nodes}\},$$
$$\mathrm{E}_T(n) = \max\{\mathrm{E}_T^G(n) \mid G \text{ is an infection topology on } n \text{ nodes}\}.$$

The first question we address is: what is the maximum expected number of infected nodes, and which topology achieves this? This maximum expectation shall be used as a benchmark to investigate the trade-off between the expected number of infected



(a) Topology B: maximizing # infected nodes (b) Topo. C: $(2, k, k')$-resilient

Fig. 5. Infection topologies B and C.

nodes and the infection time of a topology. For instance, corresponding to the maximum expectation there is a propagation topology which has a fairly high infection time. If we only need E_N to be some percentage of the maximum E_N, is it possible to reduce E_T and if so how far can we reduce it? The next lemma, whose proof is in Appendix A.4, answers the first question.

Lemma 4 *The topology G which maximizes $\mathrm{E}_N^G(n)$ is a star rooted at the source. Moreover,* $\mathrm{E}_N(n) = 1 + (n-1)(1-p)$.

Figure 3.2(a) illustrates the topology of Lemma 4. We will refer to this topology as *topology B*. The expected infection time of B is then

$$\mathrm{E}_T^B(n) = (n-1)W/r + L. \tag{4}$$

This topology, while achieving the best expected number of infected nodes, is very inefficient in terms of infection time. It is now natural to address the following question: how much can we improve in terms of the infection time if we are willing to "let go" a few percents of the expected number of infected nodes? The most obvious strategy is to choose any subset of $n' < n$ nodes and apply the same topology B as above, where n'/n is the percentage of the optimal E_N we are willing to accept. This way, the reduction in E_N is linear and the reduction in E_T is also linear.

Fortunately, there is a significantly better strategy than the simple approach above. With a linear reduction in E_T, we can still keep E_N exponentially close to optimal! This is, in a sense, the best one can hope for. (The converse is also desirable, where a linear sacrifice in E_N gives an exponential reduction in E_T. We leave this problem open.) Our infection topology is described below.

We consider a topology of three levels (including the original source) in which each node x_i $(1 \le i \le x)$ in level 2 infects k' node y_j $(1 \le j \le y)$ in level 3. Each node y_j will be infected by k nodes x_i. This topology is similar to the example 3-level tree we consider in Section 2.3, but now each node at the last layer might be infected for more than once. Figure 3.2(b) illustrates the topology for $k' = 3, k = 2$. We will refer to this

topology as *topology* C. We have $x + y = n - 1$ and $k'x = ky$; thus, $x = \frac{n-1}{1+c}$ where $c = \frac{k'}{k}$. The expected number of infected nodes can be computed as follows.

$$\mathrm{E}_N^C(n) = x(1-p) + 1 + y(1-p)(1-p^k) = (n-1)(1-p)\left(1 - \frac{c}{1+c}p^k\right) + 1 \quad (5)$$

Computing $\mathrm{E}_T^C(n)$ is significantly more complicated, even for such a simple topology. The proof of the following lemma can be found in Appendix A.5.

Lemma 5 *We have*

$$\mathrm{E}_T^C(n) = \frac{W + k'a}{r}\left[x(1 - p^l + p^{2l})\right] + \frac{W + k'a}{r}\left[lp^l + \frac{p(p^l - 1)}{1-p}\right](1 - p^l) + Lp^l + (1-p^l)^2\left(\frac{k'W}{r} + 2L\right).$$

In particular, the following limit and theorem follows immediately.

$$\lim_{n\to\infty} \frac{\mathrm{E}_T^C(n)}{\mathrm{E}_T^B(n)} = (1 - p^l + p^{2l})\left(\frac{1}{1+c} + \frac{k'a/W}{1+c}\right). \quad (6)$$

Theorem 6 *For sufficiently large n, infection topology C yields an expected number of infected nodes exponentially close to optimality, yet reduces the expected infection time by a linear factor of $\frac{1}{1+c} + \frac{kca/W}{1+c}$.*

