Worm Simulation with a Realistic Internet Model

Xueyang Xu, *GT ID: 903234112*
Xingyu Liu, *GT ID: 903233948*
Yunwei Qiang, *GT ID: 903231467*
Wenxin Fang, *GT ID: 903231847*
Nan Li, *GT ID: 903230813*

Electrical and Computer Engineering

Georgia Institute of Technology

***Abstract*—Internet worm spread is a phenomenon involving millions of hosts, who interact in complex and diverse environment. Worm spread is affected by many factors in realistic Internet environment, which makes it hard to analyze. In this page, we simulate three basic experiments about slammer using NS-3 where we take bandwidth, background traffic and patching into consideration. Through the result of simulations, we gain a deeper understanding of worms behavior.**

***Index Terms*—worm, simulation, realistic Internet, congestion, background traffic, MPI**

I. Introduction

INTERNET worm spread is a phenomenon involving millions of hosts, who interact in complex and diverse environment [1]. The Internet is a primary target for illegal activities, especially propagating malicious software programs. Worms, defined as a standalone malware computer program that replicates itself in order to spread to other computers, have been developed for more than 10 years since the first Morris worm [2]. Worms almost always cause at least some harm to the network, even if only by consuming bandwidth, whereas viruses almost always corrupt or modify files on a targeted computer. Although the computer science technology grows fast throughout the past decades, the slammer worm with a benign payload a surprising distributive capacity spread with so quickly that human response was almost ineffective. In order to defend against future worms, we need to fully understand various properties of worms: the impact of patching; packet traffic congestion conditions within the Internet networks (also known as the impact of background traffic); network topology and Internet bandwidth of each compromised machine.

Given the potential harms such as DDoS, flash crowds, spam caused by worm spread, a packet-level simulation of relevant event features, and a realistic model of background traffic on the whole Internet are primary needs for scientific researchers and industrial engineers. we have developed a worm model using NS-3 that can be used to study the behavior of these Internet worms under a variety of conditions. Our model support Slammer Worm and UDP style infection methods. Our simulation models include complete packet-level details, both for the worm and the competing(background) traffic on the modeled subnets. By using this level of details, we capture behaviors such as the effect of background traffic on the infection packets as well as the worm traffic itself. This level of details helps get a whole understanding of how the worms behave when there exist other data flows and defensive patching mechanisms.

The remainder of the paper is organized as follows: Section II gives the background knowledge for proposed worm simulation model, both in packet-level simulations and with analytical models. Section III provides the detailed description and algorithm on our model including the Internet topology, simulation strategy and implementation details using NS-3. Section IV displays the preliminary results from our simulation-based experiments of worm spreading. Section V gives some conclusions and future directions for this topic.

II. Background

Slammer is a 2003 computer worm that caused a denial of service on some Internet hosts and dramatically slowed down general Internet traffic. It spread rapidly, infecting most of its 75,000 victims within ten minutes. While Slammer had no malicious payload, it caused considerable harm by overloading networks and disabling database servers. The worms spreading strategy uses random scanning which select IP address randomly and finally selecting and infecting all the nodes in the Internet. Slammer is bandwidth-limited, making it easy to spread as fast as possible compared with the well-known latency-limited Code-Red worm [2].

There are a various of worm models proposed before: Worm models [1], Single-machine simulation with AS-level topology [3] and PAWS model [4]. They each focus on some specific features of the behavior of Internet worm but are not perfect designed to display the real-world Internet traffic conditions.

III. Simulation Topology and Strategy

A. Topology

In order to represent relatively realistic Internet model, we implemented the topology based on the double layer star

format as in project 3. However, we cannot achieve such massive scale of simulation nodes as in the original paper because of the limitation of time and performance of our computers. So, we built double layer star formation of 1 hub, 8 inner nodes and 16 child nodes per each inner node. Then, by connecting 16 of such formation using a single bus, we obtained a network topology of 2192 nodes. Among these nodes, there exist 2048 child nodes that will be used as vulnerable nodes to spread worms. The final topology should be like Fig. 1.

We would have heavy background traffic during the spreading of worm in actuality. It is quite intuitive that the worm is likely to spread slower in an environment with high background network traffic.

The background traffic in the whole network is quite randomly distributed among every nodes that participate in the communication. To make the configuration easier to understand, we set that all child nodes are almost equivalent in generate the background traffic.

