A Cooperative Immunization System for an Untrusting Internet

Kostas G. Anagnostakis* Michael B. Greenwald* Sotiris Ioannidis* Angelos D. Keromytis† Dekai Li*

*CIS Department, University of Pennsylvania
200 S. 33rd Str., Philadelphia, PA 19104, USA
Email: {anagnost,mbgreen,sotiris,dekai}@dsl.cis.upenn.edu

†CS Department, Columbia University
1214 Amsterdam Ave., New York, NY 10027, USA
Email: angelos@cs.columbia.edu

***Abstract*— Viruses and worms are one of the most common causes of security problems in computer systems today. Users attempt to protect machines from such attacks by using anti-virus programs and firewalls, with a mixed record of success at best. One of the main problems with these solutions is that they rely on manual configurations and human intervention, and may fail to react in time to defend against an attack.**

We present a cooperative immunization system that helps defend against these types of attacks. The nodes in our system cooperate and inform each other of ongoing attacks and the actions necessary to defend. To evaluate our proposal, we discuss a simple virus model and evaluate our system using simulation. Our measurements show that our algorithm is more effective against viruses and more robust against malicious participants in the immunization system.

I. Introduction

"Worms", introduced in [1] and inspired by a science-fiction novel [2], are self-replicating, segmented, distributed systems. If one segment of a worm is killed or fails, then the other segments detect the failure and spawn a new copy of that segment on a new node. Worms were originally considered a benign paradigm to ensure the longevity of distributed applications, and originally ran only on machines that exported either general or special purpose remote execution facilities.

Two factors have changed both the perception and reality of worms to be largely malignant platforms for distributed applications. First, even when programmers' motives are pure, small bugs can cause worms to proliferate and grow more rapidly than desired and overwhelm the resources of a distributed remote-execution system, as in fact occurred on the Internet in November 1988. Second, most worms or viruses no longer use legitimate remote execution interfaces to acquire a bounded number of nodes. Rather, they exploit bugs and loopholes and install themselves on machines where they are unwanted. They often try to grow without bound, attempting to infect every machine accessible to them. In the best case, these viruses simply steal cycles. However, they can easily have more destructive payloads: delete files and/or otherwise damage the host machines, steal sensitive data, participate in a denial of service attack, *etc.*

Reference [3] makes a credible case that viruses can cause truly immense damage to the network-dependent commercial, military, and social services infrastructure of nations throughout the world. That paper, and others, outlined a future world in which vigorous defense against viruses would rapidly detect virus attacks, and produce detectors, disinfectants, and antidotes. Such a world would likely contain the cyberspace equivalent of a Center for Disease Control, that would identify outbreaks, rapidly analyze pathogens, and distribute (authenticated) methods for detecting the virus and fighting the infection through both disinfection and immunization. Although such a Cyber Center for Disease Control (CCDC) can provide detection signatures as well as disinfection and immunization procedures for a given virus, it does not address the delivery mechanism for such measures. Application of preventive measures is likely to occur on human time-scales, but particularly aggressive viral attacks may be able to cover the entire Internet in a matter of minutes. Therefore it seems clear that some form of automated response must be employed.

Recent work has focused on such automated distributed mechanisms for containment [4] and disinfection [5] that may be able to spread fast enough to mitigate the effect of the virus. Some believe that there is reason for guarded optimism. Studies have shown that fairly low-levels of immunization [6] or low-level responses [7] can be enough to contain the virus or significantly slow the spread of the virus. The automated response mechanisms may be able to scan, filter, and disinfect or immunize quickly enough to prevent runaway infection and allow human intervention.

Our work is prompted by two observations that make earlier analyses that focus on individual viruses seem incomplete. First, while new viruses will continue to be created, zombie strains of old viruses will continue to circulate around the Internet by script kiddies. Further, a measurable fraction of Internet hosts will not bother to upgrade or patch to eliminate security bugs (or, worse, regress to vulnerable versions). Thus, the old viruses are still potentially virulent. Consequently, any proposed defense mechanism must be evaluated in the context of handling potentially many active viruses simultaneously. Second, distributed viral response mechanisms require some degree of trust between the automated agents cooperating in the response. In the best of times there are at most an insignificant minority of nodes in the Internet that any given node expects to be trustworthy; during a virus attack it is unreasonable to trust *any* particular node. Thus, any proposed defense mechanism must be robust in the face of inaccurate information from some of its peers.

