Self-Stopping Worms

Justin Ma, Geoffrey M. Voelker, and Stefan Savage

Collaborative Center for Internet Epidemiology and Defenses
Department of Computer Science and Engineering
University of California, San Diego

ABSTRACT

Modern network worms spread with tremendous speed—potentially covering the planet in mere seconds. However, for most worms, this prodigious pace continues unabated long after the outbreak's incidence has peaked. Indeed, it is this ongoing infection activity that is typically used to identify compromised hosts. In principle, a stealthier worm might eliminate this telltale sign by coordinating its members to halt infection activity after the vulnerable population is subverted. Thus, after a short initial spreading period all infected hosts could become quiescent "sleeper agents." In this paper, we show that such "self-stopping" capabilities are trivial to add to existing worms, and can be efficiently implemented without any explicit coordination or additional network traffic.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; D.4.6 [**Operating Systems**]: Security and Protection—*Invasive software*; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Discrete event*

General Terms

Algorithms, Security

Keywords

worms, malware, distributed protocols, self-stop

1. INTRODUCTION

Existing self-propagating worms are able to blanket millions of Internet hosts in minutes [3] and future worms may achieve the same coverage in seconds [5]. Thus, it is common that such worms are only detected after they have infected much of the vulnerable population. Indeed, when recovering from an epidemic, the standard technique for detecting infected hosts on one's network is to look for their continued scanning activity.

However, the widespread belief that indefinite scanning is an inherent *side-effect* of random scanning strategies is a fragile assumption at best—today's worms simply aren't designed to stop. In this paper we explore distributed algorithms that accurately estimate and moderate the growth of worm epidemics. Thus, a worm could spread quickly for a short period (perhaps only seconds), until a large fraction of its vulnerable population is compromised, and then globally stop itself. Such behavior could make outbreaks harder to detect and, more importantly, could make it far harder to identify the presence of infected hosts after the fact. The challenge of this problem lies in coordinating the scanning dynamics of an unknown number of independent worm instances. Some researchers have proposed alternative worm designs to provide this coordination explicitly via an overlay control network but we argue that this sophistication is unnecessary [1]. We demonstrate that it is easy to modify *existing* oblivious worm designs to autonomously monitor global progress and then change behavior when a particular fraction of the vulnerable population has been subverted. Moreover, we show that this can be done with trivial overhead and no additional network traffic.

In the remainder of this paper we describe our approach in more detail, explore alternative designs and present our initial experiences. Section 2 discusses our assumptions and evaluation criteria, followed by a description of various self-stopping algorithms in Section 3. We evaluate these algorithms in Section 4, and we conclude in Section 5.

2. ASSUMPTIONS AND MODEL

For the purposes of this paper, the goal of a *self-stopping* worm is to stop propagating when the total number of infected hosts has reached a *target population*. This target may in turn be specified as an absolute limit—e.g., 10,000 vulnerable hosts—or as a relative limit—e.g., 95% of the vulnerable population. In addition, we assume that worms propagate by "scanning" and infecting target hosts. Thus, an infected host will initiate γ infection attempts (scans) in a timestep—we refer to γ as the *per-host scanning rate*. We consider two target selection strategies: *uniform random scanning* and *permutation scanning*. In the first case, an infected host chooses each target randomly out of an address space of size A. N out of the A hosts are *vulnerable* to infection, and we assume this quantity is static. Similarly, in the permutation scanning regime, infected hosts select targets from a universally predetermined, circular sequence of addresses—i.e., the *permutation space*. An individual host stores a *seed* value, which is an index into the permutation space representing the address it will scan. After scanning, the host increments its seed to point to the next address in the permutation, and repeats [6].

The only dynamic quantities of interest in our model are $I(t)$, the

WORM'05, November 11, 2005, Fairfax, Virginia, USA.

Param	Definition
t	Time (in timesteps)
A	Size of address space
N	Total number of vulnerable hosts
γ	Per-host scanning rate (scans per timestep)
$I(t)$	Number of infected hosts at time t
$a(t)$	Number of infected hosts that are actively infecting others

Table 1: Notation for epidemic parameters and quantities.

number of infected hosts at time t, and $a(t)$, the number of hosts that are infected and active at time t. For all our evaluations, we assume that $I(0) = 1$. Table 1 summarizes the notation we shall use throughout the paper.

We also have chosen to explore the most restrictive part of the design space, in which the only form of communication allowed is that opportunistically piggybacked upon existing infection attempts. Thus, we assume that worms do not create any long-lived or structured communication channels amongst themselves. We have chosen these limitations because they require no change to existing worm designs and moreover demonstrate the simplicity of implementing effective self-stopping strategies. We will show that even with these handicaps, purely oblivious distributed algorithms are competitive with a perfect oracle in common scenarios.

We propose several criteria for evaluating the effectiveness of self-stopping strategies. The first is an algorithm's *accuracy* with respect to the target population goal. More formally, the difference between the resulting number of infected hosts and the number specified in the population goal should be small for an accurate strategy. This criterion is important because we should consider a worm that aims to infect 100% of the population but ends up infecting 50% to be ineffective. However, one must discuss accuracy *with respect* to a population goal, because for some strategies a relative limit can be easier to achieve than an absolute limit, and vice versa.

The second measure of effectiveness is a small *network footprint*—in particular a worm's *speed*, *duration*, and *aggregate scan traffic*. Speed refers to the amount of time a worm requires to infect a target portion of the population. Duration is the amount of time elapsed from the beginning of an epidemic to the time all infected hosts have deactivated. Aggregate scan rate is the amount of traffic generated by all active, infected nodes in a time interval, and the scan total measures such traffic over the entire epidemic. Presumably, worm authors would prefer to minimize network footprint and thus the opportunity for detection.

The third criterion we use to characterize effectiveness is the *ease of implementation*, which accounts for inter-node communication structure, the amount of state kept at each node, and feedback level between nodes. As we discussed earlier, strategies that try to construct overlays among infected nodes are more complex than those that simply piggy-back information with infection probes. The amount of state an infected node keeps can range from stateless, e.g., simply relying on fortuitous contact to initiate a deactivation sequence, to storing scalars (infection counts, scan counts, etc.), to detailed records like keeping a list of every infected host it has ever contacted. The feedback level refers to the amount of detail in communications between nodes. A node could receive no feedback at all on an infection attempt, like many existing UDP worms—we say that worms that operate in this regime are *blind*. Alternatively, a node could take advantage of knowledge that an infection attempt was a new hit (reaching a previously uninfected vulnerable host), a redundant hit (reaching a previously infected vulnerable host), or a miss (reaching a non-vulnerable host). Furthermore, a node could seed its descendants with epoch counters and generation numbers, or exchange data from scalars to more complex data structures. There are many possibilities, but we restrict our attention to a handful of them.

