Multiscale Modeling and Simulation of Worm Effects on the Internet Routing Infrastructure

David M. Nicol, Michael Liljenstam, Jason Liu

Department of Electrical and Computer Engineering
University of Illinois, Urbana-Champagn

Abstract. An unexpected consequence of recent worm attacks on the Internet was that the routing infrastructure showed evidence of increased BGP announcement churn. As worm propagation dynamics are a function of the topology of a very large-scale network, a faithful simulation model must capture salient features at a variety of resolution scales. This paper describes our efforts to model worm propagation and its affect on routers and application traffic. Using our implementations of the Scalable Simulation Framework (SSF) API, we model worm propagation, its affect on the routing infrastructure and its affect on application traffic using multiscale traffic models.

1 Introduction

The last two years have seen wide-scale worm infestations across the Internet, e.g. Code Red[1] (July 2001), nimda[2] (September 2001), and SQL-Slammer[3] (January 2003). Worms of these types use an activity called *scanning* in order to propagate[6]. An infected host enters a loop in which it repeatedly samples at random an IP address, and either attempts to open a connection with a device at that address (success at which leads to further attempt to impregnate the device, using another packet), or simply sends a packet which, if accepted by a susceptible and as-yet-uninfected device, infects it.

Publicity surrounding these events focused on the effects to hosts, and (in the case of Slammer) ATM machines. What is not commonly known is that the worm spread also affected devices that execute a protocol which determines how traffic is to be routed through the Internet. This protocol, the Border Gateway Protocol[7] (BGP), operates by having routers exchange prospective paths, to every point in the Internet. Every message sent in the course of executing BGP is in principle the result of a failure or closing of a communication session—somewhere—between two routers. One such failure may cause a cascade of messages to propagate through the BGP routing infrastructure. So-called "withdrawal" messages are of particular interest, as these are proclamations that the sender no longer knows of *any* path to the subnetwork named in the withdrawal. Analysis of BGP message traffic across the Internet shows that the global BGP system generated an abnormally high number of messages while the worms scanned, and in some cases generated an abnormally high number of withdrawal messages.

In the wake of the Code Red and nimba events we became quite interested in understanding and modeling the relationship between the worm scans and BGP's behavior. In addition to the direct evidence of increased BGP activity, we received anecdotal evidence of increased router CPU utilization, and reports of router crashes. Router failures can explain the observed BGP behavior[5].

Routers run real-time operating systems, and it is known that excessive CPU utilization at the highest OS priority level frequently precedes router failure, as essential background services become starved. One of the explanations we developed—eventually developed and corroborated by advisories from router manufacturers—is that that cache thrashing was a leading contributer to router failure. All routers acquire a packet's forwarding port by lookup into a forwarding table. In high end routers the entire forwarding table is in CPU speed memory. Routers at the lower end of the price/performance spectrum use a high-speed cache to hold forwarding directions for recently seen destinations, and schedule a significantly slower lookup from slow memory—at high CPU priority—when a cache miss occurs. The scanning behavior of worms, when intense, destroys the locality of reference needed for any caching system. Traffic diversity causes a high rate of cache misses. Address Resolution Protocol (ARP) behavior was likewise affected, as local area networks received worm packets whose addresses were legitimate to subnetwork level, but whose full IP address failed to match any device in the subnetwork.

We initially developed simulation models of the worm propagation, router dynamics, and BGP behavior with the goal of experimenting with hypothesized causes of the observed behavior. As those hypotheses were validated we expanded the effect to include other models of traffic, including application traffic critical to support of network-wide data gathering and distribution. This effort has resulted in a new package for the SSFNet[8] simulation system. In this paper we describe how this package models traffic at different temporal and physical scales, all within the same simulation system. We illustrate the need for multi-scale modeling in an application that sought to assess the effectiveness of worm defense mechanisms in a large-scale (but private) network.

2 Application : Effectiveness of Worm Defenses

The worms seen to date have heightened awareness of their threat, and raised interest in deploying defensive mechanisms for detecting and containing them. Simulation and modeling provides a critically needed approach for assessing the effectiveness of contemplated measures.