These observations expose several significant problems that must be dealt with. Any node that responds to a potential virus carries a cost: a node has finite resources and therefore can only actively engage a limited number of viruses at a time.

Deciding to counter one virus entails ignoring some other virus. Further, filtering packets containing potential threats carries the possibility of false positives: legitimate traffic may be blocked, too. In the absence of cost, the best response to a potential virus attack is to flood the network as rapidly as possible, causing as many cooperating agents to respond at once. The main question is simply whether the response is quick enough to stifle the virus. In the presence of a cost model, however, we still need to respond quickly, but no more quickly than necessary. A false alarm, whether malicious or unintended, can trigger an effective denial-of-service attack by the response mechanism itself!

This paper investigates a distributed response mechanism that responds quickly to real viruses and does not over-react to false alarms. Moreover, the response mechanism must be robust against malicious agents spreading false information and be able to manage its resources even when many distinct viruses are active at any time.

Our basic approach is to share information about the observed rate of infection for each virus, verifying that new reports are not incompatible with our own empirical observations. We use a simple model of the virulence of a virus to (probabilistically) determine which viruses to respond to. We present an instance of such an algorithm, called COVERAGE (for COoperative Virus Response AlGorithm), and evaluate its effectiveness through large-scale simulation. Reference [5] comments that strategies effective against fast viruses will likely be ineffective against slow viruses, and that reducing sensitivity to false alarms increases the risk in real virus attacks. Although our results should be considered preliminary in the absence of long-term comprehensive testing and real deployment, they show that by dynamically modeling the spread of the virus through shared global information, COVERAGE can maintain effectiveness against both fast and slow viruses while reducing costs under a barrage of false alarms.

II. Related Work

Computer viruses have been studied extensively over the last several decades. Cohen was the first to define and describe computer viruses in their present form. In [8], he gave a theoretical basis for the spread of computer viruses. The strong analogy between biological and computer viruses led Kephart *et al.* [9] to investigate the propagation of computer viruses based on epidemiological models. They extend the standard epidemiological model by placing it on a directed graph, and use a combination of analysis and simulation to study its behavior. They conclude that if the rate at which defense mechanisms detect and remove viruses is sufficiently high, relative to the rate at which viruses spread, they can prevent widespread virus propagation.

The Code Red worm [10] was analyzed extensively in [11]. The authors of that work conclude that even though epidemic models can be used to study the behavior of Internet worms, they are not accurate enough because they cannot capture some specific properties of the environment these operate in: the effect of human countermeasures against worm spreading (*i.e.,* patching, filtering, disconnecting, *etc.*), and the slowing down of the worm infection rate due to the worm's impact on Internet traffic and infrastructure. They derive a new general Internet worm model called *two-factor worm* model, which they then validate in simulations that match the observed Code Red data available to them. Their analysis seems to be supported by the data on Code Red propagation in [12].

Moore *et al* [4] describes a design space of worm containment systems using three parameters: reaction time, containment strategy, and deployment scenario. The authors use a combination of analytic modeling and simulation to describe how each of these design factors impacts the dynamics of a worm epidemic. Their analysis suggests that there are significant gaps in containment defense mechanisms that can be employed, and that considerable more research (and better coordination between ISPs and other entities) is needed. Their analysis focuses exclusively on containment mechanisms (*i.e.,* network filtering), which they consider the only viable defense mechanism. We believe that other types of automated defense mechanisms will eventually be invented, if only because containment mechanisms can severely impact service availability.

Wang *et al* [6] presented some very encouraging results for slowing down the spread of viruses. It simulated the propagation of virus infections through certain types of networks, coupled with partial immunization. Their findings show that even with low levels of immunization, the infection slows down significantly. Those experiments looked at a single virus. Our work investigates the detection of multiple viruses when there is no *a priori* knowledge of which viruses may attack.

One approach for detecting new email viruses was described in [13], which keeps track of email attachments as they are exchanged between users through a set of collaborating email servers that forward a subset of their data to a central data warehouse and correlation server. Only attachments with a high frequency of appearance are deemed suspicious; furthermore, the email exchange patterns among users are used to create models of normal behavior. Deviation from such behavior (*e.g.,* a user sending a particular attachment to a large number of other users at the same site, to which she has never sent email before) raises an alarm. Information about dangerous attachments can be sent to the email servers, which then filter these out. One interesting result is that their system only need be deployed to a small number of email servers, such that it can examine a miniscule amount of email traffic (relative to all email exchanged on the Internet) — they claim 0.1% — before they can determine virus outbreaks and be able to build good user behavior models.