In our evaluations we focus on an accuracy requirement where worms must infect at least 85% of vulnerable hosts before they stop, and evaluate the impact of different strategies on network footprint (in particular, speed and duration). Ease of implementation is an inherently more subjective criterion because some worm authors may be willing to implement a simple algorithm that requires a large amount of memory (high state cost), and others may be willing to invest time in implementing a more sophisticated algorithm that uses less state. Nonetheless, we discuss the self-stopping heuristics below in order of increasing complexity.

3. SELF-STOPPING HEURISTICS

This section presents the set of self-stopping worm heuristics that we will evaluate in Section 4 in order of increasing power (where power refers to the type of knowledge available to a node). We introduce these heuristics in the context of uniform random scanning, followed by descriptions of permutation scanning analogs.

As a baseline, we first describe the default, greedy mode of operation found in many worms:

Greedy: An infected node infects as many hosts as possible without stopping.

We will only evaluate the speed of this worm. Since there is no stopping strategy for Greedy, the duration of the epidemic will be infinite.

3.1 Non-Exchange, Non-Estimating Strategies

The first two self-stopping methods we explore are from the distributed systems literature and were introduced by Demers *et al.* for moderating the overhead of epidemic flooding for distributed replica updates [2]. The justification for using these techniques arise from a model of epidemic behavior that accounts for removed nodes: $\frac{dI}{dt} = \frac{\gamma}{A}(N - I)a$, and $\frac{da}{dt} = \frac{\gamma}{A}(N - I)a - \frac{1}{k}a$. The $-\frac{1}{k}a$ factor means that $\frac{1}{k}$ of active hosts simply deactivate at the end of a timestep. This yields the following strategy that does not rely on feedback of any kind:

Blind-k: An infected node deactivates with probability $\frac{1}{k}$ at the end of each timestep.

In combining the two aforementioned equations, we can express the number of active hosts in terms of the number of hosts that were ever infected in the epidemic: $a(I) = I + \frac{1}{k}\frac{A}{\gamma}\log(1 - \frac{I}{N})$. We can predict the total number of vulnerable hosts infected at the end of the epidemic by solving for $a(I) = 0$ (the epidemic is finished when the number of active hosts becomes zero for a non-trivial value of I). For example, when $A = 2^{20}$, $N = 512$, $\gamma = 250$ and $k = 32$ (these are parameters we use in the evaluations for scanning a 20-bit address space), 97.8% of the population will be infected with a Blind-32 strategy. When $A = 2^{32}$, $N = 2^{17}$, $\gamma = 4,000$ and $k = 32$ (parameters in the evaluations for scanning a 32-bit address space), 97.8% of the population will be infected with a Blind-32 strategy. Note that parameterizing k for the Blind-k strategy to reach a target population I requires *a priori* knowledge of N, A, and γ, assumptions which may be very difficult for a worm initiator to make.

The next strategy from [2] requires the worm to detect redundant hits on vulnerable nodes, and the equation for $\frac{da}{dt}$ changes to

$\frac{da}{dt} = \frac{\gamma}{A}(N - I)a - \frac{1}{k}\frac{\gamma I}{A}a$. The $-\frac{1}{k}(\frac{\gamma I}{A})a$ factor means that $\frac{1}{k}$ of the active nodes that achieved a redundant hit—where $\frac{\gamma I}{A}$ is the expected number of redundant hits generated per host—in the current timestep have deactivated. Hence, we have the following strategy:

Stop-k: Upon achieving a redundant hit, an infecting host will deactivate with probability $\frac{1}{k}$.

We can combine the above differential equations to express the number of active hosts in terms of the number of infected hosts: $a(I) = \frac{k+1}{k}I + \frac{N}{k}\log(1 - \frac{I}{N})$. Using this equation, we can predict the total number of vulnerable hosts infected by the end of the epidemic—regardless of the worm's scanning rate or the size of the address space—by solving for $a(I) = 0$. For example, when $k = 1$, 2, and 3 and $N = 2^{17}$, the infected population will reach 80%, 94%, and 98% of vulnerables respectively. Note that this strategy requires nodes to know the outcome of their infection attempts, i.e., whether the attempt resulted in a new/redundant hit or a miss. We call this type of feedback *infection-status feedback*.

If worms propagate in a structured manner so that a diagram of parent-child relationships resembles a simple data structure (in the following case, a k-ary tree), then estimating the number of hosts that will be infected at the end of an epidemic is straightforward assuming the absence of node failure. This is the motivation behind the next strategy, which forces nodes to infect others in a tree-like manner but does not use exchange or estimation. We describe this precisely as follows:

Tree: Each infected node has a generation number i, with the generation of the original infected host set to 0. Infected nodes with generation number i less than a constant m must infect k previously-uninfected hosts (and assign the new hosts generation numbers of $i + 1$) before deactivating. Nodes with a generation number equal to m may not scan, and must immediately deactivate. With this technique, $I = \frac{k^{m+1}-1}{k-1}$ at the end of the epidemic.

3.2 Estimating Epidemic Parameters

The goal of infecting a target number or fraction of the vulnerable population motivates the design of strategies that attempt to estimate the vulnerable population (N) and the infected population at a given point in time ($I(t)$). Knowing $I(t)$ alone is sufficient to reach an absolute limit, but knowledge of both N and $I(t)$ is desirable for hosts wishing to reach a relative limit.

For situations where it is difficult to empirically estimate both N and $I(t)$, it is often convenient to estimate one using empirical measurements or exchanging information with other nodes, and to calculate the other using a formula based on the logistic equation for modeling infectious epidemics. For our evaluations, an infected node uses the following equation to estimate $I(t)$ given t, an estimate for N, and the per-host scanning rate γ: $I(t) = N\frac{e^{\beta(t-t_0)}}{1+e^{\beta(t-t_0)}}$ where $\beta = \frac{\gamma N}{A}$, and $t_0 = \frac{\ln(N-1)}{\beta}$. Additionally, an infected node can estimate N given $I(t)$ and γ: $N = I(t) + \frac{A(I(t+1)-I(t))}{\gamma I(t)}$.

This equation comes from the discrete version of the differential equation $\frac{dI}{dt} = \frac{\gamma}{A}I(t)(N-I(t))$ for modeling epidemics. Note that these estimators assume that nodes have access to a global clock. In practice, a worm could approximate a global clock by seeding its descendants with the birth-time of the first infected node or the number of epochs that have elapsed so far (assuming nodes agree on the length of an epoch).

In summary, we can use the traditional analytic models of infectious epidemics to derive N from an estimate of $I(t)$ and vice versa. The remainder of this section discusses self-stopping methods that estimate N through empirical sampling and exchanging information, and presents other heuristics that do not rely on such estimates. However, we do not present techniques for estimating $I(t)$ empirically. A candidate approach would be to estimate the infected population at time t using the number of redundant hits that a node achieves in a timestep. The primary problem with this approach is that, because $I(t)$ is a dynamic quantity, the expected number of redundant hits gathered in a timestep will change—this prevents a node from using samples gathered in previous timesteps to augment estimates in later timesteps.

3.3 Empirically Estimating N

The scanning that an active host undertakes to search for vulnerable hosts (which form a static fraction of the address space) is a random process that lends itself naturally to two approaches for correlating the number of successful hits with the vulnerable population: *sampling with replacement* and *sampling without replacement*.