We consider an example of a international-scale enterprise network (meaning that the network is isolated from the rest of the Internet). The network owner is able to deploy specialized hardware or software to detect worm scans, and react to them by quarantining subnetworks suspected to be infected. The sorts of questions the owning organization asks include

- Can the counter-measures be effective at stopping worm spread? If so, how does one optimize placement and parameters of those counter-measures?

– What are the effects on critical network applications when worms attack, and what impact do the counter-measures have on those applications?
– What are the tradeoffs between the cost of defense and risk of having no defense?

We consider two specific mechanisms for detecting worm scans. One of them uses a slight modification to router software which sends a "blind-copy" ICMP control message to a repository, when the router is presented with a packet whose destination cannot be reached[4]. The idea here is that random scans will select IP addresses that don't exist, so-called "back-scatter". If one knows what fraction of IP space in a subnet is unreachable, and assumes some sampling distribution for the scans, then measured misses can be used to estimate the scanning intensity. This estimated intensity can be thresholded, and reaction taken when suspicion of scans is high. Another mechanism requires specialized hardware. Some fraction of the packets at a network access point is diverted to the device, where analysis is done (e.g., source/destination, hash functions of packet content) to produce one or more signatures for a packet, which are put into a cache. The idea is that packets from a common infestation have a great deal of structural similarity (e.g. the infection payload), so that detection of an abnormally high number of similar packets may signal the presence of a worm. Further sophistication is possible when these network devices fuse information from back-scatter with common content signals, and analyze global propagation growth patterns to provide early warning of a worm's advance.

In order to answer the questions listed earlier we'll need to address pressing modeling issues. Worm spread is dependent on the distribution of vulnerable hosts throughout the enterprise, the nature of the mechanisms it uses to spread, the effect the worm has on the network infrastructure, how the infrastructure reacts to the impact the worm has on it, and the topology of that infrastructure. Thus we see that to accurately capture worm dynamics we will have to model network-wide topology in sufficient detail to capture the interactions between worm and infrastructure. The size of the network forces us to model worm scanning traffic at a fairly coarse time-scale. At another scaling extreme, we need to model the impact that worm traffic has on the caching behavior of individual devices. Our interest spans the ISO stack model as well. We are interested in the behavior at the highest layer (the application layer), as well as behavior at the next-to-lowest layer (the data-link layer).

3 Multiscale Models

To execute multiscale network simulation models it is necessary to construct specialized models of routing devices, designed to simultaneously manage traffic flows with different characteristics, at different levels of abstraction, while properly interacting with each other.

3.1 Traffic

The approach we take is to first differentiate between individual packets (or frames), and abstracted flows. Packets and frames are handled individually. A "flow" gives a higher level of abstraction. At any point in simulation time, at any point in the network, a flow is characterized by the rate at which bits are moving. Flows may have other characteristics as well:

- fixed update epoch—a fixed time-step after which a flow's rate is updated,
- dynamic changes—some flows may alter rates dynamically,
- interactive—a flow's rate change is a function of dynamic quantities in the simulation,
- target subnet(s)—a set of IP prefixes may be specified as the targeted recipients of the flow. This general form allows specification of a single destination IP address, or a number of entire subnets as destination.
- scanning behavior—some flows may represent worm scans,

The first distinguishing characteristic of a flow is whether changes to its rate happen at fixed epochs, or dynamically as a function of the simulation state. While time-stepped updates are the norm, we have also developed discrete-event fluid formulations of UDP and TCP which allow for arbitrary spacing between update instants. A second characteristic is whether a rate update depends on the model state or not. It is legitimate to model some fraction of background traffic with rate functions that are completely specified before the simulation is run, and which are not altered while the simulation is running. These flows help provide context for the evolution of other flows whose behavior is of more interest. Interactive flows have rate updates that affect, and are affected by other elements of the model state. A third characteristic of a flow is its destination set. We allow a flow to have an arbitrary number of A third characteristic is scanning behavior. We describe the topology of a scan as a one-to-many flow. The destination set is described by a set of IP prefixes (subnets). This allows us to model a worm that has a target list, as well as a worm whose scans are purely random. Scanning flows are notable in that the flow splits at a router, with the destination prefixes and incoming flow rate being partitioned among outgoing. The fraction of incoming scan flow which is carried along an outgoing link is identically the fraction of incoming scan flow target IP addresses that are reached over that link. Standard multicast can be modeled with a minor variation which does not split the incoming flow rates, but duplicates them on outgoing links.