Reference [14] proposes the use of "predator" viruses that spread in much the same way malicious viruses do but try to eliminate their designated "victim" viruses. The authors show that predators can be made to perform their tasks without flooding the network and consuming all available resources. However, designers of predators would have to find their own exploits (or safeguard exploits for future use), which is not an attractive proposition. One encouraging result of their work was that the number of initial predators needed to contain a

highly-aggressive virus could be as small as 2,000.

The work most relevant to our is that of Nojiri *et al.* [5]. They present a cooperative response algorithm where edge-routers share attack reports with a small set of other edge-routers. Edge-routers update their alert level based on the shared attack reports and decide whether to enable traffic filtering and blocking for a particular attack. We compare the performance of this approach against COVERAGE in Section V.

III. System Model

In analyzing the behavior of our response mechanism through simulation, we necessarily make certain simplifications in our model of how viruses, hosts, switches, and our detection mechanism behave. The following subsections describe in turn these parts of our model.

A. Modeling Attackers

We use a fairly simple model to describe the behavior of potential attackers (viruses) that we consider in this research. After infecting a node, a virus attempts to infect other nodes; it may attempt to only infect a (small) fixed number of other nodes, or exhibit a greedier behavior. For our purposes, the distinction between the two types is simply in the probability of detection of a probe or attack by a detector. A virus may exhibit high locality of infection (*i.e.*, probing and attacking nodes based on network-topological criteria, such as "adjacent" IP addresses), or could use a random (or seemingly random) targeting mechanism, *e.g.*, using a large hit-list, or some pseudo-random sequence for picking the next address to attack. We expect that viruses that exhibit high locality are more difficult to detect using an Internet-wide distributed detection mechanism, but easier to do so on a local basis.

We can completely characterize a virus by the rate at which it attempts to infect other nodes and by the fraction of local attempts it makes. All attacks on susceptible nodes are successful, and in our simulation a virus never attempts to attack a non-existent node. (Therefore, our viruses are more virulent than equally aggressive viruses in the real world).

There are responses associated by the CCDC with each virus. We assume that by the time a node has enough information to decide to activate the virus detection mechanism, the CCDC has already disseminated the response to the node. The response for each detectable virus may have several optional components. An *attack detection mechanism* based on known virus signatures. In its simplest form, this is a packet filter; more complicated detection mechanisms (*e.g.*, checking for viruses inside email attachments) may also be part of the detection component. An *infection detection mechanism* also based on known virus signatures. This is a potentially expensive operation that may involve inspection of a node's disk or memory system. It can detect infections in already infected nodes. A *disinfectant mechanism* that can remove a virus from an already infected node. Finally, a *immunization mechanism* that can "protect" or immunize a node from subsequent attacks by this virus.

B. Network Topology

Our simulation topology is dictated by assumptions we make about the vulnerabilities and powers of network nodes with respect to virus attacks. We assume that, as a general rule, routers/switches are less likely to be infected by a virus, mostly because of the limited set of potentially-exploitable services they offer. Consequently, we assume that only hosts are susceptible to infection.

While considerable advantage can be had by exploiting the great levels of traffic aggregation seen in routers closer to the network core, it is unlikely that such nodes can actively check for viruses without significantly affecting their performance. Therefore, for the purposes of this paper, we assume that the only nodes in our system capable of scanning packet sequences for potential viruses are end-hosts or last-hop routers.

Our model of the network topology, then, consists entirely of a collection of *subnets*, or LANs, containing some number of *hosts*. Each subnet is connected to the global network through a single *router*. All routers are connected together in a single cloud where each router can address and forward packets to each other directly. End-hosts can only see traffic destined for themselves. Routers can inspect all traffic to or from their associated LAN.

It is likely that some organizations contain multiple subnets that frequently communicate among themselves. Therefore we collect together several subnets into a *domain*. A domain captures particular communication patterns but has no structural impact on the topology for simulation.

C. State of Nodes

A node in our environment can be in one of three states with respect to a virus: *susceptible*, *protected*, or *immune*.