The first heuristic we present uses *sampling with replacement* to estimate N. If a host records the number of vulnerable hosts it has ever attempted to infect (H) as well as the total number of scans it has produced (S), then we can model the number of hits achieved by a worm during the scanning process as a binomial distribution where there are S trials and the probability of success for each trial is $\frac{N}{A}$. Hence, $E[H] = S\frac{N}{A}$. Thus we have the following heuristic:

Sum-Count: An infected host keeps two counters, one for the number of vulnerable hosts it has contacted H, and one for the number of scans it has produced S. The host can calculate an estimate N_{est} of the vulnerable population using $N_{est} = \frac{HA}{S}$.

The next heuristic we present uses *sampling without replacement* to estimate N. If a host can record the number of distinct vulnerable hosts that it has ever infected (h) as well as the number of distinct hosts that it has scanned (s), then we can model the number of hits achieved by a worm during the scanning process with a hypergeometric distribution where the number of possible successes is N, the number of failures is $(A - N)$, and there are s trials. Hence, $E[h] = \frac{sN}{N+(A-N)} = s\frac{N}{A}$. One method for tracking the distinct hosts infected or scanned by a host is to use a bitmap with A bits, where each bit position represents a host in the address space and a bit is set for each infected/scanned host. This gives rise to the following self-stopping worm heuristic:

Bitmap: An infected host uses two bitmaps, each of size A-bits. It uses the first bitmap Bit_v to record the vulnerable hosts it has attempted to infect (new and redundant hits on vulnerable hosts are included in this bitmap), and it uses the second bitmap Bit_s to record the hosts it has scanned, regardless of whether the host receiving its scan was vulnerable. The infected host calculates an estimate N_{est} of the vulnerable population where $N_{est} = \frac{bitsset(Bit_v) \cdot A}{bitsset(Bit_s)}$.

A clear drawback to the naive Bitmap method is the large amount of state required by each infected host to store the bitmap. While there are several alternative data structures for approximating this information compactly we do not record them here. We will show that a variant of Sum-Count can almost match the Bitmap method without any need for additional sophistication.

Although the estimators we use under sampling with and without replacement are the same, the error analyses differ slightly. $Var[N_{est}] = Var[\frac{H}{S}] = \frac{1}{S}\frac{N}{A}(1 - \frac{N}{A})$ for sampling with replacement, and $Var[N_{est}] = Var[\frac{h}{s}] = \frac{1}{s}\frac{N}{A}(1 - \frac{N}{A})(\frac{A-s}{A-1})$ for sampling without replacement. In both cases, the estimate errors improve as the number scans increase over time. Note that these strategies require nodes to have infection-status feedback, but do not require nodes to exchange extra information.

3.4 Augmenting Estimates With Exchange

The Sum-Count and Bitmap techniques for locally estimating N only require the node to distinguish between vulnerable and non-vulnerable hosts for every infection attempt. However, the error analyses for those techniques suggest that we can make estimates converge more quickly if we allow nodes to combine each other's estimates. We propose the following variations:

Sum-Count-X: This technique operates like Sum-Count, except that when node A (with vulnerable hit count H_A and scan count S_A) contacts node B (with vulnerable hit count H_B and scan count S_B), their resulting sum counts become $H_A + H_B$ for hits and $S_A + S_B$ for scans.

Bitmap-X: This technique operates like Bitmap, except that when node A (with bitmaps $Bits_{v,A}$ for vulnerable and $Bits_{s,A}$ for scanned) contacts node B (with bitmaps $Bits_{v,B}$ for vulnerable and $Bits_{s,B}$ for scanned), then the new bitmaps on A and B for tracking infected/scanned nodes are $Bits_{v,A} \cup Bits_{v,B}$ for vulnerable hosts and $Bits_{s,A} \cup Bits_{s,B}$ for scanned hosts.

3.5 Permutation Scanning Variants

Staniford *et al.* designed permutation scanning as an alternative to random scanning for victim selection in Internet worms [6]. The primary motivation behind permutation scanning is to increase the speed of an epidemic by avoiding redundant address selection among probing hosts. However, the technique also offers the potential to reduce duration and scan traffic when used in conjunction with divide-and-conquer methods.

We evaluate four permutation scanning variants of self-stopping heuristics in this paper. The first is a variation of the Greedy strategy, the second is Stop-k, the third is a Sum-Count-X variation, and the last is a unique optimization where a node and its descendants partition the address space among themselves:

Greedy Permutation: If the host achieves a redundant hit, it will randomly choose a new seed and continue. No hosts stop. Reseeding is motivated by the observation that another host is already scanning forward in the current permutation space—any continued scanning with the same seed would be a duplicate effort.

Stop-k Permutation: Same as Stop-k.

Sum-Count-X Permutation: Same as Sum-Count-X, except with the reseed-upon-redundant-hit policy used by Greedy Permutation.

Partitioned Permutation: As suggested in [6], each infected host is given responsibility for scanning a disjoint interval of the permutation space. Upon achieving a new hit, a host assigns half the unscanned portion of its current interval to the newly infected descendant, and continues scanning the other half. A host stops when it has finished scanning its interval (note that achieving a redundant hit implies reaching the end of an interval, although the converse might not apply).

The analysis we applied earlier to Sum-Count-X is not a perfect fit for Sum-Count-X Permutation because permutation scanning is closer to sampling without replacement, assuming that reseeding does not overlap with previously scanned intervals. Nonetheless, we will demonstrate that dynamic estimation has a positive impact on the propagation dynamics of a permutation scanning worm as well. Also, there are more sophisticated variations of partitioned permutation scanning, such as switching to regular permutation scanning when the interval size is lower than a threshold value [6]. However, we do not explore those further.

4. EVALUATION

In this section we use simulation to evaluate the ability of the self-stopping worm heuristics described in Section 3 to balance the speed and thoroughness of infecting vulnerable host populations with the goal of achieving effective self-stopping behavior. We first describe our simulation methodology and parameterizations of the self-stopping heuristics. Then we evaluate the infection speed and stopping duration of the basic heuristics that require initial parameterization and tuning, but otherwise require no estimation or information exchange. Next we evaluate the heuristics that dynamically estimate the epidemic parameters during the infection, and demonstrate the value of having hosts exchange estimates during the epidemic. Afterward, we examine the effectiveness of permutation scanning strategies. Finally, we examine the sensitivity of the heuristics to the vulnerability density of the host population, as well as the amount of scan traffic produced by each heuristic.

