UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

MODELING, ANALYSIS AND DEFENSE STRATEGIES AGAINST INTERNET ATTACKS

A Dissertation

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

JONGHYUN KIM

Norman, Oklahoma

2005

MODELING, ANALYSIS AND DEFENSE STRATEGIES AGAINST INTERNET ATTACKS

A DISSERTATION APPROVED FOR THE

SCHOOL OF COMPUTER SCIENCE

BY

_____
Sridhar Radhakrishnan, Chair

_____
Sudarshan K. Dhall, Co-Chair

_____
S. Lakshmivarahan

_____
Krishnaiya Thulasiraman

_____
M. J. Breen

Acknowledgements

I would like to acknowledge many people for helping me during my doctoral work. I would especially like to express my deepest thanks and appreciation to my advisor, Professor Sridhar Radhakrishnan and my co-advisor, Professor Sudarshan K. Dhall, for their guidance, encouragement, and support during my research. Throughout my doctoral work they encouraged me to develop analytical thinking and research skills.

I am also very grateful for having an exceptional doctoral committee and wish to thank Prof. Krishnaiya Thulasiraman, Prof. S. Lakshmivarahan, and Prof. M. J. Breen for all their help and encouragement first in their classes and then in producing this dissertation.

I wish to express my sincere gratitude to my father and my mother and their families for their sacrifice and encouragement and for supporting me in every aspect of my graduate study.

There is one person who deserves my deepest thanks and appreciation for her continued support during the writing of this dissertation: my wife, Seunghyun Hong. I could not have done it without her.

Finally, I would like to thank my colleagues Shankar Banik, Aravind B., and Tao Zheng for sharing their experience and knowledge. I dedicate my dissertation to all whom I love.

Contents

List of Tables

List of Figures

Abstract

Many early Internet protocols were designed without a fundamentally secure infrastructure and hence vulnerable to attacks such as denial of service (DoS) attacks and worms. DoS attacks attempt to consume the resources of a remote host or network, thereby denying or degrading service to legitimate users. *Network forensics* is an emerging area wherein the source or the cause of the attacker is determined using IDS tools. The problem of finding the source(s) of attack(s) is called the "*trace back problem*". Lately, Internet worms have become a major problem for the security of computer networks, causing considerable amount of resources and time to be spent recovering from the disruption of systems. In addition to breaking down victims, these worms create large amounts of unnecessary network data traffic that results in network congestion, thereby affecting the entire network.

In this dissertation, first we solve the trace back problem more efficiently in terms of the number of routers needed to complete the track back. We provide an efficient algorithm to decompose a network into connected components and construct a terminal network. We show that for a terminal network with n routers, the trace back can be completed in $O(\log n)$ steps.

Second, we apply two classical epidemic SIS and SIR models to study the spread of Internet Worm. The analytical models that we provide are useful in determining the rate of spread and time required to infect a majority of the nodes in the network. Our simulation results on large Internet like topologies show that in a fairly small amount of time, 80% of the network nodes is infected.

Third, we have analyzed the tradeoff between delay caused by filtering of worms at routers, and the delay due to worms' excessive amount of network traffic. We have used the optimal control problem, to determine the appropriate tradeoffs between these two delays for a given rate of a worm spreading. Using our technique we can minimize the overall network delay by finding the number of routers that should perform filtering and the time at which they should start the filtering process.

Chapter 1

Introduction

1.1 Internet Attacks

The basis for the Internet was an experiment begun in 1968 by the Defense Department's Information Processing Techniques Office (ARPA/IPTO) to connect computers over a network in order to ensure command and control communications in the event of a nuclear war. In the 1980s, the number of local area networks increased significantly and this stimulated rapid growth of interconnections to the ARPAnet and other networks. These networks and interconnections are known today as the Internet [1].

Many early Internet protocols were designed without a fundamentally secure infrastructure so that network defense becomes more difficult. Because of the openness

of the Internet and the original design of the protocols, Internet attacks in general are quick, easy, inexpensive, and may be hard to detect or trace. An attacker does not have to be physically present to carry out the Internet attack. In fact, many attacks can be launched readily from anywhere in the world - the location of the attacker can easily be hidden.

Since much of the traffic on the Internet is not encrypted, confidentiality and integrity are difficult to achieve. The factor that contributes to the vulnerability of the Internet is the rapid growth and use of the network, accompanied by rapid deployment of network services. Often, these services are not designed, configured, or maintained securely. This lack of secure configuration makes them vulnerable to attacks, which sometimes occur within minutes of connection. Finally, the more systems that are connected to Internet, obviously the harder it is to control their security. Clearly, if a site is connected to the Internet at several points, it likely would be more vulnerable to attacks than a site with a single gateway.

1.1.1 Denial of service (DoS) attacks

On the Internet, a denial of service (DoS) attack attempts to consume the resources of a remote host or network, thereby denying or degrading service to legitimate users. In other words, a denial of service attack prevents the targeted site from providing network services by either flooding the site with bogus packets or consuming limited network resources. Furthermore, a denial of service attack might use multiple systems to attack one or more victim systems with the intent of denying service to legitimate

users of the victim systems. Typically, the loss of service is the inability of a particular network service such as e-mail service, or the temporary loss of all network connectivity and services. A denial of service attack can also destroy programming and files in a computer system. The major advantage of a DoS attack is that it is quite difficult to determine the actual source of the attack. Since the attacker can basically put any packet on the local wire, the attacker creates packets whose source IP address is invalid and completely random. Thus, when the victims receive these packets, they are unable to determine the source.

The most common kind of a DoS attack is simply to send more traffic to the network than it can handle, called *packet flooding*. Then the network's connection becomes congested, resulting in packet loss. Since routers cannot distinguish between attacking packets and valid client packets, they drop them with equal probability. If the attacker can send packets fast enough, the drop rate can become so high that a number of client's packets cannot get through. A more recent and well-known attack called "*smurf*" attack [2] use reflectors to multiply the effect of the DoS attack. In this type of attack an attacker is using ICMP echo request packets directed to IP broadcast addresses from remote locations to generate denial-of-service attacks.