A susceptible node can be either infected or uninfected. Susceptible nodes will become infected if subjected to an attack. Protected nodes may be infected or uninfected, but only if the detection module does not have the ability to detect and disinfect an infected machine. A protected node will not become infected as long as the protection mechanism (typically, a detection module that screens incoming packets or email) is in place. An immune node does not have the vulnerability exploited by the virus (either because it never had it — *e.g.*, different operating system — or because it was patched). Clearly, it is better to have an immunized node than to be forced to disinfect it after a successful virus attack. We assume that immune nodes are uninfected.

D. Operations

A node can monitor traffic and, for each virus, it can either ignore the virus or perform one or more of the following operations: collect and exchange information about a virus, scan for a virus, or filter viruses.

We assume that there is a cost inherent in checking for virus signatures. That is, a node cannot be actively "on the lookout" for an arbitrary number of viruses without adversely affecting its performance. (Some experimental measurements of such real-world limits are given in [15]). Edge-routers are more

likely to be constrained by high packet rates, and therefore limited in the amount of scanning they can perform. Hosts can scan for more viruses without interfering with their (lower) packet rate, but, on the other hand, have work other than packet forwarding to perform. In either case there is an upper bound on the number of viruses a node can scan for.

We assume that nodes periodically exchange information about viral infections. Although the per-virus cost of such an exchange is low, we assume that the number of known viruses exceeds the amount of information that can be reasonably exchanged at any given time. Thus, actively exchanging information about a virus incurs a cost, albeit lower than scanning.

Routers can additionally scan for viruses on all packets travelling to or from their LAN (and drop when necessary). We further assume that if a router detects a rampant viral infection for a virus that has an associated disinfectant component, the router can invoke a disinfection operation (perhaps alerting an administrator) on all the nodes in its LAN.

E. Model of Anti-virus Epidemic

Each node participating in the anti-virus response must make certain decisions: (a) the rate at which it polls other local nodes for virus information, (b) the rate at which it polls other remote nodes, chosen at random, for virus information, (c) whether, for each virus, to collect information about it, (d), whether to include that information in virus exchange packets, and (e) whether to scan for the virus (collecting the results of those scans as part of the local information for that virus).

IV. COVERAGE: The Cooperative Virus Response Algorithm

COVERAGE tries to balance the cost of scanning and filtering packets for a specific virus against the benefit of detecting, other, real viruses in several ways.

First, COVERAGE models the virulence of viruses and ranks them accordingly. With probability proportional to their virulence, COVERAGE decides in rank order whether to actively scan for the virus or not. It stops making decisions, and scans no more viruses, once the scanning schedule consumes the entire scanning budget available. Second, each COVERAGE agent exchanges information about the state of a virus with other cooperating agents in order to construct a model of the virus and determine whether incoming reports are empirically consistent with the observed state of the network. Third, COVERAGE agents determine their polling rate to maximize the probability of seeing enough viruses to confirm the current local estimate of the virus state, while reducing the probability that communication will add no new knowledge to either of the participants. We now describe the algorithm in more detail.

A. COVERAGE algorithm

Agent communication. Each COVERAGE agent polls other agents, selected randomly. Assuming that only a small fraction of the nodes are reporting false information, a randomly selected node is more likely to be trustworthy than a node that actively contacts us — a small number of malicious nodes may try to flood the rest of the network. At each poll, the sender reads the response and updates its local state variables to track the operation of the cooperative response mechanism and the status of the network in terms of observed attacks.

First, it records whether the remote agent is actively scanning. This allows the agent to estimate the fraction of agents in the network that are actively scanning for a particular virus.

Second, it updates estimates of possible infections *e.g.,* the fraction of infected nodes for each virus. We distinguish two types of estimates: direct and remote. Direct estimates are updated based on whether each remote agent has directly observed an attack (either to itself or, if a router, to a node in its LAN). Remote estimates are updated based on the fraction of infected nodes as estimated by the remote agent (the "direct" estimates of the remote agent).

Periodic updates. At regular intervals each COVERAGE agent updates its state based on the information received since the last update. To track the progress of the infection each COVERAGE agent maintains a smoothed history for each type of estimate (direct and remote), each as exponentially decaying averages with varying time constants, to approximate recent infection rate, past rate, and background rate.