4.1 Methodology

We simulate the behavior of a simple random scanning worm spreading in an address space of size A among a set of vulnerable hosts N. During the simulation, a vulnerable host undergoes the following progression of states: *uninfected*, *pending*, *infected*, *inactive*. Initially in an epidemic, one host is infected and all vulnerable hosts are uninfected. We simulate the spread of a worm by dividing time into rounds. For each round, each infected host in turn scans the address space by picking γ addresses uniformly at random from the address space. We use the term *hit* to describe the selection of an address belonging to a vulnerable host. If the scanning host hits an uninfected host, then the simulator marks the scanned host as pending and records the hit as *new*. If the scanning host hits a pending, infected, or inactive host, the state of the scanned host does not change and the hit is *redundant*. During a hit (whether new or redundant), the scanning and scanned hosts may exchange information (e.g., bitmaps, sum-counts, generation seeding). After all infected hosts have scanned in a round, the infected and pending hosts evaluate a stopping heuristic to determine whether to transition to the inactive state. At the end of the round, remaining pending hosts become infected. To focus solely on the behavior of a self-stopping heuristic, we do not simulate network effects: any host can contact any other host, and we do not model network delays, bandwidth, or contention. The ability of a self-stopping heuristic to infect the vulnerable population quickly and thoroughly before stopping is limited only by the heuristic itself.

To optimize the simulation, we preallocate h hits to each infected host and allow the infected host to choose h addresses uniformly at random from the N vulnerable hosts. The simulation chooses h for each host per round according to a binomial distribution with γ samples and a probability of success of $\frac{N}{A}$.

To simulate host selection for permutation scanning, we generate a global permutation of N entries before each run. Each entry is a pair (seq, $host$) where seq is a unique sequence number in a permutation space of size A, and $host$ is exactly one of the N vulnerable hosts. Entries in the permutation are ordered by increasing sequence number (accounting for arithmetic modulo A). For each round, an infected host with seed σ grabs entries (seq_i, $host_i$) from

Strategy	Stopping Condition	Feedback
Blind-k	Stop with probability $\frac{1}{k}$.	None
Stop-k	Stop with probability $\frac{1}{k}$ after redundant hit.	Infection-status
Tree	Stop after infecting k new hits on vulnerable.	Infection-status, generation seeding
Sum-Count	Estimate N empirically using sampling with replacement. Stop when estimated I reaches threshold.	Infection-status
Bitmap	Estimate N empirically using sampling without replacement. Stop when estimated I reaches threshold.	Infection-status
Sum-Count-X	Like Sum-Count, but incorporate sums from other nodes.	Trading scalars
Bitmap-X	Like Bitmap, but incorporate bitmaps from other nodes via bitwise union.	Trading bitmaps

Table 2: Summary of self-stopping heuristics.

	Speed			Duration		
Strategy	10th	50th	90th	10th	50th	90th
Greedy	105	117	135	∞	∞	∞
Blind-32	116	123	132	223	239	259
Stop-3	113	127	139	410	446	515
Tree ($k = 50, m = 3$**)**	1514	1637	1712	11744	13699	15961
Know N **and** I	112	119	139	155	161	181
Know I	111	123	142	153	163	184
Know N	114	121	147	154	154	154
Sum-Count	148	159	189	426	448	482
Sum-Count-X	103	119	136	156	174	199

Table 3: Speed (timesteps to infect 85% of vulnerables) and duration (timesteps for all infected hosts to stop) by percentile of worm epidemics using various self-stopping strategies. Address space size $A = 2^{32}$, number of vulnerables $N = 2^{17}$, and per-host scan rate $\gamma = 4,000$ per timestep.

the permutation so that $\sigma \leq seq_i < \sigma + \gamma$, and proceeds to infect each corresponding $host_i$. The $host_i$'s represent vulnerable hosts that the infected node would have contacted after scanning γ elements along the permutation sequence space.

Except where explicitly noted, all of our experiments simulate an address space $A = 2^{32}$ with $N = 2^{17}$ vulnerable hosts, roughly the number of hosts infected by the Slammer worm [3] (we examine the sensitivity of the size of N in Section 4.5 below). Infected hosts scan at a rate of $\gamma = 4,000$ scans per timestep. With a simulated timestep corresponding to one second, this rate corresponds to the observed average scanning rate of Slammer, with the simplifying assumption that all hosts scan at the same rate without any network bottlenecks. We experimented with other scan rates and did not find that the relative performance of the heuristics was sensitive to the parameter.

The heuristics Blind-k, Stop-k, and Tree require parameterization and tuning to be effective for particular choices of A and N for an epidemic (the dynamic estimation heuristics do not have this limitation). We choose parameters such that these heuristics will infect most of the vulnerable population while achieving good self-stopping performance for that heuristic. Because of the random nature of the worm epidemic and self-stopping heuristics, it is not practical to parameterize many of the heuristics to completely infect 100% of the vulnerables.

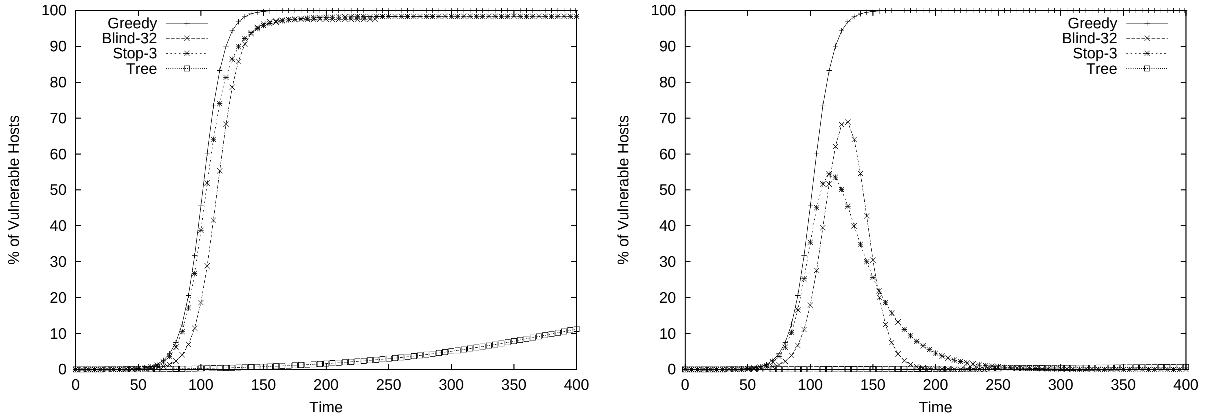
For Blind, we set its stopping parameter $k = 32$. The analysis in Section 3 indicates that, for a 32-bit address space with 2^{17} vulnerable hosts making 4,000 scans/timestep, Blind-32 will infect 98% of the vulnerable hosts on average. For Stop, we set $k = 3$ which, as per our analysis, will infect 98% of the vulnerable hosts, regardless of address space size or scanning rate. For Tree, we set a branching factor of $k = 50$, a maximum generation number of $m = 3$, and a goal of infecting 97% of the vulnerable hosts. We experimented with other parameters; for instance, $k = 2$ and $m = 16$ will infect all but one of 2^{17} vulnerable hosts. Simulations using such aggressive infection parameters, however, result in extremely long worm epidemics due to the time required to randomly scan the few remaining uninfected vulnerable hosts at the end of an epidemic.