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, if we want the expected number of infected nodes to be at least a fraction q of the optimal $\mathrm{E}_N^B(n) = (n-1)(1-p) + 1$, then we need to choose our parameters k and k' to minimize the limit

$$(1 - p^l + p^{2l})\left(\frac{1}{1+c} + \frac{k'a/W}{1+c}\right) = (1 - p^l + p^{2l})\left(\frac{1}{1+c} + \frac{c(a/W)k}{1+c}\right),$$

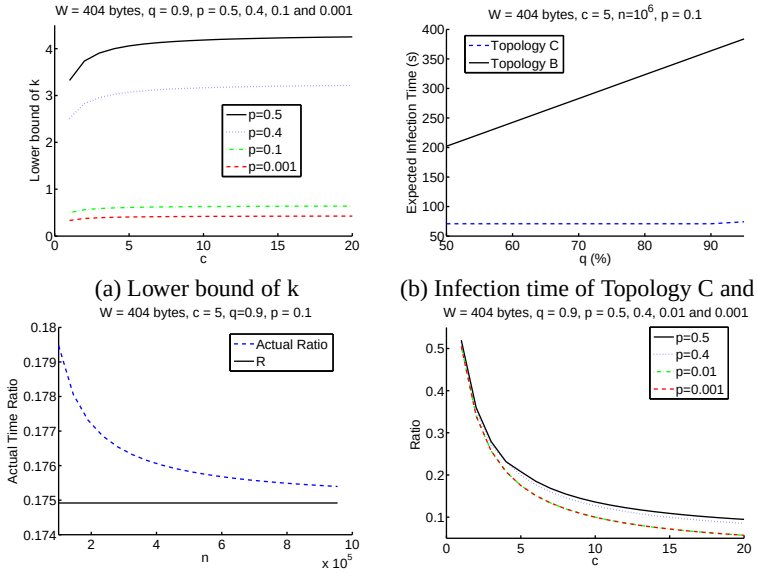
subject to the condition that

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

which is equivalent to

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

This can be done in a variety of ways, one of which is to choose a relatively large ratio $c = k'/k$ (thus reducing the infection time), then choose k to satisfy constraint (7). This



(a) Lower bound of k

(b) Infection time of Topology C and B

(c) Limit Ratio and Actual Ratio gap

(d) Limit Ratio dependency on p

Fig. 6. The infection time of 3-level and recursive topology with $W = 404$ bytes

lower bound, however, is relatively small if c is small, as shown in Figure 6(a).As can be seen from the figure, we can select $c < 10$ even when q is large (we set $q = 0.9$). We will select $c = 5$ for all other graphs.

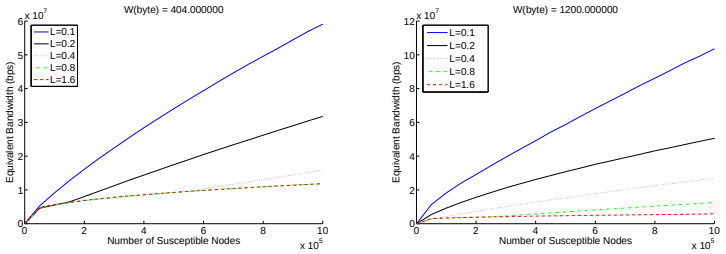
To have qn as the expected number of infected nodes, we could have used topology B with a sub-population of $n' = qn$ nodes only. However, this topology is not nearly as efficient as topology C, and the gap increases as q gets larger, as shown in Figure 6(b). Topology C, however, is efficient only if n is sufficiently large. Figure 6(c) shows the gap between the actual ratio compared to the limit ratio (6). This gap is small even when n is only about tens of thousands. For values of n at six figures, the gap is negligible and we need only to work with the limit ratio (6).

We also look into the dependence of the limit ratio on the infection failure probability p. Figure 6(d) shows the values of the limits with 4 different values of p: $0.5, 0.4, 0.01$, and 0.001. It can be seen that p doesn't have much effect on the ratio. As p becomes sufficiently small, the ratio is essentially independent of p.

4 Simulation results

To avoid the obstacle of **NP**-hardness, we refined our model by assuming uniform bandwidths and latencies. The obvious question is whether such a simple model can yield practically useful results.

