Investigating the Impact of Real-World Factors on Internet Worm Propagation

Daniel A. Ray[1], Charles B. Ward[1], Bogdan Munteanu[1], Jonathan Blackwell[1], Xiaoyan Hong[1], and Jun Li[2]

[1] Department of Computer Science
University of Alabama, Tuscaloosa, AL 35487
[2] Department of Computer and Information Science
University of Oregon, Eugene, OR 97403
lijun@cs.uoregon.edu

Abstract. This paper reports the results of our experimentation with modeling worm behavior on a large scale, fully adaptable network simulator. Our experiments focused on areas of worm scanning methods, IP address structure, and wireless links that, to the best of our knowledge, have been mostly neglected or abstracted away in prior worm simulations. Namely, our intent was to first study by direct observation of our simulations the effects of various IP scanning techniques on the effectiveness of worm spread. Second, our intent was to research the effects that having a larger IP address space (specifically a sparsely populated IP address space like that provided by Internet Protocol Version 6) would have on the effectiveness of several worms. Third, we study how the wireless media may affect the propagation of worms. In order to perform these simulations we have made use of the Georgia Institute of Technology's network simulator, GTNetS, extending the worm classes packaged with the simulator.

1 Introduction

The propagation of Internet worms has a devastating effect on the normal operations of the Internet. Because of the cost of worm attacks, much research has recently been devoted to trying to understand, detect, and prevent the spread of Internet worms. While various analytical modeling and empirical analyses have been conducted to study the propagation nature of various Internet worms, the effects of real-world factors on the propagation of worms with various scanning methods are still not fully understood. In this work, we study two major factors on the propagation of Internet worms—IP address distribution and wireless media—while worms can choose various scanning methods.

– *IP address distribution.* While relatively dense, the current IPv4 address space still has a large portion of unallocated addresses. When a worm scans for victims, its success rate is affected by whether or not a scanned address is allocated. Studying this may potentially help design a more worm-resistant

P. McDaniel and S.K. Gupta (Eds.): ICISS 2007, LNCS 4812, pp. 10–24, 2007.
© Springer-Verlag Berlin Heidelberg 2007

IP address allocation policy. In addition, it also warrants further study to know how worms may propagate in the much larger IPv6 address space.

– *Wireless media.* Every day many nodes are connected to the Internet through wireless media using WLAN, WiFi, upcoming Mesh networking, Bluetooth PAN or 3G cellular technologies. Little is known regarding the speed and style Internet worms may propagate through these wireless media to users. Various networking choices (e.g., a single large subnet vs. dispersed subnetworks) or access control techniques (MAC address filtering vs. password protection) may or may not affect the behavior of worm propagation. User mobility could have both a positive and negative impact on the worm spread. To the best of our knowledge, no work has been extensively performed in this area.

Internet worms can scan in many different ways. Scans can be random, local preference, hitlist-based, permutation, topological, or some combination of these. It can be based on either TCP or UDP, or even piggybacked onto other networking traffic. With the various scanning methods, we investigate the impact of IP address distribution and wireless media on Internet worms spread through a systematic, comprehensive analysis that compares the propagation speeds and trends in an Internet-like networking environment.

We use a packet-level network simulator, *GTNetS*. Our work identifies and explores parameters affecting worm-propagation which have been largely ignored, abstracted away, or overlooked in previous studies. Most previous simulation studies of worms have not simulated worm behavior at a per-packet level; instead, they rely on certain analytical models to reduce the computational complexity which could be substantial when simulating a worm outbreak on a network of non-trivial size. Even relatively small networks take an enormous amount of CPU time to simulate at a packet-level because of the massive traffic load needed to be simulated as a matter of course when analyzing worm behavior. However, with improved simulators, and the ever increasing computational power available, simulating outbreaks at a per-packet level on networks of non-trivial size has been shown to be feasible. For example, work has been done by George F. Riley and his colleagues at Georgia Tech [1,2].

The rest of this paper is organized as follows. We first describe previous work in Section 2. We then describe our GTNetS-based simulation environment in Section 3, and illustrate our approach in detail in Section 4. Our results are presented in Section 5. We conclude our paper and point out future work in Section 6.