3.2 Routers

Our implementation of SSFNet contains routers that handle diverse traffic models, concurrently, modeling their interactions. Packet flows may be part of this mix. The state of fluid flows in the router are used to estimate packet loss probability, queueing delay, and bandwidth allocation. The packet streams affect the

interactive fluid flows by having a fluid representation whose rates are based on observed packet arrival behavior.

BGP speaking routers are modeled to capture the interaction between worm scan traffic and router behavior. This model is also multiscale. It contains a detailed model of BGP operations, with BGP speakers communicating using full packets over TCP. The processing time of BGP messages is governed by an estimate of background CPU utilization. CPU utilization goes up as the intensity of scan traffic coming in or going out increases. BGP memory utilization is modeled too, as a function of the number of flows and rate of scan traffic.

Our model of a BGP speaker has an artificial layer in the stack model which triggers router failure. The CPU and memory utilitzation states are checked periodically (e.g. every simulated second). A decision is made randomly to fail the router. The probability of failure is a function of the CPU and memory utilizations, naturally being monotone non-decreasing as either of those utilizations increase. Upon failure a router remains inoperative for a down time, typically measured in tens of minutes. This simple model is intended to capture complex dynamics within a router of the effects that scanning traffic has on it, from both the ingress and egress sides.

BGP speakers that share sessions are required to send each other messages (if only "I'm alive") every 30 seconds. The rest of the BGP system notices a failure by sensing that it is no longer sending messages. This in turn triggers BGP announcements concerning subnetworks whose accepted paths passed through the failed router. New paths for those subnetworks are announced, if possible, else withdrawals are announced.

Thus there is a causality chain, where scan traffic intensity affects utilization, which affects BGP processing costs and a failure model, both of which affecting BGP behavior. In turn, BGP affects scan intensities, because as BGP modifies forwarding tables it affects the paths that scan flows take.

3.3 Worm Infection

Our worm infection model takes an abstract view of subnetworks (unless the modeler specifies a subnetwork in detail). For most global purposes the key attributes a subnetwork needs to have identify how many vulnerable devices there are, and how many are infected (with the possible addition of state models describing how many devices are in each state of a finite-state-machine description of an infected host's behavior). We advance propagation dynamics in a time-stepped fashion, at each time-step calculating what the incoming scan intensity to a subnetwork is. This intensity is used in conjunction with the knowledge of the size of the IP space being scanned (which contains the subnetwork's address space) and the number of vulnerable-but-uninfected devices in that subnetwork, to randomly choose the number of devices which newly become infected. One might also associate an infection duration with a device to model detection of the infection and removal (either of the worm or the the device).

An important point to consider is that the total scanning rate into the subnetwork is a function of all infected subnetworks throughout the entire network.

This single fact forces us to model infection behavior at a high level of abstraction.

3.4 Worm Detection

As we have described earlier, worm scans may be detected by the presence of common data payloads and by presence of backscatter. Our models allow the representation of such devices, and the communication between them, with the effect that when certain thresholds are reached a subnetwork that is suspected of being infected quarantines itself. When this happens no traffic enters or leaves the subnetwork. The idea behind quarantine is to contain the worm in those subnetworks it has infected, leaving the uninfected networks fully operational.

Just as the flow intensities of scan traffic affect the state of a router and so affect BGP, those intensities help to determine the rate of backscatter detection, and the rate of common content detection. The estimates of those rates are thresholded to trigger communication between detection devices, and eventually quarantine of some subnetworks.

4 Network State and Application Behavior

We have considered two kinds of network applications, both modeling functionality that provides a Common Operational Picture to the enterprise (that is, the availability of all data, to any place in the enterprise). Three types of application traffic are modeled. The first is point-to-point traffic, as would be used to browse web sites. We also represent multi-source to single destination traffic, which models the convergence of information to some critical decision maker in the enterprise. We also represent single source to multiple destination traffic, to model hot spots in data provisioning.

The network state affects networked applications through its impact on the bandwidth they receive, the end-to-end latency, and packet loss. These variables all are affected by the intensity and placement of worm scans, the failure of routers, and access to IP addresses representing application traffic sources and sinks.