For the bandwidth assumption, if the uniform bandwidth is taken to be the lower bound of all actual bandwidths, then the theoretical infection time computed from the model is a worst-case time bound for the worm designer.



(a) $W = 404$ bytes (b) $W = 1200$ bytes

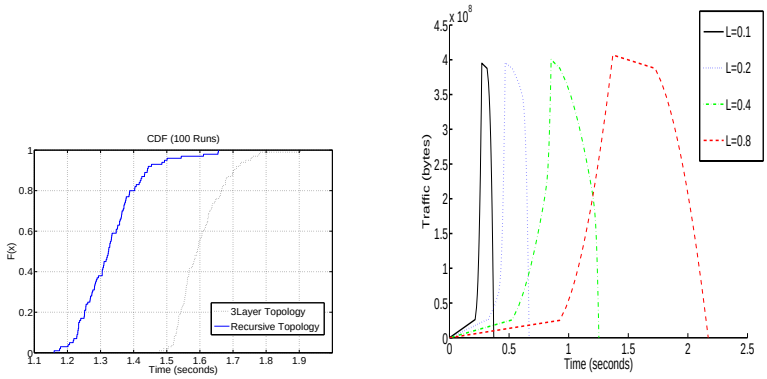
Fig. 7. Minimum source bandwidth for the 3-level topology when $W = 404$ bytes (a) and $W = 1200$ bytes (b)

The uniform latency assumption, however, seems to be too strong. How does it hold water in practice? Our simulation result shows that one can simply set the uniform latency to be the average of real-world latencies and get good propagation time.

We simulated the propagation of the worm in a network with varied latencies. These latencies were generated by the empirical latency distribution in the Skitter data set [8]. We generated 100 sets of latencies, corresponding to 100 network configurations. For each network configuration, we computed the optimal propagation topology based on relation (2), using the average latency as the uniform latency in the model. We then simulated the propagation following this topology, and compared it with the propagation following the 3-level topology on the same network. Due to the extensive computation of the simulation, we set $N = 100,000$ nodes instead of 1 million nodes. We kept $r = 1$ Mbps as before. The average latencies were around 201ms. Figure 8(a) depicts the distributions of the propagation time over 100 configurations for both topologies. It can be seen that the recursive structure computed by (2) using the average latency is still superior to the 3-level topology.

Another advantage of the recursive topology is that it can retain similar time efficiency as the Flash worm starting from a root node with much less bandwidth capacity. Figures 7(a) and (b) show the minimum bandwidth at the root of the 3-level topology (of the Flash worm) 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 7(b).

We also simulated the worm traffic generated during the propagation process. Figure 8(b) 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 400 Mbs, independent of the latencies. This number is



(a) Cumulative distributions (b) Total worm traffic, $N = 1$ Million nodes.

Fig. 8. Cumulative distribution of propagation time of the Recursive topology and the 3-level topology (a) and total worm traffic (b)

significantly smaller than the traffic generated by some actual worms such as Slammer (165 Gbs). This is because the worm we consider is not a random-scanning worm, and thus can eliminate significantly the traffic for scanning. Another note is that, for this amount of total traffic the worm is unlikely to cause any significant instability in the network core, validating our assumption for the network model. Super-fast worm thus is a very practical possibility, even under limited target resources!

5 Discussions and Future Works

The propagation of a malicious code in many ways resembles a broadcast problem [6, 7]. However, there are many differences between these problems and our problem. For instance, [6] did not consider variation of bandwidths as in our problem. In more recent works in P2P and overlay network contexts such as [7, 11], the intermediate nodes also took advantage of the P2P/overlay network structure to broadcast. In our problem, however, only the source can determine the broadcast topology and the broadcast schedule.

As a solution to our problem, propagation topology C 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, the structure is probably far from the best one can hope for. The main reason we chose topology C to analyze is the feasibility of its analysis. The key idea behind a resilient propagation topology is that there must be multiple paths to a target, and the graph should be "expanding" to allow for concurrent propagation. The obvious choice would seem to be some sort of expanders [12], which are graphs with very high connectivity and relatively low diameters, thus reducing propagation time while keeping a high level of resiliency. This line of attack is wide open for further research.