1.1.2 Internet Worms

Lately, Internet worms have become a major problem for the security of computer networks, causing considerable amount of resources and time to be spent recovering from virulent attacks. In general, worms, defined as self-propagating malicious codes,

have been developed since the Morris worm arose in 1988 [7]. Unlike a virus, which requires a user to do something to continue the propagation, a worm can propagate by itself. The convenience of Internet makes it more vulnerable for malicious Internet exploits. In other words, the Internet has become a powerful means for propagating malicious programs like computer viruses and worms. The Code Red worm incidents of 2001 have shown us how vulnerable Internet hosts are and how fast a virulent worm can spread across the Internet (Code Red infected more than 250,000 systems in just 9 hours on July 19, 2001). Moore [4] provided some characteristics of the worm spread and trace analyses of Code Red worm behavior. Weaver [14, 15] introduced worm design strategies, which can be used to produce significantly faster and longer lived Internet worms.

A worm, on the other hand, is far more powerful and faster. The Sapphire/Slammer Worm was the fastest Internet worm in history. As it began spreading throughout the Internet, it infected at least 75,000 vulnerable hosts within 10 minutes [3]. When a worm gains access to a computer (usually by breaking into it over the Internet), it launches a program which searches for other Internet locations, infecting them if it can. Moreover, the worm travels over the Internet, so all machines attached to an infected machine are at risk of attack. Some worms attempt to perform a Denial of Service attack (Code Red/W32.Blaster) or to compromise systems and deface web site (sadmind/IIS, Code Red); and others have dynamic configuration capabilities (W32.Leaves) [3]. But the biggest impact of these worms is that their propagation effectively creates a denial of service in many parts of the Internet because of the huge amounts of scan traffic generated, and they cause much substantial damage.

1.2 Computer Worms

Computer worms and viruses are typically grouped together as infectious agents that replicate themselves and spread from system to system. However, Computer worms must be differentiated from computer viruses if we are to understand how they operate, spread, and can be defended against. Computer worms alter the behavior of the computer they infect. Computer worms typically install themselves onto the infected system and begin execution, utilizing the system's resources, including its network connection and storage capabilities.

1.2.1 Worms vs. Viruses

Both worms and viruses spread from a computer to other computers. However, viruses typically spread by attaching themselves to files (either data files or executable applications). Their spread requires the transmission of the infected file from one system to another. Worms, in contrast, are capable of autonomous migration from system to system via network without the assistance of external software. In other words, a worm is an active and volatile automated delivery system that controls the medium (typically network) used to reach a specific target system. Viruses, in contrast, are a static medium that does not control the distribution medium.

From the Morris worm [7] in 1998, a computer worm was defined as follows:

> "In computers, a worm is a program that travels from one computer to another but does not attach itself to the operating system of the computer it infects. It

differs from a virus which is also a migrating program, but one that attaches itself to the operating system of any computer it enters and can infect any other computer that uses files from the infected computer."

Currently Many worms hide their presence by installing software to deliberately hide their presence, some use kernel modules to accomplish this. Such an instance of a worm would not be covered by the above definition.

1.2.2 Worm history and Taxonomy

The concept of a worm program that spreads itself from machine to machine was apparently first described by John Brunner in 1975 in his book *The Shockwave Rider*. He called these programs *tapeworms* that lived "inside" the computers and spread themselves to other machines. In 1979-1981, researchers at Xerox PARC built and experimented with *worm* programs [3]. The worms built at PARC were designed to travel from machine to machine and do useful work in a distributed environment. They were not used at that time to break into systems, researchers soon developed worms that could harness under utilized computing resources. Furthermore, the possibility of a malicious worm such as the Morris worm became after an accident with the worm at Xerox PARC. Table 1.1 shows a generalized lineage of many of the worms which have focused on windows hosts.

Worm	Discovery Date	Distinction
Morris/Internet	Nov. 1988	The first significant worm. Exploited multiple vulnerabilities
mIRC Script.ini	Dec. 1997	Attacks users of the IRC client mRC.
Melissa	Mar. 1999	It shut down Internet mail systems. It spread on word processor
Love Letter	May 2000	A VBScript worm that spread largely via e-mail as a chain letter.
Leaves	Jun. 2001	Using the installed backdoor program to upload itself.
Code Red	Jul. 2001	The self-replicating malicious code that exploits a known vulnerability in Microsoft IIS servers.
Code Red II	Aug. 2001	It causes system level compromise and leaves a backdoor on certain machines running Windows 2000.
Nimda	Sept. 2001	A hybrid windows worm – attacked client-to-client, server-to-server, client-to-server, and ser-to-client.
SQL Snake	May 2002	Internet worm targeting Microsoft SQL servers with TCP port 1433.
Sapphire/Slammer	Jan. 2003	Using a single UDP packet for explosive growth
W32/Blaster	Aug. 2003	It exploits a vulnerability in Microsoft's DCOM RPC interface using TCP port 135

Table 1.1: Traditional worms of Note [3]

1.2.3 A Worm Spreading

Now we describe how a worm spreads on Internet and attacks many systems. We explain the worm spreading techniques with one of malicious worms such as Sapphire/Slammer worm. The Sapphire/Slammer worm (also called *Slammer*) was the fastest computer worm in history [3]. As it began spreading throughout the Internet on January 25, 2003, it doubled in size every 8.5 seconds. It infected more than 90 percent of vulnerable hosts within 10 minutes. Slammer exploited buffer overflow vulnerability in computers on the Internet running Microsoft's SQL Server. This weakness in an underlying indexing service was discovered in July 2002;Microsoft released a patch for the vulnerability before it was announced. The worm infected at least 75,000 hosts, and caused network outages and significant disruption of financial, transportation, and government institutions.

Propagation speed of Slammer worm was very fast: The worm achieved its full scanning rate (over 55 million scans per second) after approximately three minutes, after which the rate of growth slowed down because significant portions of the network did not have enough bandwidth to allow it to operate. Most vulnerable machines were infected within 10 minutes of the worm's release. By comparison, it was faster than the Code Red worm, which infected over 359,000 hosts on July 19th, 2001 [4]. While Slammer did not contain a malicious payload, it caused considerable harm simply by overloading networks and taking database servers out of operation. Many individual sites lost connectivity as their access bandwidth was saturated by local copies of the worm and there were several reports of Internet backbone disruption. In other words, if

the worm had carried a malicious payload, it could have attacked a more widespread vulnerability and the effects would likely have been more severe.