5 Example

We have used the concepts described in this paper to evaluate the impact of a fast-moving worm attack on a large network modeled loosely on the NIPRNet. Our model involves 130 Autonomous Systems (AS), 3233 routers (of which 559 are BGP speakers), and represents 163 LANs. The worm dynamics are modeled after Slammer, which essentially infected all vulnerable machines in a few minutes. The experiment begins with 6 infected hosts in one AS, latent until time 300 seconds, after which they begin to scan (the first 300 seconds are used in the simulation to allow BGP to initialize and converge on forwarding tables.

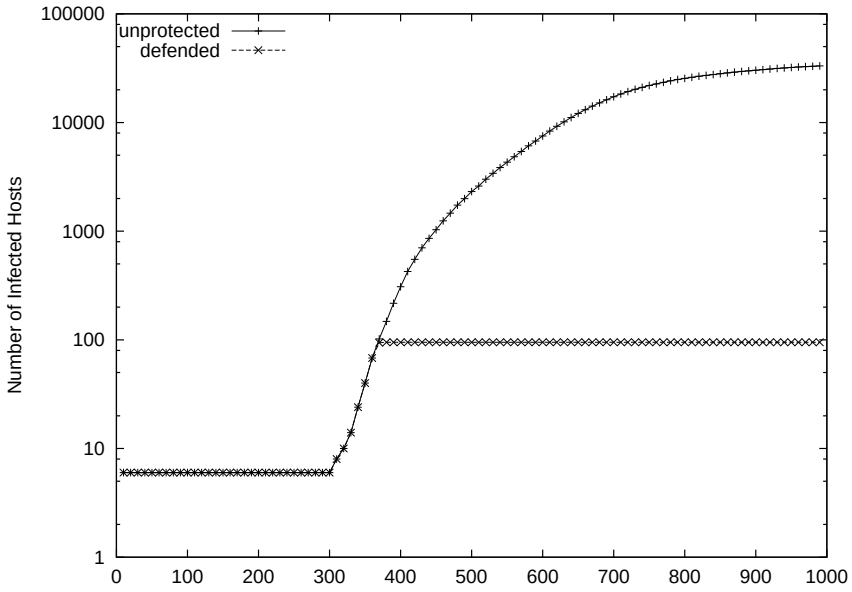

Fig. 1. Number of infected hosts as a function of time, with and without quarantine defenses

In this experiment we use a variety of traffic models. The background traffic was generated off-line using fluid models of TCP where the background flows interacted with each other. Time series of bandwidth use during these experiments were recorded, and are used as non-interactive background traffic in the worm experiments. The time-step for background traffic rate updates is approximately 5 seconds. Application traffic is fluid-based, but is interactive. Rate updates occur every second. BGP message traffic is packet oriented, and is handled discretely rather than continuously.

This model uses a variety of resolutions for devices as well. BGP speaking routers are modeled in detail. Non-BGP speaking routers are represented more in terms of the effects they have on traffic bandwidth and latency than they are in terms of actual forwarding tables. Non-BGP routers within an AS are assumed to use the OSPF protocol, which essentially maintains shortest-path information within an AS. Device failures cause our simulation to recompute the shortest path information. LANs are represented very abstractly, with just enough detail and state information to capture worm propagation dynamics. We assume that the outbound link between the LAN and the rest of the Internet is the bottleneck on flows entering and leaving the LAN, and so do not explicitly model the interior of the LAN (although this would certainly be possible, as would a mixture of abstract and concrete LAN representations).

Figure 1 shows the number of infected hosts (on a log scale) as a function of time, for both the situation where we deploy worm defenses and when we

don't. The unprotected case shows the characteristic exponential growth curve with tail-off we expected of such worms. We see that the worm defenses detect the worm just over a minute after it begins scanning, and effectively isolates the infected networks.