For the general latency case, we have shown that the problem MTMP is **NP**-hard. The obvious open problem is to devise a good approximation algorithm for this prob-

lem. 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 [20] can thwart these infection schedules. Finally, the second problem we formulated – MEMP – has not been addressed at all.

References

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

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

[3] *http://www.icir.org/yoid/*.

[4] I. ARCE AND E. LEVY, *An analysis of the slapper worm*, IEEE Security and Privacy, vol. 1 (2003), pp. 82–87.

[5] S. BANERJEE, B. BHATTACHARJEE, AND C. KOMMAREDDY, *Scalable application layer multicast*, in SIGCOMM 2002, New York, NY, USA, 2002, ACM Press, pp. 205–217.

[6] A. BAR-NOY AND S. GUHA, *Message multicasting in heterogeneous networks*, SIAM J. Comput., vol 30 (2000), pp. 347–358.

[7] E. BROSH AND Y. SHAVITT, *Approximation and heuristic algorithms for minimum delay application-layer multicast trees*, in INFOCOM 2004, vol. 4, Mar. 2004, pp. 2697–2707.

[8] CAIDA, *Skitter datasets*. `http://www.caida.org/tools/measurement/skitter/`.

[9] 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.

[10] M. W. EICHIN AND J. A. A. ROCHLIS, *With microscope and tweezers: An analysis of the internet virus of november 1988*, in Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy, Oakland, Ohio, 1989, pp. 326–343.

[11] S. EL-ANSARY, L. O. ALIMA, P. BRAND, AND S. HARIDI, *Efficient broadcast in structured P2P networks*, in IPTPS'03: Proceedings of the 2003 International workshop on Peer-to-Peer Systems, 2003, pp. 304–314.

[12] S. HOORY, N. LINIAL, AND A. WIGDERSON, *Expander graphs and their applications*, Bull. Amer. Math. Soc. (N.S.), 43 (2006), pp. 439–561 (electronic).

[13] L. LI, D. ALDERSON, W. WILLINGER, AND J. DOYLE, *A first-principles approach to understanding the internet's router-level topology*, in SIGCOMM 2004, New York, NY, USA, 2004, ACM Press, pp. 3–14.

[14] 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, 2003, ACM Press, pp. 24–33.

[15] 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, 2003, ACM Press, pp. 365–378.

[16] B. MCCARTY, *Botnets: big and bigger*, in IEEE Security and Privacy Magazine, vol. 1, 2003, pp. 87–90.

[17] D. MOORE, V. PAXSON, S. SAVAGE, C. SHANNON, S. STANIFORD, AND N. WEAVER, *Inside the slammer worm*, IEEE Security and Privacy, vol. 1 (2003), pp. 33–39.

[18] ———, *The spread of the sapphire/slammer worm*. `http://www.caida.org/publications/papers/2003/sapphire/sapphire.html`, 2003.

[19] 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, 2002, ACM Press, pp. 273–284.

[20] 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.

[21] 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.

[22] A. ODLYZKO, *Data networks are lightly utilized, and will stay that way*, Review of Network Economics, 2 (September 2003), pp. 210–237.

[23] S. RATNASAMY, P. FRANCIS, M. HANDLEY, R. KARP, AND S. SCHENKER, *A scalable content-addressable network*, in SIGCOMM 2001, 2001, pp. 161–172.

[24] C. SHANNON AND D. MOORE, *The spread of the witty worm*, IEEE Security and Privacy Magazine, 2 (2004), pp. 46–50.

[25] N. SPRING, R. MAHAJAN, D. WETHERALL, AND T. ANDERSON, *Measuring isp topologies with rocketfuel*, IEEE/ACM Trans. Netw., 12 (2004), pp. 2–16.

[26] 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, 2004, ACM Press, pp. 33–42.

[27] 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, 2002, USENIX Association, pp. 149–167.