Each node will send and receive packets from two other nodes. Here we use the notation (i, j, k) to represent the k^{th} child node in the j^{th} inner star-topology that connect to the i^{th} hub. Under the current configuration, the node(i, j, k) will send packets to the node(i+1, j, k) and receive packets from the node(i-1, j, k) in a round robin pattern. The parameter we used in our program is applied to all connections that carry background traffic but we currently set separated speed for these connections.

To implement the inter-nodes background traffic, we use the build-in on-off helper of NS-3 to create sending and receiving applications then assign them to each nodes. The on-off helper is used here to provide the background traffic with some controllable randomness, which is, although we do not know the exact behavior of each application, we have the overall proportion of time that each node is contributing to the traffic in our backbone network.

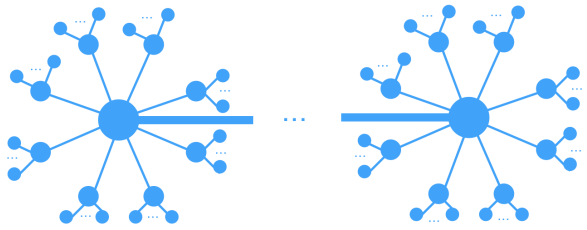

Fig. 1: Topology of simulation network

B. Worm

To obtain a similar spreading pattern of worms such as Slammer, we need implement worm application that can simulate the different stages. It should be able to listen, spread, infect and might be patched at some given simulation time.

While setting up, we installed worm application into every childnodes. This does not means all child nodes are infected from beginning, but only a easier way to keep track of the spreading of worm. Then, before simulation starts, we manually pick a node (e.g. child node 0 from hub 0) as patient zero and start infecting others. In our worm application, each worm have two sockets. One socket is the listening socket. It check whether current node receive a package and check if it is a worm package. If received package is a worm package, then this node is marked as infected and move into propagation stage. Another socket is the spreading socket. Once a node is infected, this socket start working and begin sending out worm package to random selected node. More specifically, for Slammer worm, we used uniformly distribution to generate IP address which could represent any node of 2048 child nodes.

In actual network topology, the sub-topology for each hub formation should not be symmetric as we implemented. So, we added a parameter called vulnerability. By modifying this parameter, we can set the possibility for each child node to be immune to worm. The reason we did this is we need simulate the situations that some guessed IP should be empty but the worm package should still be sent out until it is dropped at some node on the way. However, in NS-3 if a package is sent to a non-exist node, it will be dropped at the beginning. In this way, we cannot observe its impact on the whole topology. Considering the massive amount of worm data sending in the network, this problem cannot be ignored. By applying vulnerability, we can simulate the situation that the corresponding IP address represents an empty node.

Patching is a good defense to fight slow-propagation worms. After patching, a host would be immune to future infection and stop sending worm packages if it was infected. To simulate this, we assign a probability for each host to define if it will be patched or not. If so, we decide when this host will be patched. Since worm is spreading and have more and more impact on the whole network. People will start patching hosts in order to control the spreading. Thus, patching should start at some point when people noticed and the possibility should be increasing while worm is spreading. We applied a distribution similar to Gamma distribution to simulate such processing. After patching, the patched host will be marked as invulnerable. If it has been infected before, the worm application on it will be disabled and the total number of infected node will decrease.

C. MPI

In this simulation, we create a topology with 2048 nodes. This is a relatively large scale for a single machine as we try to run our program in a laptop with 4-core Intel i7 processor.

So that utilizing the MPI module of the NS-3 can help us to improve the overall performance of the simulation.

Like what we had done in the previous project, we equivalently split the topology then assign each piece to the corresponding core. However, the speedup cannot be as large as the number of cores we have since there are quite a lot data exchanging and communication between cores.

IV. Simulation Topology and Strategy

V. Simulation Result

After setting up topology, we are ready to simulate worm in a realistic Internet environment. We run three basic experiments and their results are shown as below.

A. Simulation with different degree of congestion

To simulate with different degree of network congestion, we set bandwidth to 50MB and change background traffic data rate from 1B to 50KB. The higher background traffic data rate is, the more serious network congestion would happen which will affect worm propagation.