4.2 Basic Heuristics

We start by evaluating the effectiveness of the basic heuristics Blind-32, Stop-3, and Tree, and compare them against the Greedy worm. These heuristics are straightforward to implement and depend on only local information to stop. The top part of Table 3 shows the *speed* and *duration* of these heuristics for simulated epidemics. We define speed as the number of timesteps required for a worm to infect 85% of vulnerables; at this stage of the epidemic, most vulnerables are infected and the worm is well beyond the exponential growth phase. We define duration as the number of timesteps required for all hosts infected by a worm to stop scanning relative to the start of the epidemic.

For each heuristic, we simulate 20 worm epidemics and report the 10th, 50th, and 90th percentiles of each metric. Because these worms perform random scans, their speed and duration will vary across simulation runs depending upon the random chance of early infections. "Lucky" worms infect vulnerable hosts sooner and therefore propagate faster; "unlucky" worms take more random trials to infect vulnerable hosts, and propagate slower. The percentiles show the sensitivity of each heuristic to the effect of randomness. In general, the sensitivity is not substantial: both the 10th and 90th percentiles are within roughly 10% of the median.

As expected, Greedy is the fastest strategy since it does not slow down as the epidemic completes the infection. With a simulated timestep corresponding to a second, the median Greedy speed corresponds to a 3-minute epidemic (the real Slammer worm infected



(a) Infected

(b) Active

Figure 1: Infected and active hosts over time for the basic heuristics Greedy, Blind-32, Stop-3, and Tree.

most vulnerable hosts in similar time [3]). Blind-32 and Stop-3 are fast as well, with median speeds marginally slower than Greedy. Tree, however, is an order of magnitude slower than the other heuristics; we discuss this effect in more detail below.

In terms of duration, the heuristics have similar relative performance but with more dramatic variation. Blind-32 has the shortest duration. In comparison, the median duration for Stop-3 is 1.9 times longer than the median for Blind-32, and the duration of Tree is almost 2 orders of magnitude longer. Greedy has an infinite duration since it does not try to stop.

To illustrate the heuristics in more detail, Figures 1a and 1b show the behavior of the heuristics as a function of time. Corresponding to the goal of infecting as many hosts of the vulnerable population as quickly as possible, Figure 1a shows the percentage of vulnerable hosts that are *infected* over time for each of the heuristics. Corresponding to the goal of stopping as quickly as possible once the worm has infected the vulnerable population, Figure 1b shows the percentage of vulnerable hosts that are *active* over time for each of the heuristics. As a representative simulation run for a given heuristic, we use the run that results in the median duration time from Table 3.

The Greedy worm represents the ideal infection performance for a worm since it has the fastest infection speed. A "perfect" self-stopping worm would spread like Greedy until it infects all vulnerable hosts, and then all active infected hosts would stop simultaneously. The goal of the heuristics is to approximate Greedy infection speed while balancing the additional goal of stopping quickly. Figure 1a shows how Blind-32 and Stop-3 spread nearly as quickly as Greedy. They infect almost all of the vulnerable hosts, but not 100% due to their stopping behavior. Stop-3 starts by spreading more aggressively than Blind-32, but hosts begin to deactivate sooner by the middle of the epidemic. As a result, both heuristics take about the same amount of time, for instance, to infect 85% of vulnerable hosts.

Figure 1b shows the dynamic stopping behavior of the heuristics. With Stop-3 the number of simultaneously active hosts peaks at 55% of the vulnerable hosts, but then active hosts quickly begin to stop themselves as they contact other infected hosts. At the same time, it becomes increasingly unlikely that an active host will scan an uninfected vulnerable host, slowing the worm down even more. Eventually, the time it takes for an active host to randomly scan a remaining uninfected vulnerable host is longer than the time to contact multiple already-infected hosts. As a result, all active hosts eventually decide to stop and the worm halts after 446 timesteps.

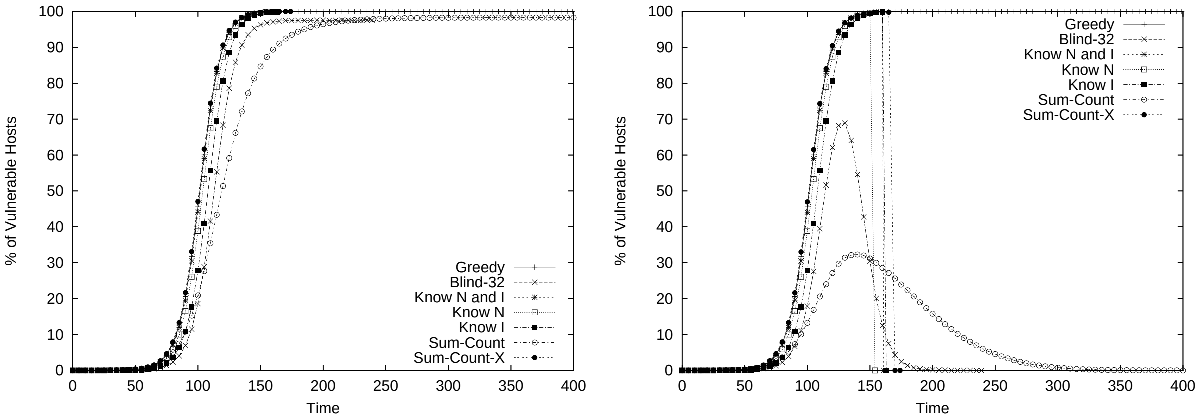
With Blind-32 the active hosts peak higher at 70% of vulnerable hosts, and they stop more aggressively than with Stop-3. Although Blind-32 spreads less aggressively than Stop-3 (infected hosts randomly stop even in the early part of an epidemic), with a higher peak of active hosts it is able to infect as many hosts as Stop-3. Since Blind-32 depends only on time to stop, rather than randomly contacting already-infected hosts, a Blind-32 worm stops substantially sooner than a Stop-3 worm (239 vs. 446 timesteps).

Figures 1a and 1b illustrate the limitations of the basic Tree heuristic. A Tree worm spreads much slower than the other heuristics because infected hosts are constrained in the number of vulnerable hosts they infect. With other heuristics a host infected early will try to infect vulnerable hosts throughout the epidemic, whereas with Tree an infected host will stop when it reaches its infection quota. As a result, basic Tree epidemics take substantially longer to propagate. More complex optimizations could make Tree more aggressive, such as widening the fan-out, assigning redundant targets, etc. But since our goal is to avoid complexity, and simple heuristics already perform comparatively well, we did not investigate further refinements of Tree.

4.3 Dynamic Estimation

The basic heuristics above enable a worm to spread quickly, but they vary widely in their self-stopping ability. More critically, they also require the worm initiator to correctly estimate the size of the vulnerable population ahead of time. Since this information is rarely known, we next evaluate heuristics that dynamically estimate the epidemic parameters during the infection and use these estimates to decide when to stop. The crucial advantage of this approach is that it does not require parameterization or tuning.