Using these estimates, an agent can compute the fraction of nodes believed to be infected as well as the growth of the infection, assuming exponential growth[1]. If p_n is the fraction of infected nodes at timestep n and α is the growth rate, the progress of the infection at timesteps $n+1$ and $n+2$ is expected to be:

$$p_{n+1} = p_n(1+\alpha),\ p_{n+2} = p_n(1+\alpha)^2, \ldots$$

Taking the direct estimates p_1^d, p_2^d, and p_3^d maintained by the agent, we can obtain estimates $\alpha_\star^d$ and $p_\star^d$. We calculate the *virulence*, v^d, of a virus as the estimated number of timesteps needed by the virus to infect the entire network. This estimate is: $v^d = -\log p_\star^d / \log(1+\alpha_\star^d)$.

Using the same method as above the agent can also compute $\alpha_\star^r$, $p_\star^r$ and v^r based on the remote estimates.

Scanning/filtering. Given the estimates an agent can decide whether it needs to scan for a given virus. There is a basic, low level of scanning for every virus. When a virus becomes active the scanning rate may increase. In the general case, the agent can sort viruses in order of their virulence v^d and decide whether to scan for each virus, in turn, stopping when the scanning budget is filled. (In our simulation, we only scan viruses whose v^d is below *threshold*.)

An inactive agent, A, may also start scanning seemingly low-virulence viruses, if *enough other* agents claim the virus is virulent, and A finds that the fraction of scanning nodes is too low to detect virus activity in a single timestep at the

[1]We assume all growth is exponential for the purpose of deciding whether to trigger a reaction. We believe that linear growth worms can be detected by humans, and need not be countered by an automatic, distributed, algorithm. If our assumption is incorrect and growth is, in practice, sub-exponential then we recover naturally because we observe a decrease in α and gradually back-off as the predicted "virulence" of the virus drops.

current polling rate. The test is whether n (simply the fraction of agents that were polled and found to be scanning in the last interval) is less than twice the estimated fraction of infected hosts (*e.g.*, if $n < 2p^r_\star$). Similarly, if the agent is active but $n > p^r_\star$ then it decides to stop. The agent also stops scanning if $\alpha^r_\star$ approaches 0. This ensures that the fraction of scanning agents is bounded if there is insignificant progress for a given infection or if the infection is small compared to the number of actively scanning agents. Such heuristics are essential for controlling the behavior of the algorithm, keeping the response mechanism "ahead" of the virus but also limiting the damage and cost when malicious agents spread false information.

Polling rate. An agent communicates with agents within the same domain at a constant, high rate, as the cost of intra-domain communication is assumed to be very small. Inter-domain communication is generally more expensive; agents therefore need to adapt the rate of polling remote agents, avoiding excessive communication unless necessary for countering an attack. When there is no virus activity, agents poll at a pre-configured minimum rate (at least an order of magnitude lower than the rate for intra-domain communication). An agent periodically adapts the remote polling rate if v^r is less than a given threshold. The new rate is set so that the agent polls $1/(p^r_\star)^2$ remote agents in each update interval, unless this rate exceeds a pre-configured maximum rate. This is used to increase the polling rate when the remote estimate indicates that an attack is imminent (but not yet reflected in the direct estimate). If the more recent direct estimate p^d_n is non-zero, then the polling rate is increased so that at least a few samples can be collected in each update interval. Finally, if the estimated virus population $p^r_\star$ is very small (*e.g.*, $< 10\%$) and the estimated virus growth rate is close to zero, the agent throttles back its remote polling rate to the minimum rate.

These adjustments are always performed on the polling side. We avoid changing the state or behavior of the polled agent to reduce the risks associated with malicious agents. Otherwise they could spread false information and raise false alarms more effectively by increasing their own communication rate.

V. Simulation Results

We focus on the behavior of COVERAGE in response to a single virus. We model the impact of multiple active viruses by specifying a threshold under which a virus will not have high enough priority to be scheduled in the scanning budget. If many viruses are active then the threshold will be a small number, such as 5 (recall that the virulence is a measure of how many measurement intervals it will take before the virus has covered the *entire* Internet). Unless the current virus is poised to conquer the entire net at its current rate of growth from its current coverage within `threshold` intervals, it will not have high enough priority to be scheduled in the scanning budget. We compare the performance of our algorithm to the NRL03 algorithm described in [5].

A. Results

An example run of the COVERAGE algorithm against a virus is shown in Figure 1. One can see the initial stage of the infection and the response of the algorithm: the virus manages to infect roughly 10% of the hosts; cooperation between COVERAGE agents results in a rapid activation of filtering on roughly 60% of the network effectively eliminating the virus. Soon after stopping the attack, the COVERAGE agents deactivate scanning/filtering.