2 Previous Work

Worms pose a substantial threat to the Internet and much work has been done to study real worm outbreaks [3,4,5,6,7,8]. The previous work is generally divided into two categories, analytical modeling and empirical simulations. In addition, researchers have studied various approaches in detecting worms, such as [15]. Theoretical analyses of worms have been performed as far back as [9], but not

until Staniford et al. systematically categorized and analyzed the threats from various worm propagation techniques in [10] was the threat really well understood. Another study by Staniford et al. [21] was on flash worms, showing that using a pre-computed spread tree, an UDP worm could infect up to 1 million hosts in half a second, or in 1.3 seconds if the worm uses TCP. In [22], address space distribution is modeled far from uniform. The authors analyze the impact of unused blocks of the IP space and provide a model for implementing a distributed traffic monitor system. Further work on worm propagation models [11,12] and the potential sophistication of worms [13,14] also show that worms are an ever-increasing threat that will not be easily stamped out.

The previous work on worm simulations has not focused on the key issues of our research. Namely, these are the effects of varying worm IP block scanning methods and the effects of a larger IP address space and other changes provided by IPv6, as well as the effect of wireless media. Work in [21] focuses on one scanning strategy - hit-list, while our work analyzes multiple scanning methods as well as different topologies and IP address distributions. Also, compared to work in [22], we study the effect of a much larger IP address space while also taking into consideration multiple network topologies.

2.1 Analytical Modeling

Analytical modeling is a significant area of research. Internet worm research, in particular, has made use of analytical modeling to study worm behavior. Generally speaking, the idea of analytical modeling is to, through analysis of a problem domain, define and apply a mathematical model that fits the system being analyzed within an acceptable margin of error. Analytical models benefit from computational efficiency because calculations are largely independent of the size of the network. Generally, however, there will be some necessary uncertainty inherent in the relationship between the mathematical model and real world behavior. On top of this uncertainty, analytical models are often easily implementable and cannot interact with any proposed detection and defensive mechanism or varying network and worm parameters without altering the mathematical model on which they depend [16].

Chen, Gao, and Kwiat [12] give an analytical model for the spread of worms which they dub Analytical Active Worm Propagation (AAWP) model. Their AAWP model characterizes the propagation of worms that employ random IP-block scanning. This model is a prime example showing that analytical models are suitable for the study of Internet worms. However, the model's mathematical complexity as well as its inflexibility (especially in the method for handling IP-block scanning) shows that it still suffers from problems inherent to all analytical models.

2.2 Empirical Simulations

With the problems inherent in analytical models many researchers have begun a push to use empirical simulations to test hypotheses concerning the behavior

and propagation of worms in various network models. Our research falls in this vein. To our knowledge, however, there have been no papers published that focus on the comparative effects of varying IP-block scanning techniques as well as the effects of a larger address space.

As discussed before, Riley et al. [1,2] give an interesting framework for how Internet worms might be simulated using the GTNetS simulator.

Wei, Mirkovic, and Swany [17] perform research on worm simulation on simulators that are very similar in nature to GTNetS. Their simulations exhibit high-fidelity through their use of packet level simulations. They also provide some flexibility for varying worm types, though not to the level proposed by our research. For instance, they developed classes for worms with random scanning and subnet scanning, but have not extended the classes beyond these approaches. Wei et al. also expound upon the limitations of GTNetS. Mainly, GTNetS requires a good deal of centralized computational power. The simulator from Wei, et al. is a distributed simulator that uses normal PC nodes to run their distributed simulation algorithms and treats the PC nodes as existing at the Autonomous System network level. Unfortunately, this distributed algorithm suffers from the complexity inherent in distributed computing.

Weaver, Staniford, and Paxson [18] present an algorithm for worm propagation retardation which they call the Threshold Random Walk (TRW) algorithm. They begin with this algorithm for containment of scanning worms and try to make *ad hoc* changes to general purpose system's hardware and specific choices for software implementation that work together to form an *ad hoc* simulator that is suitable to test their algorithm and various hypotheses. This approach turns our approach, in which we move from a general simulator to a specific worm implementation, on its head, and is a much less flexible approach.

3 GTNetS-Based Worm Simulation Environment

The work by Riley et al. has resulted in the development of a fully adaptable real-world network simulator that is capable of supporting the modeling of Internet worms. Some simple worm examples have already been created for use with the simulator.