Slammer's spreading strategy is based on *random scanning* - it selects IP addresses at random to infect, eventually finding all susceptible hosts. Random scanning worms initially spread exponentially rapidly, but the rapid infection of new hosts becomes less effective as the worm spends more effort retrying addresses that are either already infected or immune.

Slammer spread nearly two orders of magnitude faster than Code Red, yet it probably infected fewer machines. Both worms used the same basic strategy of scanning to find vulnerable machines and then transferring the exploitive payload; they differed in their scanning constraints. While Code Red was *latency limited*, Slammer was *bandwidth-limited*. Slammer contains a simple, fast scanner in a small worm with a total size of only 376 bytes. This can be contrasted with the 4kb size of Code Red, or the 60kb size of Nimda. Previous scanning worms, such as Code Red, spread via many threads, each invoking *connect*() to probe random addresses. Thus each thread's scanning rate was limited by network latency, the time required to transmit a TCP-SYN packet and wait for a response or timeout. In contrast, Slammer's scanner was limited by each compromised machine's bandwidth to the Internet. Since the SQL Server vulnerability was exploitable using a single packet to UDP port 1434, the worm was able to send these scans without requiring a response from the potential victim. Slammer was frequently limited by the access bandwidth to the Internet rather than its own ability to generate new copies of itself. The Slammer worm's scanning technique

was so aggressive that it quickly interfered with its own growth. Consequently, the rate of growth from later infections was reduced since these instances were forced to compete with existing infections for scarce bandwidth. Thus Slammer worm achieved its maximum Internet-wide scanning rate within minutes.

The following is the procedure of what the worm's payload is doing after infection:

1. Retrieves the address of *GetProcAddress* and *Loadlibrary* from the IAT in sqlsort.dll.
2. Calls *gettickcount*, and uses returned count as a pseudo-random seed
3. Creates a UDP socket
4. Performs a simple pseudo random number generation using the returned *gettickcount* value to generate an IP Address that will later be used as the target.
5. Send worm payload in a SQL Server Resolution Service request to the pseudo random target address, on port 1434 (UDP).
6. Return back and continue generating new pseudo random addresses.

In general, the response to Slammer was quick. Within an hour, many sites began filtering all UDP packets with a destination port of 1434. Slammer represents the idealized situation for network-based filtering: the worm was easily distinguished by a signature that is readily filterable on current hardware and it attacked a port that is not generally used for critical Internet communication. Thus almost all traffic blocked by these filters represents worm-scanning traffic. If the worm had

exploited vulnerability in a commonly used service (e.g. DNS at UDP port 53 or HTTP at TCP port 80), such filtering could have caused significant disruption to legitimate traffic with resulting denial-of-service more harmful than the worm itself.

1.2.4 Worm Detection

In this section we attempt to illustrate one of the methods of detecting worms using signature-based detection which is called *pattern matching*. We are interested in network payload signature that deals with packet headers and packet payloads, as is used in network intrusion detection systems (NIDS) [26]. The detection method used by NIDS engines perform an evaluation of packet contents received from the network. This can include matching signatures based on payload contents measured by string comparison, application protocol analysis, or network characteristics.

Signature-based detection

Signature-based detection is the method of analyzing the content of captured data to detect the present of known strings. These signatures are kept in a database and are derived from the content of known malicious files. These files are typically the executable programs associated with worms.

The strength of signature-based detection is that the behavior of one instance of malicious worm is representative of all instances. This means that by detecting one node of the worm, the behavior of all nodes that are compromised by the worm can be reliably predicted. However, this signature-based detection also has several weaknesses.

One of drawback is that they rarely can be used to detect a new worm. Only after an attack is known, it can be used to detect a worm. Another of drawback is that it is hard to keep up with variants of worms and viruses.

Worm Signature

Worms typically have distinctive signatures as they attack other hosts on the network. By building up a library of known malicious signatures, a network monitor can alert an administrator to the presence and activity of a worm.

In case of the Code Red worm, a distinctive request is made to the target server that contained the exploit as well as the malicious executable. By examining packets observed passively on the network, a detection system can identify Code Red worm activity. The largest problem with this signature for Code Red is its size. This signature is more than 100 bytes in length and must be fully matched against to successfully detect the worm's traffic. If this payload is fragmented due to network transmission sizes, the larger signature will not match the smaller payloads in the fragments.

There are numerous ways to monitor our network and protect it from Internet worms. For instance, companies commonly use a firewall for network protection. Although firewall logs often provide a lot of information regarding intrusion attempts, sometimes they contain too much data to solve the problem quickly. Some companies also use intrusion detection systems (IDSs) on border routers to monitor incoming traffic for patterns that indicate specific intrusion attempts. Worms that infect internal systems behind a firewall may be difficult to isolate since firewalls and intrusion

detection systems are used primarily on borders with the Internet, rather than on internal networks.

1.3 Epidemiological Models

Epidemiological models have traditionally been used to understand and model the spread of biological infectious diseases [9, 10]. Furthermore, in the area of virus and worm modeling, many studies have employed simple epidemiological models to understand general characteristics of worm's propagation [5, 8]. In this section we introduce two classical epidemiological models.

1.3.1 SIS model

Let $S(t)$ be the number of susceptible individuals at time t, and let $I(t)$ be the number of infected individuals. For an SIS model, infected individuals return to the susceptible class on recovery because the disease confers no immunity against re-infection. The classical SIS model is given by

$$\frac{dS}{dt} = -\beta SI + \delta I \tag{1.1}$$

$$\frac{dI}{dt} = \beta SI - \delta I$$

Let's briefly explore the meaning of these terms.

- The βSI term is understood as follows: An average infected individual makes contact sufficient to infect βN others per unit time. Also, the probability that a given individual that each infected individual comes in contact with is susceptible is S/N. Thus, each infected individual causes $(\beta N)(S/N) = \beta S$ infections per unit time. Therefore, infected individuals, I, cause a total number of infections per unit time of βSI.
- The δI term is even simpler to understand: δ is the fraction of infected individuals who recover (and re-enter the susceptible class) per unit time.