We simulate a simple, relatively small network of 100,000 edge-routers, each connected to 8 hosts. The network contains 2,000 domains consisting of 50 edge-routers each. We examine the performance of the COVERAGE algorithm against the NRL03 algorithm. For the COVERAGE algorithm, we set the local-domain polling interval to 1.8 seconds , the maximum and minimum remote polling intervals to 6 seconds and 1.8 seconds respectively. For both algorithms we assume that 4% of the edge-routers are permanently scanning for the virus.

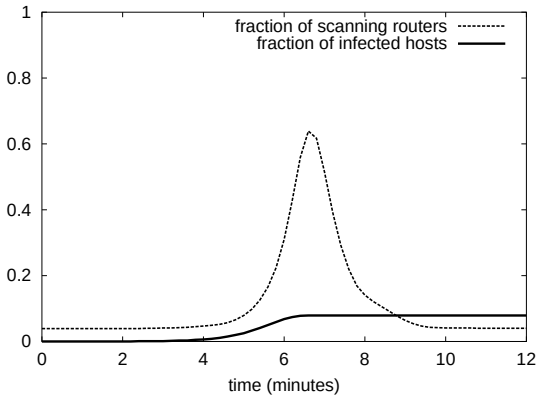

Fig. 1. Fractions of infected hosts and scanning edge-routers over time.

Our analysis uses two key metrics. In the case of a virus attack we are interested in the maximum fraction of infected hosts. In the case where a set of malicious attempt to spread false information to confuse the system, we are interested in the maximum number of false scanning/filtering edge-routers.

The maximum fraction of infected hosts for different virus infection rates is shown in Figure 2. We see that COVERAGE is significantly more effective than NRL03 for slow and medium-fast viruses but the relative benefit is reduced for faster viruses. NRL03 achieves the same maximum infection independent of virus infection rate because alert communication is triggered by virus scan events and follows a push model. In contrast, communication rate in COVERAGE is bounded and rate adaptation is delayed. This has been a deliberate design choice in an attempt to make the algorithm robust against false information from malicious nodes.

We also examine how COVERAGE and NRL03 perform in a setting involving a small fraction of malicious participants that attempt to spread false information. The measured maximum number of false scanning/filtering nodes for a given fraction of malicious hosts is presented in Figure 3. We see that COVERAGE is significantly more robust to malicious nodes compared to NRL03. Considering the results of Figure 2 it becomes clear that for both algorithms there's a trade-off between effectiveness against real viruses and robustness

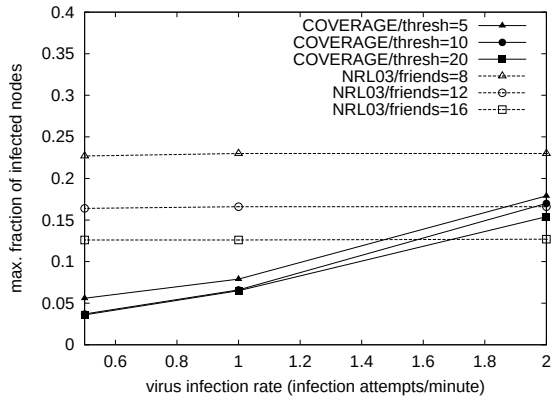

Fig. 2. Maximum fraction of infected hosts vs. virus infection rate

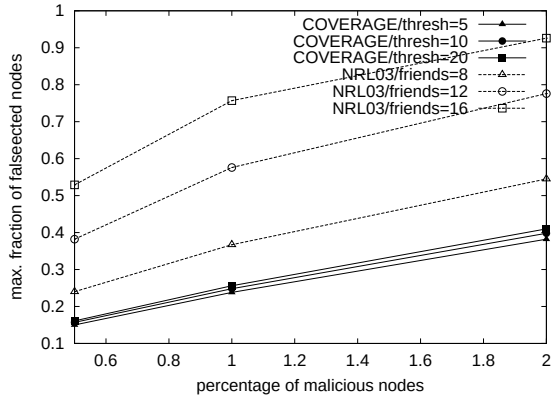

Fig. 3. Maximum fraction of false scanning edge-routers.

against malicious participants. The results show that while NLR03 is bound to perform poorly in one of the two metrics, COVERAGE achieves a more balanced trade-off. We believe that this is a valuable property for our approach.