GTNetS is unique amongst computer network simulators in that it is designed to allow for large-scale packet-level network simulations. There are several issues that must be solved in order to allow for simulations of these kinds due to the extreme demands on memory and computational power. We will briefly discuss how GTNetS solves these issues. For a full explanation, refer to [1,2].

3.1 Simulating Large-Scale Networks

GTNetS has been designed from the ground up to provide for large-scale network simulations. GTNetS uses several approaches to limit the computational complexity and memory use of simulating networks on a large scale, thus freeing

us from these considerations. First, GTNetS addresses the memory complexity of such simulations by employing NIx-Vectors [19] for routing simulation. This approach does not maintain routing tables, but rather uses a source-based method in which routing information is stored in each packet. As such, routes are computed only as needed, and are cached at packet sources for later use. This approach is useful because normal routing tables in network simulations cause a large overhead in memory.

Also in an effort to reduce memory complexity for the simulator, special consideration is taken of leaf nodes or subnetworks of leaf nodes that are known to have a single gateway router as an access point to the rest of the network. GTNetS first attempts to route the packet within the subnetwork (this will only be possible if leaves in the subnetwork are interconnected). If this is not possible then the packet is unconditionally forwarded to the gateway router. This simple step saves a large amount of memory, and is an advance over other simulators.

Second, GTNetS addresses the complexity inherent in maintaining an "event list" by attempting to control the size of the list. The event list is the list of events (sending packet, receiving packet, etc.) that the simulator must simulate. The first method for controlling the size of this list is to use FIFO receive queues which, they explain, will limit the number of events necessary in the event list for receiving packets. Also, they note that in many cases the queuing delay for a packet in a FIFO queue (such as a basic Drop Tail queue) can be deterministically calculated. Thus, GTNetS uses "Abstract Queuing" such that information about transmitted packets is stored deterministically and packets are never queued at the transmitter. Instead, these packets are given the appropriate queuing delay and sent directly to the receiver. Finally, GTNetS uses a data structure called a "Timer Bucket" which is used to abstract out network delays such as the round-trip time (RTT) in order to model TCP timeout events in an efficient way and thus reduce the size of the event list.

Third, and finally, GTNetS reduces the computational and memory complexity of simulating large-scale networks by limiting the size of log files that are normally kept by simulators for storing the results of a simulation. It does this by providing pre-packaged statistical packages that can create the desired statistics and allow for the removal of raw fine-granularity files from the kept log files.

3.2 The Worm Simulation Environment

As we mentioned above, not only is GTNetS specifically designed for large-scale networks, it is also highly adaptable to various network environments. This turns out to be very useful, especially in the simulation of worms. A major concern of our research, as well as all research in the area, is exactly which elements of the network environment to hold constant and which elements to test against. We discuss these decisions in depth below. Here, however, we discuss exactly what options are available to us via the GTNetS simulator.

The first and foremost concern is the network itself; that is, the network topology. This is important because GTNetS is fully flexible in this regard, both in network structure, bandwidth, and IP address assignment. GTNetS provides a robust interface for creating network graphs, including limited functionality for generating random network topologies fitting certain regular patterns. These include common graph types such as star graphs, dumbbell graphs (bandwidth bottleneck graphs), and trees. However, it does not natively provide support for generating more complicated types of random networks. Fortunately, it does provide a facility whereby random graphs can be imported from random graph generation programs such as BRITE, which generates random graphs which mathematically resemble the AS structures of the existing Internet. Other graph generation tools such as iNet (another Internet-like topology generator) can also be used [20].

Second, but no less important, is the structure of the worm packets themselves which help define the worm's structure. GTNetS is capable of supporting worm packet classes of either TCP or UDP. Thus, worms with varying infection lengths, selected infection ports, and varying number of connections can be appropriately simulated. Also very important to our research is the fact that GTNetS allows for worm classes with varying IP block scanning and selection patterns via the use of a standard extensible C++ class, as well as varying scan rates.

In short, the full flexibility of GTNetS allows us to gather empirical data from simulations that are based on a great variety of simulated networks and worm types that behave as in the real world. Thus, our research results should very closely mimic empirical observations of the real world, without putting any hardware at risk or causing billions of dollars of damage.