[28] F. WANG AND L. GAO, *On inferring and characterizing internet routing policies*, in IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, New York, NY, USA, 2003, ACM Press, pp. 15–26.

[29] 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.

[30] ———, *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.

[31] C. C. ZOU, W. GONG, D. TOWSLEY, AND L. GAO, *The monitoring and early detection of internet worms*, IEEE/ACM Trans. Netw., 13 (2005), pp. 961–974.

[32] 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, 2005, IEEE Computer Society, pp. 199–206.

A Appendix

A.1 Proof of Theorem 1

We will 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 $k \geq m$. It is **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.

$$r_{v_0} = r_S^{(d)} = R_1 := c, \ \ \forall S \in \mathcal{S}$$
$$r_S^{(u)} = r_e = R_2 := 2nc, \ \ \forall S \in \mathcal{S}, \forall e \in U.$$

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

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

Lemma 7 *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 structure and schedule with total infection time at most $(m + n + 3/2)$. Consequently,* MTMP *is* **NP***-hard.*

Proof. 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 structure $G = (V, E)$ 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 structure $G = (V, E)$ 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 $T_1 + T_2 > n + m + 3/2$.

A.2 Proof of Theorem 2

Consider a node that starts to infect m targets at time 0 following a preemptive schedule. For each target v_i, let T_i, $1 \leq i \leq m$, be the time the source finishes transmission to v_i

in the preemptive schedule. Also, let W_i be the amount of data transmitted to v_i (which includes the base malicious code W and other information). Without loss of generality, suppose $T_1 \leq T_2 \leq \cdots \leq T_m$. The actual time for v_i to be infected would then be $T_i + L$. Denote $f_i(t)$ as the amount of bandwidth reserved for the transmission to v_i at time t. We then have $W_i = \int_0^{T_i} f_i(t)dt$, and $\sum_{j=1}^{m} f_j(t) \leq r$.

Now consider the non-preemptive schedule following the same order $(1, 2, \cdots, m)$, in which the source infects one node at a time using the whole bandwidth capacity. For $1 \leq i \leq m$, let T_i' be the amount of transmission time the source uses to infect v_i. The total amount of time until v_i is infected is $T_i' + L$. To finish the proof, we need to show that $T_i' + L \leq T_i + L$. We have:

$$\begin{aligned} T_i' + L &= \sum_{j=1}^{i} \frac{W_i}{r} + L = \sum_{j=1}^{i} \frac{\int_0^{T_j} f_j(t)dt}{r} + L \\ &= \sum_{j=1}^{i} \frac{\int_{T_{j-1}}^{T_j} (\sum_{k=j-1}^{i} f_k(t))dt}{r} + L \leq \sum_{j=1}^{i} \frac{\int_{T_{j-1}}^{T_j} r dt}{r} + L \\ &= T_1 + (T_2 - T_1) + \cdots + (T_i - T_{i-1}) + L = T_i + L \end{aligned}$$

A.3 Proof of Theorem 3

We show this by induction. When $n = 1$, $T(1) = 0$ since the source is already infected. Suppose (3) holds for all $n \leq k - 1$. For $n = k$, we have

$$\begin{aligned} T(k) &= \min_{1 \leq n_1 \leq \lfloor \frac{k}{2} \rfloor} \left\{ \frac{W - a}{r} + \frac{an_1}{r} + T(k - n_1) \right\} \\ &= \min_{1 \leq n_1 \leq \lfloor \frac{k}{2} \rfloor} \left\{ (1 + \lceil \log(k - n_1) \rceil) \left(\frac{W - a}{r} \right) + (k - 1)\frac{a}{r} \right\} \\ &= (1 + \lceil \log(k - \lfloor k/2 \rfloor) \rceil) \left(\frac{W - a}{r} \right) + (k - 1)\frac{a}{r} \\ &= \lceil \log k \rceil \left(\frac{W - a}{r} \right) + (k - 1)\frac{a}{r}. \end{aligned}$$

This value is achieved at $n_1 = \lfloor n/2 \rfloor$.