We evaluate the dynamic estimators Sum-Count and Sum-Count-X (we discuss the Bitmap heuristic below) and compare them with Greedy, Blind-32 (the best basic heuristic), and three idealized heuristics Know-NI, Know-N, and Know-I. Recall that Sum-Count estimates N by tracking the relative number of vulnerable hosts it has scanned, and derives I based upon its estimate of N. Sum-Count-X further refines Sum-Count by having hosts exchange their estimates when they contact each other. This exchange improves local esti-



(a) Infected

(b) Active

Figure 2: Infected and active hosts over time for the dynamic estimator heuristics.

mates by disseminating global observations. The idealized heuristics have perfect instantaneous knowledge of the number of vulnerable hosts N and/or the number of infected hosts I at all times, and represent an upper bound on estimation accuracy. By comparing against these ideals, we can determine the effectiveness of an estimator and the necessity of estimating accurately for a capable self-stopping worm.

The second half of Table 3 lists the speed and duration of these heuristics. Starting with the ideals, Know-N and Know-I estimate the number of vulnerable hosts or infected hosts in terms of perfect knowledge of the other using the equations in Section 3. Know-NI has perfect knowledge of both quantities and requires no estimation: it knows precisely when it has infected all vulnerable hosts and can stop immediately.

Not surprisingly, the ideal estimators are fast and stop quickly. They have slightly different stopping behaviors, however, and these differences provide additional insights into estimation and subtleties of our metrics. The speed results indicate that estimating N using perfect knowledge of I (Know-I) is slightly slower than Know-N, but this difference is simply an artifact of the random infection process. On the other hand, Know-N stops faster with no variance compared to Know-I; Know-N calculates its I estimate from a static quantity N, whereas Know-I derives estimates based on a quantity whose growth rate varies from epidemic to epidemic.

Note that, with Know-N and Know-I, having perfect knowledge of one of the quantities will result in different behavior than Know-NI because the heuristics still have to estimate the other quantity from the known one. Since host selection is a random process, and the equations in Section 3 model an *expected* propagation, the estimated quantities still introduce error relative to perfect knowledge of both quantities. For example, Know-I knows the true number of infected hosts I, but each infected host still estimates N based upon the analytic model. Depending on the random infection behavior of a worm epidemic, Know-I can have hosts stop before or after the time when all vulnerable hosts have been infected.

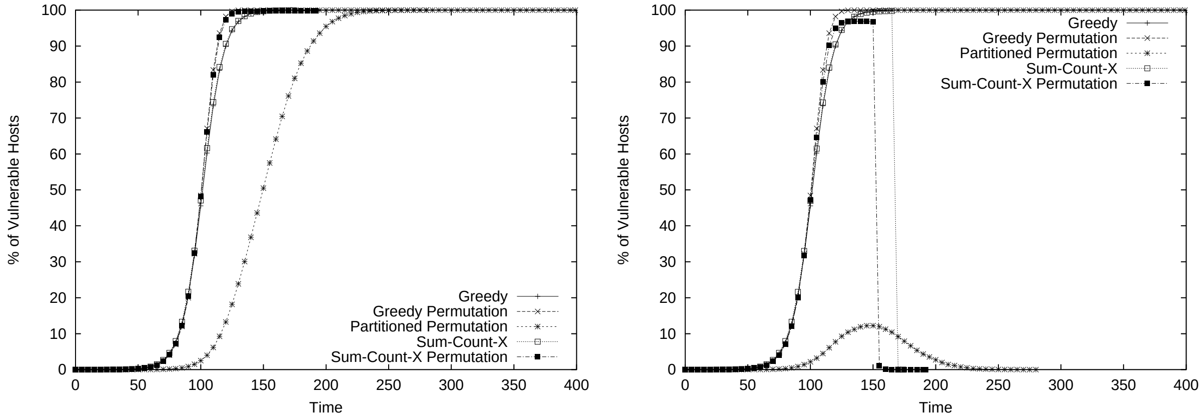
Comparing the durations of Know-NI, Know-N, and Know-I illustrates the effects of estimation even with partial perfect knowledge. For example, Table 3 shows that Know-NI takes longer to self-stop than Know-N. The reason is that Know-NI will not stop until 100% of vulnerable hosts have been infected, whereas Know-N stops based upon its estimate of the number of infected hosts. As a result, Know-N can decide to stop when less than 100% of vulnerable hosts have been infected. So Know-N may stop sooner, but it will not infect quite as many hosts as Know-NI.

Moving on to the practical dynamic estimators, Table 3 shows that Sum-Count is not much of an improvement over the basic estimators. It is slower to infect 85% of the vulnerable population than Blind-32 and Stop-3, and takes as long as Stop-3 for all infected hosts to self-stop. In other words, worms using purely local estimation spread slower than the basic heuristics and do not improve on their ability to stop, but local estimation does have the benefit of not requiring parameterization.

The Sum-Count-X heuristic highlights the value of exchanging information for self-stopping worms. With Sum-Count-X, each host exchanges its counts of vulnerable and scanned hosts with other infected hosts on each contact. As a result, it infects faster than all previous heuristics and stops 1.3 times sooner than Blind-32 (using the median results). It also spreads as fast as the idealized estimators, and stops nearly as quickly: it takes only 8% longer to stop than Know-NI, which has perfect knowledge of both N and I. Sum-Count-X demonstrates that self-stopping worms are practical to implement using a simple distributed algorithm, and that self-stopping, random scanning worms can realize nearly ideal spreading and stopping performance.

Figures 2a and 2b show the dynamic behavior of the estimation heuristics. Figure 2a shows the percentage of vulnerable hosts that a worm has infected over time, and Figure 2b shows the percentage of vulnerable hosts that are active over time. We show results for Know-NI, Know-N, Know-I, Sum-Count, and Sum-Count-X, and we include the results for Greedy and Blind-32 from before for comparison. For all heuristics, we use the simulation run that results in the median duration time reported in Table 3. Figure 2a shows that the idealized estimators indeed produce fast-spreading worms: in terms of infected hosts, they spread as quickly as Greedy. Further, Figure 2b shows that they are able to stop nearly instantaneously once they estimate that all vulnerable hosts have been infected (note that Know-N stops sooner than Know-NI as described above).

The figures also show the effect of a purely local estimator compared with one that exchanges information and incorporates global observations. Sum-Count, the local estimator, spreads even slower than Blind-32, which performs no estimation (although Blind-32



(a) Infected

(b) Active

Figure 3: Infected and active hosts over time for Partitioned Permutation as well as permutation and random scanning versions of Greedy and Sum-Count-X.

	Speed			Duration		
Strategy	10th	50th	90th	10th	50th	90th
Know-NI	58	71	81	91	101	115
Sum-Count	79	100	143	202	231	268
Bitmap	83	100	126	206	237	331
Sum-Count-X	62	73	83	91	109	133
Bitmap-X	60	71	83	101	103	120

Table 4: Speed and duration by percentile of worm epidemics using select self-stopping strategies under a 20-bit address space with 512 vulnerables and a per-host scanning rate of 250 per timestep.

does require tuning, as described in Section 4.1 above). In contrast Sum-Count-X, which exchanges information, is nearly indistinguishable from the idealized estimators.