4 Our Approach

Our approach centers around the manipulation of code provided with the GTNetS simulator to facilitate our simulations. Our first step was to design a working model on a single processor. As described above, the GTNetS simulator is designed to provide simulations with characteristics of real networks in mind. Thus, there are several variables for both network related and worm related characteristics that needed to be addressed. Specifically, what needed to be decided was which variables would be held constant across the spectrum of our simulations. As we previously stated, the goal of our research is to discover the effects of varying IP address population density and scanning methods on the propagation of worms. With that in mind, we made decisions concerning network and worm characteristics as described below.

4.1 Constants

There are really two types of constants that we must address. We will deal with network-related characteristic constants first. Prime among these was the

network topology itself. Rather than deal with the uncertainty that multiple network topologies would bring into our simulations, we decided that the best course of action was to select a suitable topology and to hold this topology constant across various simulations. GTNetS comes pre-equipped with a simple network topology generator. However, this generator only accepts the number of required nodes and arranges the nodes in patterns which are unlike real-world networks. Because we are primarily interested in Internet worms, we decided to create a network topology that would be more characteristic of the Internet. In order to do this, we created a network using the BRITE graph generation program. BRITE creates Internet-like AS and subnet structures at random. After creating a network topology we assigned IPs at random with respect to subnets such that subnets would have IPs within a given range. The network topology was held constant, but we eliminated unnecessary overhead by not holding IP addresses constant over all the simulations.

The wireless networks are treated differently. They are subnets in the Internet topology with different link bandwidth and packet loss rate. A wireless subnet can be a large address space that adheres to a higher level on the topology tree or a small one to a lower address topology tree. For example, a campus wireless network can use a campus-wide universal subnet that accommodates a large number of IP addresses, or a small subnet that builds within each department. These two types of configurations are modeled using GTNetS directly as: a *wider-tree* with a few levels in a topology tree and lowest level has large amount of fan out links for the first case, and a *deeper-tree* with more levels and less fan out links in the lowest level.

There is also the issue of network traffic that is not worm related. In dealing with this issue we simply abstracted out all other network traffic. We do so with full realization that congestion in one network subnet could affect worm propagation. However, for ease of simulation, and because congestion is partly a factor of the network topology itself (i.e. number of nodes in a subnet, speed of connections, etc.), we have chosen to focus only on "ideal" networks where congestion is uniform (or non-existent). Future endeavors may deal more directly with this issue.

Finally is the issue of individual node vulnerability. GTNetS decides worm vulnerability by assigning a vulnerable state to individual generated nodes according to some probability. In an effort to make our simulated network as Internet like as possible we held this probability constant across our simulations and assigned the probability to 90%. We arrived at this number by determining that approximately 97% of machines on the Internet run Windows operating systems of one variety or another, thus making them the main targets of most attacks. We then reduced the overall percentage marginally to account for instances where, for whatever reason, a machine may be running a Windows O.S. and not be vulnerable to attack, thus arriving at the approximated value of 90% vulnerability across our simulated network.

Next, we deal with worm related characteristic constants. Because our research focuses only on the worm scanning method all other worm characteristics were

held constant with the exception of that one. Scan rate, infection length, infection port, and the number of TCP connections have all been held constant at default GTNetS settings across simulations.

4.2 Variables

Obviously, those network and worm characteristics that we have not held constant must still be dealt with. In fact such characteristics allow us to make observations as to how certain aspects affect worm propagation. We vary the IP address space population density to simulate the effects of implementing IPv6 (with its larger address space) on the Internet. Specifically, our simulations were set to use approximately one IP address in every thirty-five available IP addresses for dense (IPv4 like) networks, and one in every one hundred and thirty-four for sparsely populated (IPv6 like) networks. These numbers are best guess efforts at assigning Internet like IP addresses. For IPv4 approximately 75% of available /8 blocks have been assigned. However, there are many IP addresses within each block that are still available. Unfortunately, specifics about individual networks are not readily available, so we have made our best guess in assigning 1/35 addresses for our simulations [11]. We further noted that the full address space of IPv4 is much less than one percent of the entire address space of IPv6. However, for the sake of the computational time of our simulations, and to obtain more meaningful results, we assigned our simulations of IPv6 address space to have just less than 1% of address spaces to be occupied.