We see that

$$\frac{d}{dt}(S + I) = 0 \tag{1.2}$$

Therefore,

$$S + I = N \text{ is constant.}$$

1.3.2 SIR model

The SIR model has been proposed by Kermack and McKendrick who considered the removal process of infected individuals [8]. We divide the population into three classes S, I, and R. The SIR model is very similar to the SIS model except that recovered individuals return to class R instead of passing to class S through immunization against infection. $R(t)$ denotes the number of individuals who have been infected and then removed from the possibility of being infected again or of spreading infection.

The classical SIS model is given by

$$\frac{dS}{dt} = -\beta SI$$

$$\frac{dI}{dt} = \beta SI - \alpha I \tag{1.3}$$

$$\frac{dR}{dt} = \alpha I$$

where β is the infection rate; α is the rate of removal.

We note that $N = S + I + R$.

1.4 Optimal control problem

Optimal control can be regarded as one of the possible methodologies of the control system's design. The most general optimal control problem is described by four types of data: (1) system constraints, (2) the initial state and the target state, (3) the class of admissible controllers, and (4) the cost functional. We attempt to investigate such an optimal control problem of minimizing the cost function described in chapter 4. The objective of our optimal control problem is to determine the control variables that will cause a system to satisfy the constraints and at the same time minimize the total cost of infection.

1.4.1 Control System Model

We consider a control problem where based on a system model we have to determine the control inputs $u(t)$ such that the system behavior $x(t)$ meets our requirements as shown in figure 1.1.

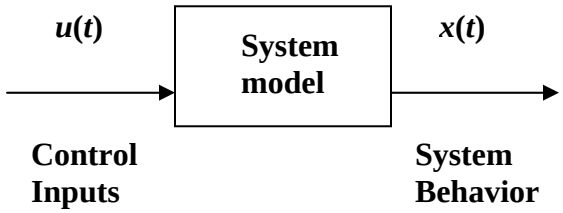


Figure 1.1: Control and behavior related by a system model

In case of optimal control we have a mathematical system model,

$$\dot{x} = f(x,u,t), \quad x \in R^n, u \in R^m \tag{1.4}$$

A formal statement of the control problem is comprised of the state variable, the control variable, time, a set of differential equations, the determination of terminal time, and the objective function.

Time, t, is measured in continuous units and is defined over the relevant interval from initial time t_0, which is typically given, to terminal time t_1, which must often be determined. Thus the relevant interval is: $t_0 \le t \le t_1$

At any time t in the relevant interval the state of the system is characterized by n real numbers, $x_1(t), x_2(t),..., x_n(t)$, called *state variable*, and summarized by the state vector:

$$\mathbf{x}(t) = (x_1(t), x_2(t),..., x_n(t))', \tag{1.5}$$

is a continuous vector valued function of time, the value of which at any time t in the relevant interval is the state vector. The initial state, $\mathbf{x}(t_0) = \mathbf{x}_0$, is assumed given, and the terminal state, $\mathbf{x}(t_1) = \mathbf{x}_1$, must often be determined.

At any time t in the relevant interval the controls to be made are characterized by r real numbers, $u_1(t), u_2(t),..., u_r(t)$, called *control variables* and summarized by the control vector:

$$\mathbf{u}(t) = (u_1(t), u_2(t),..., u_r(t))', \tag{1.6}$$

is a continuous vector valued function of time, the value of which at any time t in the relevant interval is the control vector.

The state trajectory $\{\mathbf{x}(t)\}$ is characterized by a set of n differential equations giving the time rate of change of each state variable as a function of the state variables, the control variables, and time:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \tag{1.7}$$

Or, written out in full:

$$\frac{dx_j}{dt}(t) = \dot{x}_j(t) = f_j(x_1(t), x_2(t), \ldots x_n(t); u_1(t), u_2(t), \ldots u_r(t) : t), \; j = 1, 2, \ldots, n, \quad (1.8)$$

where each of the n functions $f_1(\cdots), f_2(\cdots), \ldots, f_n(\cdots)$ is assumed given and continuously differentiable. If the differential equations do not depend explicitly on the time then the equations are autonomous.

The behavior of system is fully determined by $\mathbf{x}(t)$. Based on these state and control variables, an optimal control $\mathbf{u}^*(t)$ can be computed which minimizes the cost function $\boldsymbol{C}(\mathbf{u}(t))$. So optimal relates to the system model and cost function. Associated to the optimal control $\mathbf{u}^*(t)$ is the associated optimal system's behavior $\mathbf{x}^*(t)$ as shown in figure 1.2.

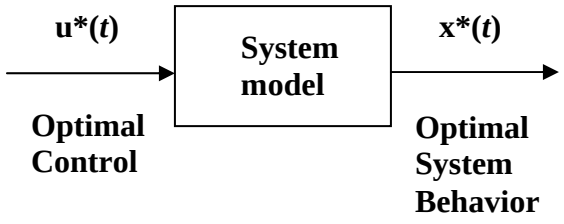


Figure 1.2: Optimal control based on a system model

Now we define the general form of optimal control problem as follows;

Given the system,

$$\dot{x}_i = f_i(x_1,...,x_n,u_1,...,u_m,t), \qquad i = 1, \ldots, n,$$

Or in vector form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t),$$

Where

$$\mathbf{x} = (x_1,...,x_n)' \qquad \text{and} \qquad \mathbf{u} = (u_1,...,u_m)'$$

with the known initial condition

$$\mathbf{x}(t_0) = \mathbf{x}_0$$

and the final condition that we wish to reach is $\mathbf{x}_1 \in R^n$. $\mathbf{x}_1$ is often called the *target point*, and may or may not be given.

Find the optimal control $\mathbf{u}^*(t)$ that minimizes the cost function

$$\boldsymbol{C}(\mathbf{u}) = \int_0^{t_1} f_0\,(\mathbf{x}(t), \mathbf{u}(t), t)dt, \tag{1.9}$$

where f_0 is a given continuous real-valued function, $\boldsymbol{C}(\mathbf{u}^*) \leq \boldsymbol{C}(\mathbf{u})$ for all $\mathbf{u}$.

1.4.2 Calculus of Variations

Calculus variations are suitable for solving linear or nonlinear optimal control problems with linear or nonlinear boundary conditions [56, 57]. Basically, it is a

collection of many different analytical methods and they are discussed differently from book to book. Here, a typical approach which leads to more general and widely used modern theories is introduced.