Finally, recall that Section 3 described a second dynamic estimator, Bitmap-X, that also exchanges estimation information when hosts contact each other during the epidemic. Whereas Sum-Count-X keeps track of counts of vulnerable and scanned hosts, Bitmap-X keeps track of precisely which hosts in the address space are vulnerable and which ones have been scanned and infected. Because it maintains and exchanges more precise information than Sum-Count-X, Bitmap-X can potentially estimate N more accurately and perform better.

We have not included results for Bitmap-X in the table and graphs above because of the difficulty of simulating an accurate version of Bitmap-X for a 32-bit address space in a reasonable amount of time and resources (roughly we need to maintain a 2^{32}-bit bitmap for each of the 2^{17} vulnerable hosts). We have, however, simulated it for a smaller 20-bit address space, together with Know-NI and Sum-Count-X for comparison [7]. In scaling back the size of the address space, we have also scaled back the number of vulnerable hosts (512) and worm scanning rate (250 per timestep) to keep the same propagation constant as our 32-bit parameterization.

Table 4 shows the speed and duration results for several heuristics for the 20-bit address space. The results indicate that the additional estimation accuracy of Bitmap-X provides little improvement over Sum-Count-X. These results are not surprising given that Sum-Count-X performs nearly identically to perfect estimators—Bitmap-X has little room to improve over Sum-Count-X.

4.4 Permutation Scanning

The discussion of permutation scanning in [6] as a technique to improve the propagation characteristics of a worm leads us to ask the following question: Is the performance of a self-stopping heuristic independent of a worm's scanning regime? To find the answer, we evaluate the improvement in speed and duration that strategies such as Greedy Permutation, Stop-k Permutation and Sum-Count-X Permutation provide over their respective random scanning analogs, Greedy, Stop-k and Sum-Count-X. Additionally, we study the effectiveness of a simple version of partitioned permutation scanning, where each infected host stops after scanning its assigned interval. Table 5 shows the speed and duration of the permutation scanning strategies and their random scanning analogs.

When comparing Greedy Permutation to Greedy, we see that permutation scanning provides a slight gain in speed because the Greedy Permutation worms are designed to avoid duplicating scan effort, and hence reach uninfected vulnerables sooner.

When we compare Stop-k Permutation to Stop-k, we see a similar improvement in speed as between the permutation and random Greedy variations. However, there is no improvement in duration because the few lingering hosts that remain must contact an already-infected host to deactivate, and the probability of doing so seems independent of the fact that a host is doing random or permutation scanning. (The difference in the probability of success between sampling with and without replacement is negligible when the number of possible successes, N, is small compared to the sample space size, A.)

On its surface, Partitioned Permutation seems like an optimal-speed strategy since assigning disjoint address intervals to worms should minimize duplicate effort. Counterintuitively, it is roughly 50% slower than Greedy Permutation. The reason is that the assignment of intervals to worm instances is rigid—a worm that is given a very small interval deactivates far too soon, since it could otherwise spend time assisting another worm that has been assigned a large, sparsely-populated interval. Nonetheless, Partitioned Permutation maintains a reasonable duration of roughly 280 timesteps because there is no duplicate scanning.

	Speed			Duration		
Strategy	10th	50th	90th	10th	50th	90th
Greedy	105	117	135	∞	∞	∞
Greedy Permutation	102	111	124	∞	∞	∞
Stop-3	113	127	139	410	446	515
Stop-3 **Permutation**	105	113	128	410	456	507
Partitioned Permutation	154	167	195	261	280	330
Sum-Count-X	103	119	136	156	174	199
Sum-Count-X Permutation	104	115	134	178	192	218

Table 5: Speed (timesteps to infect 85% of vulnerables) and duration (timesteps for all infected hosts to stop) by percentile of worm epidemics using permutation scanning and other strategies. Address space size $A = 2^{32}$, number of vulnerables $N = 2^{17}$, and per-host scan rate $\gamma = 4,000$ per timestep.

Finally, we compare Sum-Count-X Permutation against Sum-Count-X to see whether permutation scanning benefits from dynamic estimate exchange and vice versa. We see that permutation scanning slightly improves the speed of the epidemic, although the duration gets worse (increasing from 174 to 192 timesteps). We do not see an improvement in duration because the motivation of the immediate reseeding policy in Sum-Count-X Permutation is to *avoid* contacting infected nodes, whereas hosts under dynamic estimation *benefit* from contacting infected nodes to improve their estimates. To further explain this peculiarity, let us examine the qualitative behavior of these strategies.

Figure 3 shows the infected and active hosts over time for worm epidemics using random and permutation scanning variations of Greedy and Sum-Count-X, as well as Partitioned Permutation. As with previous versions of this graph, we show the results for runs with median duration out of 20 trials for each heuristic.

Figure 3a shows that Sum-Count-X Permutation propagates more quickly than Sum-Count-X, and Figure 3b demonstrates that the vast majority of hosts stop in Sum-Count-X Permutation before they do in Sum-Count-X. However, because the few remaining hosts in the permutation version are avoiding infected hosts by reseeding themselves after a redundant hit, their estimates take longer to converge. By contrast, a random scanning host is more likely to improve its estimates by being less discriminating in contacting infected nodes. We imagine that a retooled redundant-hit policy for Sum-Count-X Permutation (e.g., continue scanning using the current seed despite a threshold number of redundant hits) could eliminate these tail effects and lessen the duration.

Despite the sometimes opposing requirements of permutation reseeding and dynamic estimation, the effect that dynamic estimation has on decreasing duration is apparent since the duration of Sum-Count-X Permutation is noticeably shorter than Partitioned Permutation. Therefore, although permutation scanning improves the speed of a self-stopping worm, that technique alone cannot provide the dramatic stopping times of dynamic estimate exchange.

4.5 Vulnerability Density

In previous experiments, we have assumed a fixed vulnerability density (number of vulnerable hosts relative to the total number of hosts). Next we test this assumption and evaluate the sensitivity of the self-stopping heuristics to the vulnerability density. We simulate worm epidemics using the most effective heuristics across the categories: Greedy as an upper bound on speed, Know-N as an upper bound on estimation, Blind-32 for the basic heuristics, and Sum-Count-X for practical dynamic estimation. We vary the number of vulnerable hosts from 2^{11} to 2^{20} (roughly the number infected by Code Red [4]) and simulate 20 epidemics for each heuristic for each vulnerability density.

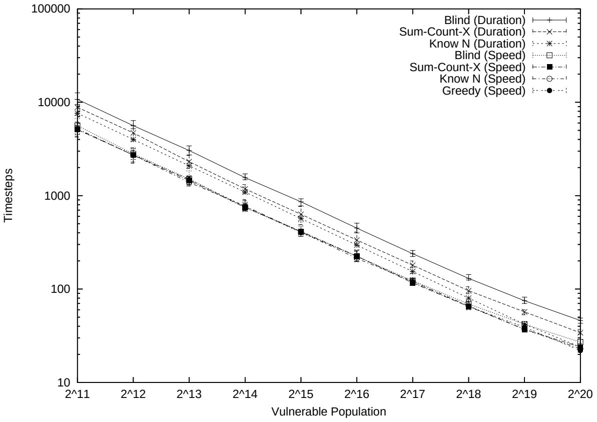


Figure 4: Speed and duration of self-stopping heuristics across a range of vulnerability densities. The points and error bars represent the 50th and 10th/90th-percentile results among 20 simulation runs. For each density, we tuned Blind's k parameter to infect 98% of the vulnerables. Note the log scale of the axes.