There are also two worm characteristics which varied across our simulation. First, we obviously varied the worm scanning method across our simulations. GTNetS provides simple worm scan methods including uniform random IP scanning, and local preference scanning. We have added to these hit-list scanning which assumes that worm propagation does not begin until a certain set of key nodes has been pre-infected.

Finally, in the case of hit-list scanning, unlike in the other cases, we have run simulations using both TCP connections and UDP connections. All of our other simulations have been run using UDP connections with a constant default scan rate because UDP worms are most effective. So, in order to compare high and low density address space simulations using hit-list scanning with our other simulations we have first run UDP hit-list worms. However, because we are already spending time to select certain nodes across the network to pre-infect, we wanted to test what effect a TCP hit-list worm would have. As such, we have run an additional TCP hit-list worm simulation for both high and low density address space populations.

4.3 Simulations

This subsection, in way of an overview of the previous subsection and a precursor to the next section, gives a complete account of the simulations which we have performed. For each UDP worm the scan rate has been held at a constant default

value of 100, and for each TCP worm the number of connection threads has been held at a constant default of 3. Also, in the case of all simulations, the infection length and infection port, which we have not concerned ourselves with as we have abstracted out all other traffic, have been held at constant default values. Finally, each worm type simulation has been run twice, once with a densely populated (IPv4 like) IP address space and once with a sparsely populated (IPv6 like) IP address space. Densely populated networks were generated such that 1/34 of the available address space is used, and sparsely populated networks were generated such that 1/134 of the available address space is used.

Each of the following worm simulations was run as described above. First was the UDP uniform random scanning worm. Second was the UDP local preference scanning worm. Next, we generated a simulation using a UDP hit-list worm using local preference scanning after the hit-list is established. Finally, the same hit-list local preference scanning worm was simulated again, only this time TCP connections were used to propagate the worm. The results of these scans are given in the next section of this paper.

4.4 Results

The final aspect of our approach involves exactly what statistics we have chosen to acquire from our simulations. Using the data provided by GTNetS and the program gnuplot, we have plotted interval infection rates for our various simulations. This allows us to provide a graphical representation of how well a worm is able to propagate across a network at given time intervals. In each case our simulations were run until all 1000 nodes were infected. The graphs that are provided in the next section each contain at least two simulation plots for comparative purposes.

5 Analysis of Results

Our results, like our simulations, are divided into worm classes based on IP block scanning methods as described above. Below we attempt to both give an overview of results for individual worm classes as well as results from comparisons of our results.

5.1 Comparing Worm Types on Sparse and Dense Graphs

First we consider Figure 1 and Figure 2. The two worm types are the uniform random scanning and the local preference scanning. We tested each of these two methods in both dense and sparse graphs. There are several areas of interest on the curve of the infection interval. The first is the slope of the curve itself. If the slope is steep, then the worm has infected the nodes very quickly. The less steep the curve is, the longer it has taken the worm to infect the network's vulnerable nodes. This is true for each of the graphs provided.

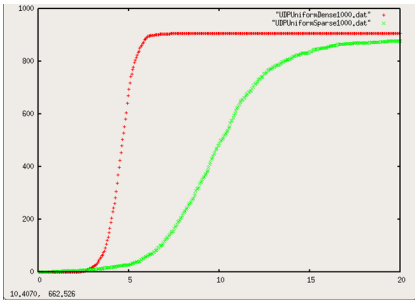

Fig. 1. Comparison of Uniform Random Worms

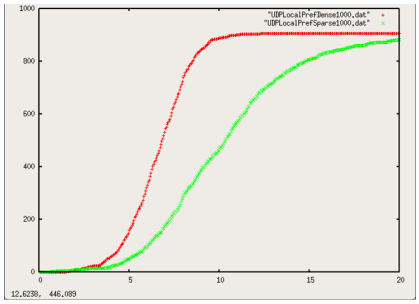

Fig. 2. Comparison of Local Preference Worms

The graphs of Figure 1 and Figure 2 show that the worms which are operating on dense graphs are much more successful than the worms that are operating on sparse graphs. In other words, if the network is sparsely populated then the worm has much more trouble finding and infecting vulnerable nodes quickly. This is not unexpected, but it does present a good argument for migrating the Internet to IPv6. However, it is interesting to note that it has still taken about the same amount of total time to infect all vulnerable nodes for both sparse and dense graphs in the case of these simulations.