Pontryagin's maximum principle

Pontryagin's maximum principle is one of approaches to solve the optimal control problem. Pontryagin's maximum principle serves to identify on optimal path or trajectory. If we define $x(t)$ to represent the state of system at time t and $u(t)$ represents the control at time t, then the optimal control problem is to find trajectory $\{x(t)\}$ by choosing a set $\{u(t)\}$ of controls so as to maximize or minimize some objective function. The maximum principle therefore has been the basic approach to computing optimal controls in many important problems in mathematics, engineering, and economics.

The general formula of the maximum principle problem is:

$$\max_{\{u(t)\}} J = \int_{t_0}^{t_1} F(\mathbf{x}, \mathbf{u}, t)dt \tag{1.10}$$

s.t. $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$

$\mathbf{x}(t_0) = \mathbf{x}_0$ for $t = 0$ is initial point

$\mathbf{x}(t_1) = \mathbf{x}_1$ for $t = T$ is the final state

$\{\mathbf{u}(t)\}$ the control trajectory $t_0 \le t \le t_1$

Ω a set of all admissible controls

where $F(\cdots)$ and $\mathbf{f}(\cdots)$ are given continuously differentiable functions; and $\{\mathbf{u}(t)\}$ must belong to the given control set Ω. The maximum principle can be considered the extension of the method of Lagrange multipliers to optimal control problems. We introduce new variables, called *costate variables*, are the dynamic equivalents of the Lagrange multipliers of maximization problems:

$$\boldsymbol{\varphi}(t) = (\varphi_1(t),\ \varphi_2(t),\ \ldots,\ \varphi_n(t)) \tag{1.11}$$

It also notes that each of the *costate variables* corresponds to one of the differential equations of motion and in general varies over time.

The next step is to define a Lagrangian function which equals the expression to be maximized plus the inner product of the Lagrange multiplier vector and the constraints. The inner product is properly treated under the integral sign, the Lagrangian expression being:

$$L = J + \int_{t_0}^{t_1} \varphi[\mathbf{f}(\mathbf{x},\mathbf{u},t) - \dot{\mathbf{x}}]dt \tag{1.12}$$

$$= \int_{t_0}^{t_1} \{F(\mathbf{x},\mathbf{u},t) + \varphi[\mathbf{f}(\mathbf{x},\mathbf{u},t) - \dot{\mathbf{x}}]\}dt$$

To develop the necessary conditions, note that the term-$\varphi(t)\dot{\mathbf{x}}(t)$ in equation (1.12) can be integrated by parts to yield:

$$L = \int_{t_0}^{t_1} \{F(\mathbf{x},\mathbf{u},t) + \varphi \mathbf{f}(\mathbf{x},\mathbf{u},t) + \varphi \dot{\mathbf{x}}\} dt - [\varphi(t_1)\mathbf{x}(t_1) - \varphi(t_0)\mathbf{x}(t_0)] \qquad (1.13)$$

Hamiltonian function

From equation (1.13) the first two expressions under the integral sign are defined to be the **Hamiltonian function**:

$$H(\mathbf{x}, \mathbf{u}, \boldsymbol{\varphi}, t) \equiv F(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\varphi}\mathbf{f}(\mathbf{x}, \mathbf{u}, t) \qquad (1.14)$$

That is, the Hamiltonian function (called ***Hamiltonian***) is defined as the sum of the intermediate function (integrand) of the objective functional plus inner product of the vector of costate variables and the vector of functions defining the rate of change of the state variables.

For a maximum it is necessary that the change in the Lagrangian function must hold for a change in the control trajectory $\{\Delta\mathbf{u}(t)\}$, that:

$$\frac{\partial H}{\partial \mathbf{u}} = 0, \qquad t_0 \le t \le t_1 \qquad (1.15)$$

Necessary condition equation (1.15) states that the Hamiltonian function is maximized by choice of the control variables at each point along the optimal trajectory $\{\mathbf{u}^*(t)\}$.

To summarize, the maximum principle technique involves adding to the problem n costate variables $\varphi(t)$, defining the Hamiltonian function as:

$$H(\mathbf{x}, \mathbf{u}, \boldsymbol{\varphi}, t) \equiv F(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\varphi}\mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{1.16}$$

and solving for trajectories $\{\mathbf{u}(t)\}$, $\{\boldsymbol{\varphi}(t)\}$, and $\{\mathbf{x}(t)\}$ satisfying.

$$\max_{\{\mathbf{u}\in\Omega\}} H(\mathbf{x}, \mathbf{u}, \boldsymbol{\varphi}, t) \quad \text{for all } t, \ t_0 \le t \le t_1 \tag{1.17}$$

$$\dot{\mathbf{x}} = \frac{\partial H}{\partial \varphi}, \qquad \mathrm{x}(\mathrm{t}_0) = \mathrm{x}_0$$

$$\dot{\varphi} = -\frac{\partial H}{\partial \mathbf{x}}$$

The form of the solution for the optimal control problem often follows readily from the maximization of Hamiltonian, which usually gives the optimal control variables not as functions of time but rather as functions of the costate variables.

In particular, if the problem is autonomous in that both $F(\cdots)$ and $\mathbf{f}(\cdots)$ show no explicit dependence on time then the Hamiltonian shows no explicit dependence on time and, since $dH / dt = 0$, along the optimal trajectory the value of Hamiltonian is constant over time. Another advantage is that Hamiltonian functions are easier to solve than Lagrangian functions. Ultimately, it will produce the same optimum as Lagrangian approach.

1.5 Organization and Contribution of the dissertation

The rest of the dissertation is organized as follows. Chapter 2 introduces several trace back techniques and defines a trace back problem more formally. We present a simple and efficient algorithm for detecting the source of attack in a network. The algorithm uses the dynamic centroid decomposition technique to select nodes for monitoring packets to identify an attack packet. Advantage of the algorithm requires limited resources and does not require change in Internet protocols. Contribution of our work is to identify the set of routers that are requested to log, mark, or authenticate depending upon the type of attack. The number of routers identified for this task will be kept at a minimum yet sufficient to reduce the burden on the routers. In chapter 3, we describe the two classical simple epidemic models and an extended model, allowing for loss of immunity that causes recovered hosts to become susceptible again. With real Internet topology data, we find that there are two effective factors that influence Internet worm propagation: temporary immunization time and network delays. We note that our simulation results can explain how fast a virulent worm can spread and suggest effective mechanisms to monitor and defend against the propagation of worms. It also shows that we can find location(s) in the network that when quarantined would slow down the rage of spread. In chapter 4, we attempt to investigate a new approach to such optimal control problems of minimizing the cost of infection which can be interpreted as the network delay. Furthermore, we define the objective of minimizing the total cost of infection and derive the necessary conditions for our cost optimization problem which is solved numerically. We show that our simulation results can answer the

question of how many nodes needed to filter and when to start a filtering treatment, and this treatment of worm infection is very effective for reducing the spread of worm infection. Finally, Chapter 5 presents our conclusions and future work.