Figure 4 shows the results of this experiment. It graphs the speed of a heuristic to infect 85% of the vulnerable population as a function of the number of vulnerable hosts, and also graphs the duration of a heuristic to self-stop all active infected hosts at the end of an epidemic. We show two curves with error bars for each heuristic (except Greedy, which only has a speed curve). The points on a curve correspond to the median of the 20 simulated epidemics, and the lower and upper error bars correspond to the 10th and 90th percentiles among the 20 epidemics, respectively.

These graphs indicate that the relative performance of the heuristics is not sensitive to vulnerability density. In terms of speed, the relative ordering of the heuristics by their medians varies slightly over the wide range of densities, but still within the percentile intervals. In terms of duration, the heuristics preserve their relative ordering across the range of densities with very small variation. The curves appear to flatten towards high densities with slightly more spread among them. At these high vulnerability densities, however, practical issues like network capacity and contention will likely prevent worms from achieving suggested speeds.

4.6 Scan Traffic

Finally, we evaluate the network overhead of the various self-stopping heuristics. Although reducing network overhead is not an explicit goal of our heuristics, it does indicate their relative cost for spreading aggressively.

	Aggregate Scan Rate ($\times 10^8$)		Aggregate Scan Total ($\times 10^{10}$)	
Strategy	50th	90th	50th	90th
Greedy	5.24	5.24	∞	∞
Greedy Permutation	5.24	5.24	∞	∞
Know-NI	5.24	5.24	3.00	3.06
Know-I	5.24	5.24	2.89	2.99
Know-N	5.22	5.23	2.45	2.85
Blind-32	3.63	3.64	1.58	1.59
Stop-3	2.86	2.87	1.75	1.76
Stop-3 **Permutation**	3.54	3.55	1.88	1.89
Tree ($k = 50, m = 3$**)**	0.0754	0.0806	1.55	1.56
Partitioned Perm.	0.599	0.623	0.430	0.430
Sum-Count	1.53	1.74	1.54	1.79
Sum-Count-X	5.22	5.23	3.20	3.98
Sum-Count-X Perm.	5.20	5.21	3.63	4.57

Table 6: Aggregate scan rate and total by percentile of worm epidemics using various self-stopping strategies under a 32-bit address address space with 2^{17} vulnerables and a per-host scanning rate of 4,000 per timestep. We did not simulate the Bitmap heuristics.

Table 6 lists the peak aggregate scan rate and scan total for the heuristics. The peak aggregate scan rate is the number of scans in a timestep made by a worm across all infected hosts during the peak of worm activity (the peaks of the "active" curves in previous graphs). The aggregate scan total is the total number of scans made by a worm across all infected hosts for the duration of the epidemic. The table reports the 50th and 90th percentiles across 20 simulation runs for each heuristic to show variation among random epidemics.

As expected, the most aggressive heuristics (Greedy, Know-NI, Know-N, Know-I, and Sum-Count-X) have the highest scan rates and scan totals. Excepting Tree and Partitioned Permutation, the remaining heuristics have scan rates that are factors of 1.4–3.4 smaller and scan totals 1.7–2 times smaller than the most aggressive heuristics. Because Tree limits activity by each host, the number of simultaneously active hosts is relatively small. As a result, its peak scan rate is two orders of magnitude smaller than the other heuristics. Partitioned Permutation also has a low peak scan rate because the strategy restricts each host to its assigned interval. It has a slightly higher scan rate than Tree because all hosts are not restricted in the same manner, allowing older nodes to continue scanning despite multiple successful contacts. Furthermore, Partitioned Permutation has the lowest scan total because partitioning the address space means that worms can infect all hosts with a minimal number of scans (e.g., 4 billion for a 32-bit address space) to infect all hosts.

5. CONCLUSION

In this paper we demonstrate that scanning worms can use simple, practical heuristics to immediately stop scanning at the end of an epidemic in an entirely distributed fashion. Such self-stopping worms can adapt their scanning and stopping behavior to the dynamic conditions of the epidemic, requiring no parameterization or tuning. In particular, we describe the *Sum-Count-X* heuristic and show that random scanning worms can use it to realize nearly ideal spreading and stopping performance with trivial state overhead and no additional communication. Using this approach, hosts locally track the number of scanned and vulnerable hosts and exchange this information opportunistically upon contact with other hosts during the normal propagation of an epidemic. Our experiments also indicate that having hosts exchange these dynamic estimates is crucial for achieving such ideal stopping performance, even for more sophisticated scanning methods such as permutation scanning.

Finally, the feasibility of self-stopping worms reminds us that the window of opportunity to detect, contain and treat a network worm epidemic is short and getting shorter. Therefore, efforts to develop quick detection and containment systems remain of paramount importance for preventing infestation, as are post-hoc forensic capabilities for finding pockets of *sleeper* hosts whose infection is no longer obvious.

6. ACKNOWLEDGMENTS

We would like to thank our UCSD colleagues Jay Chen for challenging us to design better epidemic parameter estimators, Barath Raghavan for frequent and useful discussions, Marvin McNett for maintaining our systems, and Michelle Panik for proofing our text. As well, we are indebted to Nicholas Weaver, a partner in CCIED, who provided insightful feedback on permutation scanning and provided his simulation for cross-validation. Finally, we thank the anonymous reviewers for their helpful comments.

This work was supported through NSF grants CNS-0433668, CNS-0311690, a gift from Microsoft Research, and a grant from the UCSD Center for Networked Systems.

7. REFERENCES

[1] Brandon Wiley. Curious Yellow: The First Coordinated Worm Design. http://blanu.net/curious_yellow.html.

[2] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, British Columbia, Canada, Aug. 1987.

[3] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the Slammer Worm. *IEEE Security and Privacy*, 1(4):33–39, July 2003.

[4] D. Moore, C. Shannon, and J. Brown. Code-Red: A Case Study on the Spread and Victims of an Internet Worm. In *Proceedings of the ACM/USENIX Internet Measurement Workshop (IMW)*, Marseille, France, Nov. 2002.

[5] S. Staniford, D. Moore, V. Paxson, and N. Weaver. The Top Speed of Flash Worms. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, Fairfax, VA, Oct. 2004.

[6] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in Your Spare Time. In *Proceedings of the USENIX Security Symposium*, San Francisco, CA, Aug. 2002.

[7] N. Weaver, I. Hamadeh, G. Kesidis, and V. Paxson. Preliminary Results Using Scale-Down to Explore Worm Dynamics. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM)*, Fairfax, VA, Oct. 2004.