VI. Conclusions and Future Plans

We have examined how cooperative pooling of information can slow the spread of malicious viruses in the Internet. We have described an algorithm that allows cooperating agents to share information about such attacks and use this information for controlling the behavior of detection and filtering resources. What makes the design of such systems difficult is the need to perform well without fully trusting global information. This is crucial as full trust could result in costly false alarms if a fraction of the participating agents are malicious. We have proposed one solution to this problem, based on the idea of carefully sampling of global state to validate claims made by individual participants. Simulation results confirm that this method is effective in limiting the damage of virus attacks and that it is robust against attacks by malicious participants.

There are several directions for further experimentation. First, in this paper, we have focused on reducing the cost of scanning without describing much of our efforts on reducing the communication cost of polling. We have algorithms that should reduce communication cost without measurably reducing effectiveness of the anti-virus mechanism (it is possible to significantly back-off polling rate when the polls do not significantly change the state variables). These algorithms have not been refined by experimentation, nor extensively tested yet. Second, more experimentation is needed to examine the performance of COVERAGE in the case of multiple, perhaps simultaneously active viruses. Due to space considerations, this paper simply assumes that many COVERAGE nodes are occupied by higher-virulence viruses, and that we must reserve processing cycles to deal with lower virulence viruses, too. Much work remains to be done in improving the actual choices each node makes of which set of viruses to monitor. Finally, it is necessary to consider a richer network model involving end-hosts and core-routers as well as more representative network topologies.

Acknowledgments

This work was supported by the DoD University Research Initiative (URI) program administered by the Office of Naval Research under Grant N00014-01-1-0795.

References

[1] J. Shoch and J. Hupp, "The "worm" programs – early experiments with a distributed computation," *Communications of the ACM*, vol. 22, no. 3, pp. 172–180, March 1982.

[2] J. Brunner, *The Shockwave Rider*. Canada: Del Rey Books, 1975.

[3] S. Staniford, V. Paxson, and N. Weaver, "How to Own the Internet in Your Spare Time," in *Proceedings of the USENIX Security Symposium*, August 2002, pp. 149–167.

[4] D. Moore, C. Shannon, G. Voelker, and S. Savage, "Internet quarantine: Requirements for containing self-propagating code," in *Proceedings of 22nd Annual Joint Conference of IEEE Computer and Communication societies (INFOCOM 2003)*, April 2003.

[5] D. Nojiri, J. Rowe, and K. Levitt, "Cooperative response strategies for large scale attack mitigation," in *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition*, April 2003.

[6] C. Wang, J. C. Knight, and M. C. Elder, "On Computer Viral Infection and the Effect of Immunization," in *Proceedings of the 16th Annual Computer Security Applications Conference*, 2000, pp. 246–256.

[7] J. Kephart and S. White, "Directed-graph epidemiological models of computer viruses," in *Proceedings of the IEEE Symposium on Security and Privacy*, 1991, pp. 343–361.

[8] F. Cohen, "Computer Viruses: Theory and Practice," *Computers & Security*, vol. 6, pp. 22–35, Feb. 1987.

[9] J. O. Kephart, "A Biologically Inspired Immune System for Computers," in *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*. MIT Press, 1994, pp. 130–139.

[10] "CERT Advisory CA-2001-19: 'Code Red' Worm Exploiting Buffer Overflow in IIS Indexing Service DLL," http://www.cert.org/advisories/CA-2001-19.html, July 2001.

[11] C. C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, November 2002, pp. 138–147.

[12] D. Moore, C. Shanning, and K. Claffy, "Code-Red: a case study on the spread and victims of an Internet worm," in *Proceedings of the 2nd Internet Measurement Workshop*, November 2002, pp. 273–284.

[13] M. Bhattacharyya, M. G. Schultz, E. Eskin, S. Hershkop, and S. J. Stolfo, "MET: An Experimental System for Malicious Email Tracking," in *Proceedings of the New Security Paradigms Workshop*, September 2002, pp. 1–12.

[14] H. Toyoizumi and A. Kara, "Predators: Good Will Mobile Codes Combat against Computer Viruses," in *Proceedings of the New Security Paradigms Workshop*, September 2002, pp. 13–21.

[15] K. G. Anagnostakis, M. B. Greenwald, S. Ioannidis, and S. Miltchev, "Open Packet Monitoring on FLAME: Safety, Performance and Applications," in *Proceedings of the 4rd International Working Conference on Active Networks (IWAN'02)*, December 2002.