Chapter 2

On Intrusion Source Identification

2.1 Introduction

Network forensics is the science of analysis and detection of network based intrusions, including evidence gathering, and locating and isolating intruder(s). A well-known network based attack on computing resources is the Denial-of-Service (DoS) attack wherein the intruder sends several requests to the server so as to overwhelm the server and prevent it from serving legitimate user requests. The DoS attack can be either from a single intruder, a distributed set of intruders, or a distributed set of compromised hosts. Attacks of this nature can be *connection-oriented* involving TCP's three-way handshake protocol or *connectionless* that uses UDP packets. The source of the packets to the victim can be from the real intruder with possibly spoofed source IP address in the packets, or from the compromised machine(s). DoS attacks are considered *continuous* in the sense that a continuous stream of packets is sent from the intruder(s)

or the compromised machine(s) to the victim. An example of a *non-continuous* attack is the SQL Slammer attack wherein a single UDP packet contains the necessary code to attack SQL servers running on port 1434.

The *source identification* or the *trace back* problem deals with identifying the source of the intruder after the intrusion has been detected. Solutions to the trace back problem involve enabling the routers to monitor intruder's packets (packets that have a specific signature that has been singled out as a packet(s) that caused the intrusion) and executing a detection algorithm based on the information collected from the routers. For example, in packet marking schemes proposed in the literature, addresses of routers through which the packets are routed are added to the packet. When the *victim* (where the intrusion has occurred) gets the intruder's packets with addresses marked in them, it can reconstruct the path of the intruder's packets all the way to the source. If the source address is spoofed by the intruder, then the marking system will trace the origin of the packet all the way to the router that is uncompromised and closest to the source.

The rest of the chapter is organized as follows. Section 2.2 presents overviews of trace back techniques and their limitations and problems. Section 2.3 reviews recent solutions for trace back against DoS attacks. Section 2.4 discusses the trace back problem more formally and provides assumptions used by our detection system. Section 2.5 presents an algorithm that will use a minimal amount of network resources to either detect the sources of attack or perform quarantine operation that will isolate portions of the network from possible attacks. The conclusions and future work is presented in section 2.6.

2.2 Preliminaries

Recently, so many network security communities have made reasonably good progress in the development of attack prevention and intrusion detection systems for protecting hosts against network-based attacks launched remotely by attackers. However, because of the design of *Internet Protocol (IP),* back tracking the source of such attacks remains relatively difficult. Furthermore, it is difficult to eliminate spoofed packets in mounting denial of service (DoS) attacks on the Internet. This section describes techniques for tracing internet packets with spoofed source addresses back to their point of origin and presents their limitations and problems.

2.2.1 Overview of Trace back problems

A simple mechanism to prevent spoofed packets from leaving the subnet is to use Egress filtering wherein every packet's source address is examined to make sure that its source address matches the subnet from which the packet originated. Such internal policing can stop spoofed packets from entering the Internet, but this scheme will be beneficial only when all border routers cooperate. Certainly, not all ISP's and large networks can use Egress filtering [54]. Egress filtering cannot help in case of compromised machines sending attack packets with legitimate source addresses. Given the fact that some routers perform Egress filtering while others do not, the authenticity of the source IP address of the attack packet is still in doubt.

Yet another scheme to determine the subnet from which the attack packet originates is to force the border router at the subnet to mark the packet with its IP address. Such a

scheme would allow the victim to trace the attack packet closest to the subnet level and the border router at the subnet can be further instructed to block all the packets destined for the victim. This will work only when we assume that the subnet's router is not compromised. Clearly, this scheme adds additional complexity to the packet structure and incurs additional network bandwidth.

In order to trace back spoofed attack packets, the routers that border the subnets attached to the victim have to be requested to monitor packets. Not only the number of such routers can be very large, but needs also to be made an unreasonable assumption of a continuous stream of spoofed attack packets. The problem is further compounded if the attached border routers belong to different Internet Service Providers (ISPs), since all of them have to cooperate. In any case, a desirable solution is to perform logging of packets using SYSLOG or NETFLOW. These logging techniques are based on efficient storage mechanism such as a bloom filtering [33] and they provide tools for determining if a packet with a specific signature visited that router. These techniques range from simple matching to intelligent data mining [26, 32]. Placing effective monitors on every possible location in the network or marking every possible packet by all the routers is highly cost prohibitive and a severe drain on resources. For example, assuming that all packets through routers are logged, the victim can send a copy of the attack packet to all the SYSLOG or NETFLOW databases to be searched. Routers or its associated SYSLOG database that report the presence of the packet can "ring an alarm" and new routers attached to the "ringing" routers can be searched. To avoid resource draining process of logging every packet, the routers can be made to log packets on demand by the victim. A victim will request such a longer upon receiving

attack packets. The routers that "ring" and are farthest away from the victim would be requested to block the packets destined for the victim until the victim recovers. Clearly, the logging mechanism is very effective in finding source of non-continuous attacks.

A mechanism using IPSec security associations can be used to authenticate packets received from a router. For example, if a victim *v* would like to determine whether an intruder packet is routed from a particular router say *R*, then either *v* or the router closest to *v* can establish an IPSec with *R*. The premise of the approach is that if an attack packet has been correctly authenticated by a certain router *R*, the attack packet must have transited that router. Therefore, iteratively building security associations with routers at increasing distances from the victim will allow one to perform a secure trace route that will trace the attack packet to the router closest to it, even if the attacker used spoofed IP addresses. The technique proposed in this chapter will reduce the number of associations that need to be established to trace the router closest to the source.