5.2 Comparing Sparse and Dense Networks Overall

We now look at comparing the effects of sparse and dense networks overall. Figure 3 gives a side-by-side comparison of all three worm types on a dense network and Figure 4 gives a side-by-side comparison of all three worm types on a sparse network. Here we add worm type of hit-list pre-seeding for both dense and sparse networks. In this section we attempt to shed more light on exactly what the effects of dense and sparse networks are by comparing the three together.

For the case of dense networks we can examine Figure 3 to see our graphical comparison of infection intervals. What we note from this graph is that the curves of the infection intervals themselves are very similar. This is not unexpected because the worm types of hit-list and local preference scanning use the same IP block scanning method. However, more importantly, we can actually see the effect of the pre-seeding of worm infected nodes in various subnetworks from the hit-list worm. The hit-list worm infection interval is shifted to the left, indicating an overall increase in virulence.

As for the case of sparse networks we examine Figure 4. What we find is that for hit-list and local preference plots the slope of the graph is flatter. This effect is the same effect as noted before for worms working on sparse networks. However, we still see, as was hinted at above, that the hit-list worm infection interval is shifted to the left.

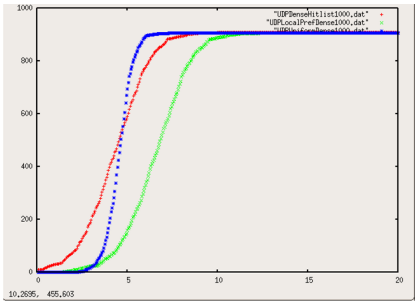

Fig. 3. Comparison of UDP worms on dense graph

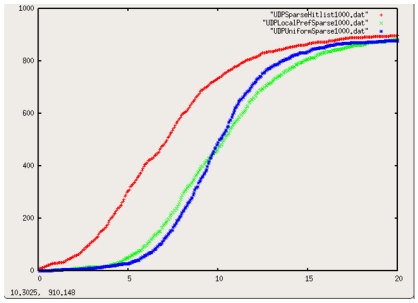

Fig. 4. Comparison of UDP worms on sparse graph

These graphs highlight that the order of the effectiveness of the three worm types is not changed drastically due to the change from densely to sparsely populated graphs, though uniform random scanning worms suffer a greater flattening effect than the others. The graphs also further indicate that a sparsely populated network is going to suppress the effectiveness of worm spread, regardless of worm type.

5.3 Comparing TCP and UDP Hit-List Worms

Finally, we examine the tests we ran on hit-list worms with local preference scanning using TCP and UDP connections respectively as in Figure 5 and Figure 6. Dense and sparse networks are compared. The most obvious thing to note is that the simulation for sparse networks is unfinished for TCP hit-list worm. The reason for this is that the overhead for creating a TCP worm is exemplified in the simulation itself. As a result, the machine on which the simulation was run ran out of memory before the simulation could complete, even after a substantial memory upgrade. We discuss options for further research with more computational power in the final section. However, what we can tell from this simulation is that the overhead is substantial.

Further, what we can tell by comparing dense and sparse networks is that the hit-list worm with TCP connections is an improvement in the case of dense networks, but it has no real effect in the worm spread for sparse networks. Not surprisingly, TCP hit-list worms work better on dense networks and do not work as well with sparse networks. In short, overall TCP hit-list worms seem to add significantly to the overhead of hit-list worms, especially in sparse networks, without adding any benefit to worm spread.

5.4 Worm Propagation in Wireless Media

We simulate wireless media in two different network architectures, namely, a wired Internet backbone (or high bandwidth link) with local WLANs directly

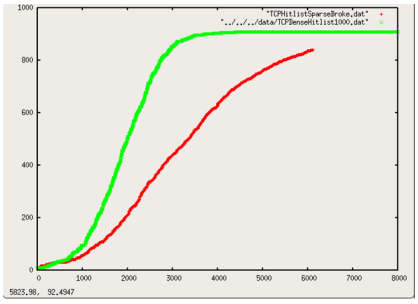

Fig. 5. Comparison of TCP Hit-list Worms

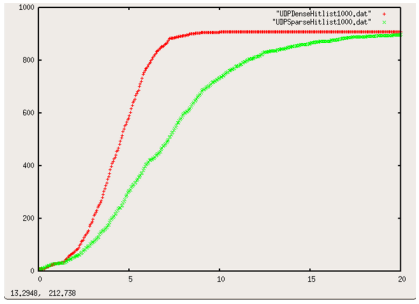

Fig. 6. Comparison of Hit-list Worms

attached to it, and a leveled organizational network tree with the WLAN subnets penetrated as smaller subnets at lower levels. A *wider-tree* topology is designed to reflect the first network architecture and a *deeper-tree* topology represents the second case. In the following Figures, curves numbered 1 are results run on *deeper-tree* topology and curves numbered 2 are results run on *wider-tree*. We compare both TCP and UDP worms with uniform scanning and local preference scanning methods.