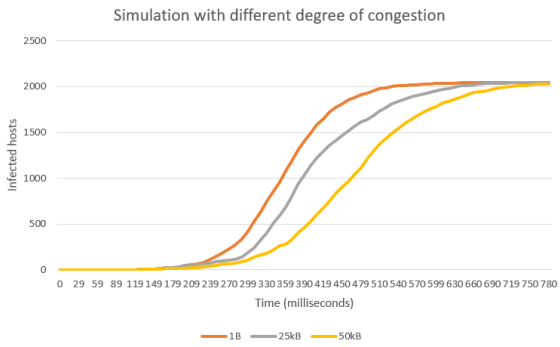

Fig. 2: Simulation with different network congestion

Fig. 2 shows worm simulation result with different degree of network congestion. We note here that the propagation can be slowed down more when traffic background data rate is going up and thus increasing congestion effects.

B. Simulation with different bandwidth

To simulate with different bandwidth, we set background traffic to 1KB and change bandwidth between hubs and hub to child from 50MB to 1GB.

Fig. 3 shows worm simulation result with different bandwidth. We can find that with higher bandwidth, worm propagation would be faster. But finally they will all reach same amount of infected hosts.

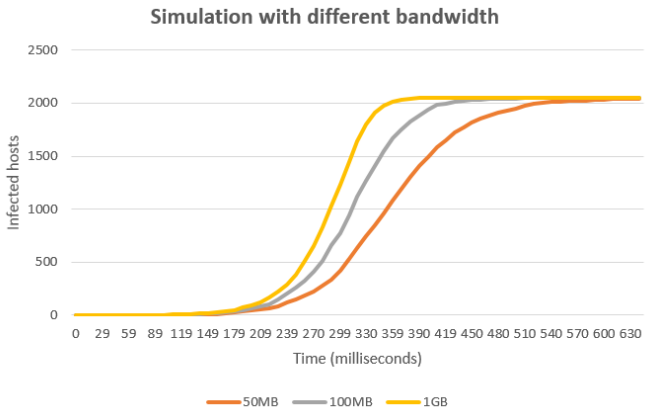

Fig. 3: Simulation with different bandwidth

C. Simulation with patching

We simulate worms infection under Internet environment with patching and without parching respectively. To simulate patching, some time after starting simulation, we set each node in the network to invulnerable at random time. Therefore, the total number of nodes infected may decrement at the setting starting time.

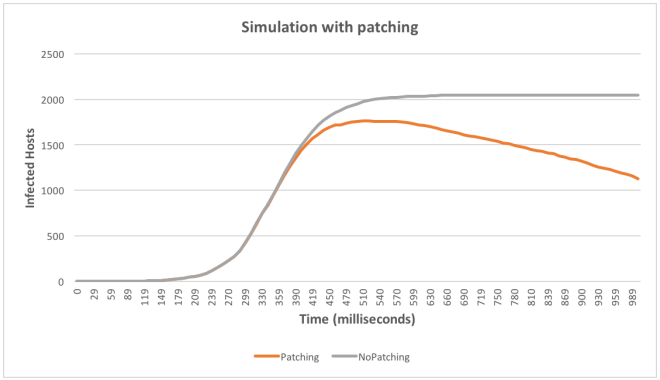

Fig. 4: Simulation with patching

Fig. 4 shows that result of worm simulation with and without patching. As the picture shows, the number of infected nodes in two strategy remains the same before roughly 390ms, where patching takes into effect. After that, the number of infected nodes decreases gradually with patching while no-patching does not.

VI. Conclusion

In summary, we implement three simulations of worm infecting process under realistic Internet environment on our laptop. Changing the parameters such as bandwidth, background data rate and patching enables us to investigate the congestion and patching effects of the worm spread. To speed up our simulation, we also adapt MPI to run simulations parallel. Compared with the result in [5], our result is similar and provides an running example to worm

researchers.

For future work, we will integrate worm vulnerability and more complex network topology into into existing simulation. Besides, we will compare universal patching and partial patching.

REFERENCES

[1] Z. Chen, "Modeling and defending against internet worm attacks," Ph.D. dissertation, Georgia Institute of Technology, 2007.

[2] D. Moore, C. Shannon, *et al.*, "Code-red: a case study on the spread and victims of an internet worm," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. ACM, 2002, pp. 273–284.

[3] G. M. V. D. Moore, C. Shannon and S. Savage, "Internet quarantine: Requirements for containing self- propagating code," *INFOCOM*, 2003.

[4] S. Floyd and V. Paxson, "Difficulties in simulating the internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 392–403, 2001.

[5] S. Wei, J. Mirkovic, and M. Swany, "Distributed worm simulation with a realistic internet model," in *Principles of Advanced and Distributed Simulation, 2005. PADS 2005. Workshop on*. IEEE, 2005, pp. 71–79.