A.4 Proof of Theorem 4

Consider an arbitrary infection topology G. For each node v_i ($0 \leq i \leq n - 1$), where v_0 is the source, let Z_i be the random variable indicating if v_i is infected using this topology. Then, clearly Prob$[Z_i = 1] \leq 1 - p$. Thus, by linearity of expectation,

$$\mathrm{E}_N^G(n) = \mathrm{E}\left[\sum_{i=0}^{n-1} Z_i \right] = \sum_{i=0}^{n-1} \mathrm{E}[Z_i] \leq 1 + \sum_{i=1}^{n-1} \mathrm{E}[Z_i] = 1 + (n - 1)(1 - p) \quad (8)$$

Equality holds if and only if Prob$[Z_i = 1] = 1 - p$, which means that there is a direct edge from v_0 to v_i. Otherwise, there is a positive probability that every path from the root to v_i has a node not infectable, implying v_i not infectable.

A.5 Proof of Lemma 5

Recall that infection time is the amount of time starting from when the source sends out its first bit until the last bit of the worm is gone from the network.

Firstly, the time it takes until the last packet from the source completely disappears from the network is

$$Z_1 = x\frac{W + k'a}{r} + L.$$

Secondly, let x_I be the last node at level 2 which is infected. Note that, I is a random variable taking values from 0 to x, where the value of 0 indicates that no node at level 2 is infected. It follows that

$$\text{Prob}[I \leq j] = p^x + \sum_{i=1}^{j}(1-p)p^{x-i} = p^{x-j} \tag{9}$$

$$\text{Prob}[I > j] = 1 - p^{x-j} \tag{10}$$

If $I > 0$, the total amount of time until the last bit from x_I disappears from the network is

$$Z_2 = I\frac{W + k'a}{r} + L + \frac{k'W}{r} + L = I\frac{W + k'a}{r} + \frac{k'W}{r} + 2L$$

If $I = 0$, set $Z_2 = 0$. Depending on the relationship between various parameters $(k, k', L, \ldots)$, the source might still be transmitting when x_I has finished, or vice versa. The total time the worm is on the network is $Z = \max\{Z_1, Z_2\}$, where Z_1 is a constant while Z_2 is a random variable. We wish to compute

$$\text{E}_T^C(n) = \text{E}[Z] = Z_1\,\text{Prob}[Z_1 \geq Z_2] + \text{E}[Z_2 \mid Z_1 < Z_2]\,\text{Prob}[Z_1 < Z_2]. \tag{11}$$

Note that $Z_1 \geq Z_2$ is equivalent to $I \leq x - \left\lceil \frac{k'+Lr/W}{k'a/W+1} \right\rceil$. Assuming n is large, then x is greater than the constant $l = \left\lceil \frac{k'+Lr/W}{k'a/W+1} \right\rceil$. From (9) and (10), we have $\text{Prob}[Z_1 \geq Z_2] = p^l$, and $\text{Prob}[Z_1 < Z_2] = 1 - p^l$. It remains to compute $\text{E}[Z_2|Z_1 < Z_2]$. Since $Z_1 < Z_2$ is equivalent to $I > x - l$, we have

$$\begin{aligned}
E[Z_2|Z_1 < Z_2] &= \sum_{j=x-l+1}^{x} \text{E}[Z_2 \mid I = j]\,\text{Prob}[I = j] \\
&= \sum_{j=x-l+1}^{x} \left(j\frac{W + k'a}{r} + \frac{k'W}{r} + 2L\right)(1-p)p^{x-j} \\
&= \frac{W + k'a}{r}\left[x(1-p^l) + lp^l + \frac{p(p^l-1)}{1-p}\right] + (1-p^l)\left(\frac{k'W}{r} + 2L\right)
\end{aligned}$$

Combined with (11), we get

$$\begin{aligned}
\text{E}_T^C(n) = \frac{W + k'a}{r}\left[x(1-p^l+p^{2l})\right] + \frac{W + k'a}{r}\left[lp^l + \frac{p(p^l-1)}{1-p}\right](1-p^l) \\
+ Lp^l + (1-p^l)^2\left(\frac{k'W}{r} + 2L\right).
\end{aligned} \tag{12}$$