Figure 7 shows a comparison of TCP worms using uniform scan in the two topologies. The Figure shows that worms start infection much later and propagate much slower in the deeper-tree topology than them in the wider-tree topology. Figure 8 shows a comparison of TCP works using local preference scanning. Similar trends as observed before are present here as well.

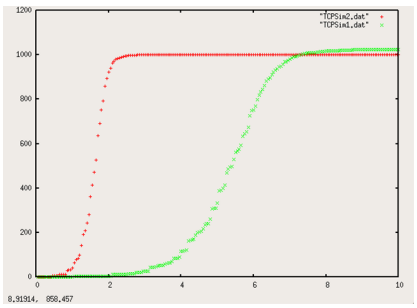

Fig. 7. Wireless TCP Uniform Worms

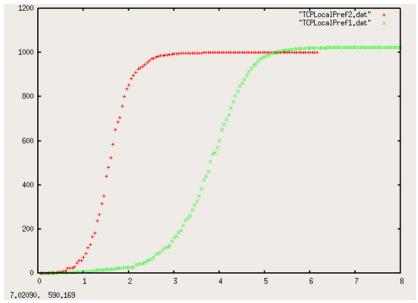

Fig. 8. Wireless TCP Local Preference Worms

Figure 9 and Figure 10 show a comparison of UDP worms using uniform and local preference scan in the two topologies respectively. Again, the Figures show that deeper tree topology can slow down worm propagation due to the fact that

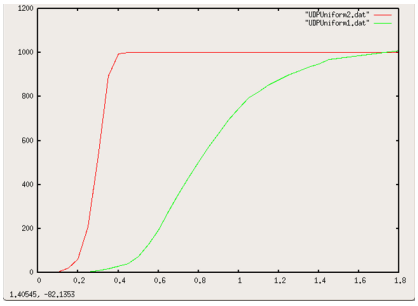

Fig. 9. Wireless UDP Uniform Worms

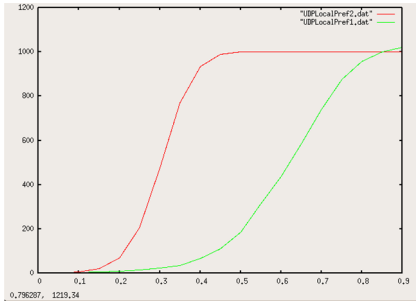

Fig. 10. Wireless UDP Local Preference Worms

more links need to be searched for the preferred addresses. These results suggest that the commonly used WLAN configuration of creating a single direct subnet for wireless access as in parallel to other organizational subnets may not be a preferred network topology in hampering worm propagation speed.

6 Conclusion and Future Directions

In this paper we investigated two major factors that impact the propagation of worms with a few major propagation methods. Especially we investigated the influence of IP address space and wireless links. We use GTNetS, a detailed network packet level simulator to conduct the evaluation. Our simulation results show that worms propagate slower in IPv6 address space due to the sparse address allocation. The results also show that WLAN configuration impacts the worm propagation. A deep tree type of topology can slow down the propagation.

Certainly, there are many future directions in which this research could head. Prime among these is the distribution of the simulations amongst various processors to give us more computational power. Originally, our hope was to be able to distribute the simulations in some manner for this reason. Indeed, GTNetS is designed to be distributable. However, due to lack of access to proper computer hardware and documentation, at this time we have not been able to do so. In the future, distribution of the simulation processes will allow many networks of much greater size to be analyzed.

Also, another direction is the implementation of more unusual worm types which we chose not to implement due to the exotic nature of the worms and the foundational nature of our initial research. Namely, these worm types are those which use permutation scanning and topological scanning to obtain a measure of synchronization among worms; this could prove to render the worms more effective in sparse networks if the overhead of such synchronization could be managed.