2.2.2 Types of Attacks

Based on the discussion above we recognize that technique of logging, packet marking, or IPSec authentication is dependent on the type of attack. In general, the type to be used often depends on the attacker's motives and aims. Types of attacks can be classified as follows:

Destructive – Attacks which destroy the ability of the device to function, such as deleting or changing configuration information or power interruptions.

Resource consumption – Attacks which degrade the ability of the device to function, such as opening many simultaneous connections to the single device.

Bandwidth consumption – Attacks which attempt to overwhelm all available bandwidth capacity of the network device.

Furthermore, these attack types include continuous and non-continuous versions of single intruder and multiple intruders. For example, in the case of single intruder continuous attack the victim can request certain routers to mark the packets with the IP address in order to determine the route the packets take. For non-continuous attack proactive logging of packet information by certain routers would be very beneficial. A victim that sees a regular non-continuous attack can request a router to log the packets in a reactive sense. Our goal is to identify the set of routers that are requested to log, mark, or authenticate depending upon the type of attack. The number of routers identified for this task will be kept at a minimum yet sufficient lead to reduce the burden on the routers.

2.3 Recent Solutions for Trace back

Several types of DoS attacks have been identified [23, 24, 25], with the most basic DoS attack demanding more resources than the target system or network can supply. Resources may be network bandwidth, file system space, processes, or network connections [24]. While host-based DoS attacks are more easily traced and managed,

network-based DoS attacks which exploit weaknesses of the TCP/IP protocol suite [30], represent a more subtle and difficult threat [24, 34]. Network-based DoS attacks employ spoofing to forge the source address, and thereby hide identity of the physical source [29]. Previous works have focused on detecting DoS attacks and mitigating their detrimental impact upon the victim [27, 28].

A number of recent works have studied source identification (also called IP trace back [34]) which spans a range of techniques with their individual pros and cons. IP trace back is to identify the origin of sequential IP packets when the source IP addresses of these packets are spoofed. IP trace back is usually performed at the network layer, with the help of routers and gateways.

Link Testing

In link testing the identification of the physical source of an attack is done by tracing it back hop-by-hop through the network MAC addresses [39]. Trace back is typically performed manually, and is recursively repeated at the upstream router until the originating host is reached. The drawbacks of link testing include multiple branch points, slow trace back during an attack, communication overhead due to message exchange, and administrative constraints between network operators [39]. In behavioral monitoring [24], the likely behavior of an attacker during a DoS attack is monitored to identify the source. For example, an attacker may perform DNS requests to resolve the name of the target host which may not be resident in its local name server's cache. During a DoS attack, an attacker may try to gauge the impact of the attack using

various service requests including Web and ICMP echo requests. Logging of such events and activities can reveal information about the attacker's source.

Ingress Filtering

Packet filtering is a network mechanism for controlling what data can flow to and from a network affected routers or firewalls [42]. Filtering decisions, typically, are made based on packet content including source/destination addresses and port numbers. As a means of preventing network-based DoS attacks, ingress filtering in border gateways has been proposed for limiting IP source address spoofing [37, 38]. Ingress filtering requires a prolonged period to be broadly deployed on the Internet.

Probabilistic Packets Marking (PPM)

In packet-based trace back, packets are marked with the addresses of intermediate routers, in some sense, an inverse operation of source routing and similar to the IP Record Route option [31]. The victim uses information inscribed in packets to trace the attack back to its source. In this method, overhead in the form of variable-length marking fields that grow with path length, or traffic overhead due to extra messaging packets is incurred.

Probabilistic packet marking (PPM) [34, 35, 36] has been proposed for achieving space efficiency in the form of constant marking field and processing efficiency in the form of minimal router support. The basic idea of the approach is that routers probabilistically encode partial path information into the packets during forwarding and try

to reconstruct the complete path from the packets that contain the marking. In spite of its efficiency properties, PPM has several drawbacks: packet storage requirements and high router overhead to record the path information. For a large amount of packets, it may result in unnecessary fragmentation. To reduce the resource overhead, a hash-based technique was proposed to store the information into 16-bit IP Identification field used for fragmentation in the IP header. However, the ID field of IP header can not be modified if either fragmentation is necessary or IPSec authentication is provided. In addition, it is necessary for a victim to accumulate huge amount of data in order to determine true attack path. Improved marking schemes including authentication were studied in [35].

IPSec authentication and encryption

Chang, et al [43, 44] proposed a security management framework, DECIDUOUS (*Decentralized Source Identification for Network-Based Intrusions*), to securely identify attack sources by using existing network security protocols and services, specifically IPSec authentication and encryption services. With this method, when an attack is detected, the Internet key exchange (IKE) protocol establishes IPsec security associations (SAs) between the target host and some routers in the administrative domain (for example, autonomous system boundary routers). Routers at the SA ends add an IPsec header and a tunnel IP header containing the router's IP address to traversing packets. If the attack continues and one of the established SAs authenticates a subsequent attack packet, the attack must come from a network beyond the

corresponding router. The receiver checks the source IP address of the tunnel IP header to find out which routers the attack packet traversed. Repeating this process recursively, the receiver finally reaches the attack source. Because this technique uses existing IPsec and IKE protocols, implementing a new protocol for tracing is unnecessary.

ICMP Trace back Message (iTrace)

ICMP trace back proposes to introduce a new message "ICMP trace back" (or an iTrace message) so that routers can generate iTrace messages to help the victim or its upstream ISP to identify the source of spoofed IP packets [40]. For example, routers would be modified to randomly (for example, one trace back message for every 20,000 packets) generate a trace back message about a packet and send it to the packet's destination. Each trace back message would provide authenticated information about the packet being traced, what time it was sent, where it came from, where it went. With enough trace back messages from enough routers, a network manager could find the source of a spoofed flow. Of course, this would require that the Internet routers would have to be modified to support the new ICMP trace back. An intention-driven iTrace is also introduced to reduce unnecessary iTrace messages and thus improve the performance of iTrace systems [41].

Hop-by-hop Input Debugging

Robert Stone's CenterTrack uses an overlay network of IP tunnels to selectively reroute suspicious datagrams from edge routers to special tracking routers [45]. The

tracking router can determine the ingress edge router by noting the tunnel on which the packet arrived. The tracking router can inspect the suspicious datagram and then either drop it or forward it. The scheme permits rerouting flooding packets and can determine the ingress point on the enterprise network.