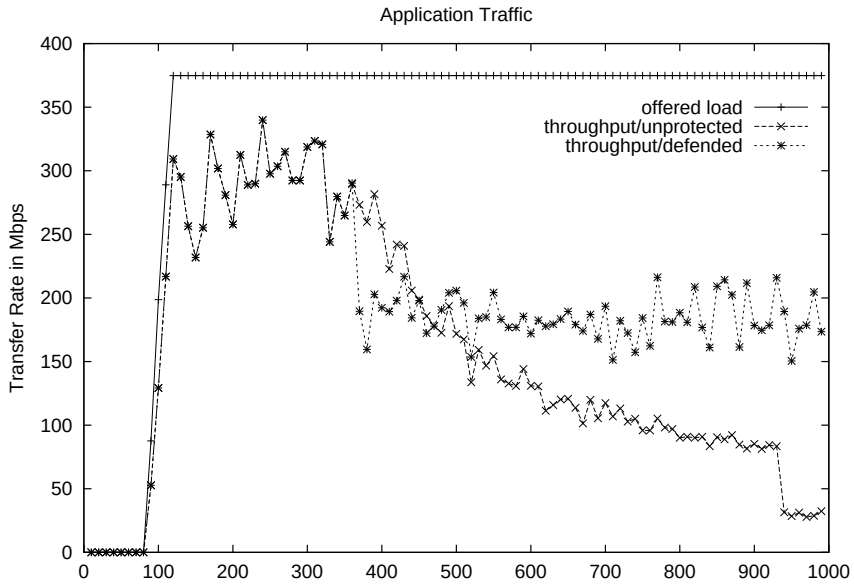

Fig. 2. Aggregate delivered bandwidth as a function of time, with and without quarantine defenses

Figure 2 illustrates the aggregate bandwidth consumed by all applications. The offered load is shown; the other two curves describe the behavior with and without defenses. The variation in application throughput is largely due to variation in non-interactive background traffic. These curves are identical until the time (around 380 seconds) when the detection mechanisms quarantine infected networks. For a short time after this the aggregate application throughput of the unprotected case is larger than the protected case, which dropped when quarantines were established. As the worm spreads it consumes most bandwidth at bottleneck points where LANs attach to the Internet, so that the bandwidth available to application traffic decreases. A significant drop occurs around time 930 seconds, when a router fails and isolates an important subnetwork.

Figure 3 shows the network performance from the viewpoint of an IP address that is destination for a large set of concurrent transfers. The IP address is one generated by our simulation—any resemblance to an actual network of that name is purely coincidental! The y-axis plots the fraction of those transfers that are live as a function of time (on-off cycling explains why we don't observe 100%

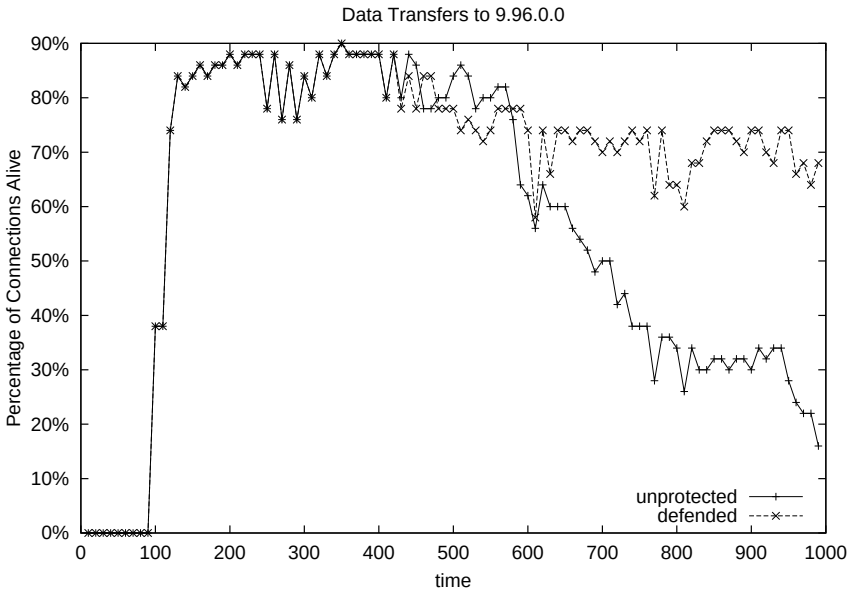

Fig. 3. Fraction of working transfers to 9.96.0.0 with and without quarantine defenses

activity prior to the worm spread at time 300 seconds.) The unprotected and defended curves track each other until infected subnetworks are quarantined, at which point the defended curve drops slightly but appears to stabilize while the unprotected curve declines, again showing a significant drop shortly after time 930, where we have seen already other effects of a router failure.