Finally, future research could take advantage of other scanning methods than local preference for hit-list worms. Local preference was chosen for this research

because it seemed to be a best fit for a worm type that is pre-seeded in different subnets. However, this hypothesis could be tested by choosing different scanning methods.

References

1. Riley, G.F.: Large-scale network simulations with GTNetS. In: Proceedings of the 2003 Winter Simulation Conference (2003)
2. Riley, G.F., Sharif, M.I., Lee, W.: Simulating internet worms. In: Proceedings of MASCOTS (2004)
3. Spafford, E.: The Internet worm: Crisis and aftermath. Communications of the ACM 32(6), 678–687 (1989)
4. Arron, E.J.: The Nimda worm: An overview (October 2001), `http://www.sans.org/rr/catindex.php?cat_id=36`
5. Moore, D., Shannon, C., Claffy, kc.: Code-Red: A case study on the spread and victims of an Internet worm. In: Proc. ACM Internet Measurement Workshop, pp. 273–284 (2002)
6. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: The spread of the Sapphire/Slammer SQL worm. CAIDA, La Jolla, CA, Tech. Rep. (2003), `http://www.caida.org/analysis/security/sapphire/`
7. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: Inside the Slammer worm. IEEE Security and Privacy 1(4), 33–39 (2003)
8. Yegneswaran, V., Barford, P., Ullrich, J.: Internet intrusions: Global characteristics and prevalence. In: Proc. ACM SIGMETRICS, pp. 138–147. ACM Press, New York (2003)
9. Kephart, J.O., White, S.R.: Directed-graph epidemiological models of computer viruses. In: Proc. IEEE Symposium on Security and Privacy, pp. 343–361 (May 1991)
10. Staniford, S., Paxson, V., Weaver, N.: How to 0wn the Internet in your spare time. In: USENIX Security Symposium, USENIX, Berkeley, CA, pp. 149–167 (August 2002)
11. Garetto, M., Gong, W.: Modeling malware spreading dynamics. In: Proc. IEEE INFOCOM, vol. 3, pp. 1869–1879. IEEE Computer Society Press, Washington, DC (2003)
12. Chen, Z., Gao, L., Kwiat, K.: Modeling the spread of active worms. In: Proc. IEEE INFOCOM, pp. 1890–1900. IEEE Computer Society Press, Washington, DC (2003)
13. Zou, C.C., Gao, L., Gong, W., Towsley, D.: Advanced polymorphic worms: Evading IDS by blending in with normal traffic. In: Proc. Conference on Computer and Communications Security (2003)
14. Chen, Z., Ji, C.: A self-learning worm using importance scanning. In: Proc. Workshop on Rapid Malcode (2005)
15. Zou, C.C., Gao, L., Gong, W., Towsley, D.: Monitoring and early warning for Internet worms. In: Proc. Conference on Computer and Communications Security, pp. 190–199. ACM Press, New York (2003)
16. Sharif, M., Riley, G., Lee, W.: Comparative study between analytical models and packet-level worm simulations. In: PADS (2005)
17. Wei, S., Mirkovic, J., Swany, M.: Distributed worm simulations with a realistic internet model. In: PADS (2005)

18. Weaver, N., Staniford, S., Paxson, V.: Very fast containment of scanning worms. In: USENIX Security Symposium, USENIX, Berkeley, CA, pp. 29–44 (2004)
19. Riley, G.F., Fujimoto, R.M., Ammar, M.: A generic framework for parallelization of network simulations. In: MASCOTS. Proceedings of Seventh International Symposium of Modeling, Analysis and Simulation of Computer and Telecommunication Systems (1999)
20. Winick, J., Jamin, S.: Inet-3.0: Internet topology generator, `http://topology.eecs.umich.edu/inet/inet-3.0.pdf`
21. Staniford, S., Moore, D., Paxson, V., Weaver, N.: The top speed of flash worms. In: WORM 2004. Proceedings of the 2004 ACM workshop on Rapid malcode, pp. 33–42. ACM Press, New York (2004)
22. Rajab, M.A., Monrose, F., Terzis, A.: On the Effectiveness of Distributed Worm Monitoring. In: USENIX Security Symposium (2005)