Figure 4 likewise focuses on performance from the perspective of a single device, in this case the source of a multicast video stream (again with an artificial IP address). Here we plot on the y-axis the fraction of packets lost among all of the streams it pushes. Once again we see that after quarantine the loss rate stabilizes, while in the system with no defenses the loss rate grows in tendency, with a significant jump after a router failure.

6 Conclusions

Multiscale modeling of network traffic and device behavior is essential if one is to capture the detailed effects that a large-scale Internet event such as a worm attack may have. This paper sketches our present approach in the context of the SSFNet simulation system. We illustrate the concepts using an example where one wishes to assess the effectiveness of worm detection and defense mechanisms. The network considered is very large, yet through aggressive modeling techniques the whole simulation model can be handled on a laptop class computer.

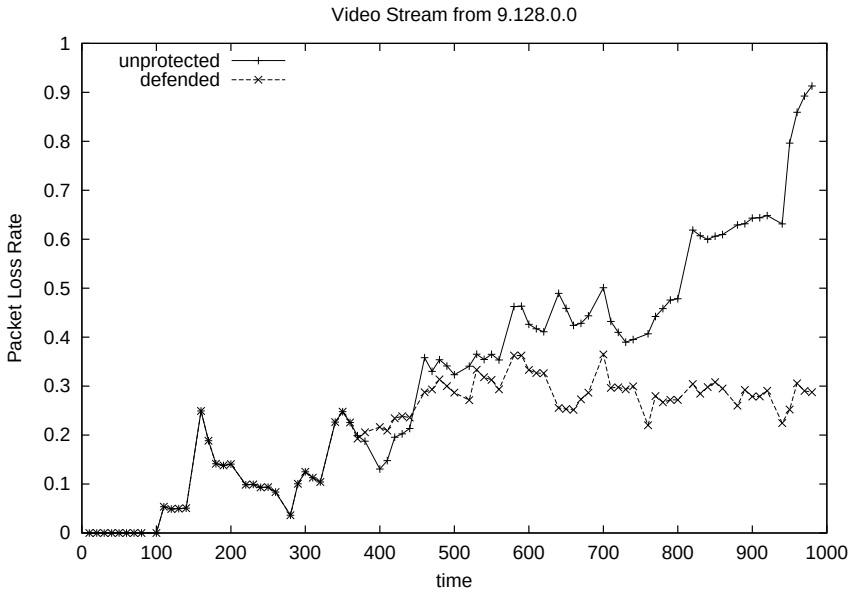

Fig. 4. Packet loss rate on streams from 9.128.0.0 with and without quarantine defenses

Acknowledgements

The authors thank BJ Premore for all the help he's given supporting our work with BGP, and for his development of the BGP implementation in SSFNet.

This research was supported in part by DARPA Contract N66001-96-C-8530, NSF Grant ANI-98 08964, NSF Grant EIA-98-02068, Dept. of Justice contract 2000-CX-K001, and Department of Energy contract DE-AC05-00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of t his contribution, or allow others to do so, for U.S. Government purposes.

References

1. CERT. Cert advisory ca-2001-19 code red worm exploiting buffer overflow in iis indexing service dll, July 2001. http://www.cert.org/advisories/CA-2001-19.html.
2. CERT. Cert advisory ca-2001-26 nimda worm, September 2001. http://www.cert.org/advisories/CA-2001-26.html.
3. CERT. Cert advisory ca-2003-04 ms-sql server worm, January 2003.
4. ISTS. Dib:s scan detection and correlation, 2003. http://www.ists.dartmouth.edu/IRIA/projects/dibs/.
5. M. Liljenstam, Y. Yuan, B. Premore, and D. Nicol. A mixed abstraction level simulation model of large-scale internet worm infestations. In *In Proceedings of the 2002 MASCOTS Conference*, July 2002.

6. Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to own the internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium (Security '02)*, 2002.
7. L. van Beijnum. *BGP*. O'Reilly, 2001.
8. `www.ssfnet.org`.