Logging and Storage

Snoeren et al [46] describe a hash based technique and an implementation of digest tables using space-efficient data structures known as Bloom filters; it records packet digests for recently forwarded traffic within the network and reconstructs the attack paths with these digests. A software engine called Source Path Isolation Engine (SPIE) that uses the packet digests is proposed in [46]. Using the SPIE environment, it has been shown that tracing attacks that use single packet rather than a series of packets is feasible with low storage requirements. The packet digest is computed over the invariant bytes of a packet header and the first 8 bytes of payload. This approach is based on the assumption that this packet digest will not be frequently modified by a packet transform. However, if the invariant portion of a packet header is used and modified frequently to store extended information, then this assumption is infeasible. In this case, packet transformations will occur frequently and result in resource overhead of logging transformation information.

2.4 Trace back: Centroid Approach

Definition 2.1 (*Attack paths and tree*): The attack propagation model of a network is given by the undirected network $G = (V, E)$, where V is the set of nodes and E is the set of edges. The set of nodes V could be further partitioned into end systems and routers. The edges denote physical links between elements in V. Let $A_i \in V$ denote the potential attack source, and let $V_t \in V \setminus A_i$ denote the victim. In case of a single attacker, $|A_i| = 1$, and the path $P_i = (A_i, R_1, R_2, \ldots, R_d, V_t)$ composed of d routers $R_1, \ldots, R_d$, an attacker A_i, and a victim V_t is called an ***attack path***. In other words, an *attack path* from A_i is the sequential route that the attack packet has traversed between A_i and V_t. If $|A_i| > 1$, then we have distributed DoS attack (DDoS) wherein the attack paths are joined together to form an ***attack tree*** rooted at the victim. ■

Definition 2.2 (*Trace back Problem*): Let $G' = (V', E')$ be an attack network of G, where V' represents the set of nodes associated with attack path(s) and edge $(u, v) \in E'$ represents a link on which an attack at $u \in V'$ propagates to $v \in V'$. In other words, the removal of vertices Vi not associated with attack path(s) from the network G results in the induced attack network G', e.g., from figure 2.1 we obtain an attack network G′ containing two attack paths, $P_2 = (A_2, R6, R3, R2, R1, V_t)$ and $P_3 = (A_3, R7, R4, R2, R1, V_t)$ by removal of nodes A_1 and R_5. The ***trace back problem*** is to construct an attack network G' containing the attack path(s) and the associated attack source(s) for each victim. ■

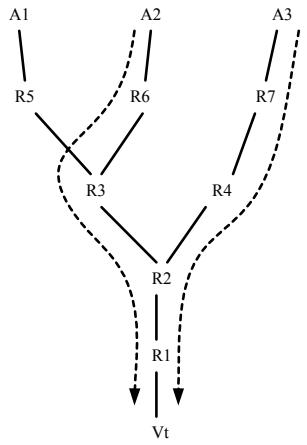


Figure 2.1: Attack paths: (A_2, *R6, R3, R2, R1*, V_t) and (A_3, *R7, R4, R2, R1*, V_t).

The attack paths form an attack tree is shown above.

2.4.1 Assumptions

We state in the following the assumptions on the mode and operations pertaining to the attack on the victim by the intruder(s).

1. Attackers may generate any packet
2. Attacker may disguise its IP source address
3. Routers are both CPU and memory limited
4. Routers are not widely compromised
5. Routers always choose the shortest routes with least hops to forward packets
6. In the case of continuous attack (like the DoS), the route taken by the attack packets is stable.

The assumption that routers always select a shortest path to forward packets is probably the fundamental property of our proposed algorithm. Routing path selection in several deployed routing protocols is based on well-known shortest path algorithms. It follows that packets from an attacker to victim must be transmitted through the shortest path. The final assumption that routing would be systemically stable until an efficient tracing system determines the attack source is the most controversial. Paxson [47] states that two packets sent by the two same end hosts may take different directions of the Internet paths due to network congestion. Labovitz et al. [48] have also shown the routing instability from BGP routing messages. However, it is very difficult for any tracing system to seek to determine attack source with multiple attack paths. Chinoy [49] measured that almost 90% of the EGP routing updates in the NFSNET system of networks contained close to 0% new information (whereas EGP updates occur every 3 minutes). Furthermore, Govindan and Reddy [50] used a year's worth of inter-domain routing traces collected in 1994-95 and analyzed the Internet inter-domain topology, its routing stability behavior. Shaikh and Kalampokas [51] have performed extensive experimentation and developed analytical models to capture the stability and robustness properties of routing protocols in congested networks. It shows that the path through which packets are transmitted between two end hosts does not change frequently despite the growth of the topology. As a result, routing of multiple attack packets should be stable during the period of the trace time to identify the attack source. These last two assumptions motivate us to look for a scheme which transfers a general network topology to a terminal network (described in next section) of the

network in order to simplify network topology and reduce the amount of network resources required to perform trace back.

2.4.2 Terminal Nodes

Terminal nodes are routers on the network that see large amount of network traffic. Typically border routers that connect autonomous areas to the rest of the Internet and Internet core routers experience heavy network traffic. At the subnet level, these are routers that connect one subnet to the other as they join to form the autonomous system. Our proposed solution for the trace back problem makes use of these terminal nodes wherein logging, marking, or IPSec association is performed. The number of terminal nodes can be large considering the fact that there are over 2 million core routers.

Consider the Abilene Network which is an Internet2 high-performance backbone network that connects hundreds of end users that range from universities, research labs, and technology companies. The structure of the Abilene network is shown in figure 2.2. Traffic from the west coast can reach the east coast through a combination of two routers selected one each from sets {Kansas City, Houston} and {Indianapolis, Atlanta}. These routers experience plenty of traffic compared with routers at the edges say Seattle for example. Our goal is to identify these routers based on the topology of the network and use these nodes to monitor intruder packets. As a first step, we will imagine that our network is a set of glued bi-connected components. The bi-connected components of the Internet2 backbone network are {Sunnyvale, Seattle, Denver}, {Sunnyvale, Denver, Kansas City, Houston, Los Angeles}, {Kansas City, Houston,

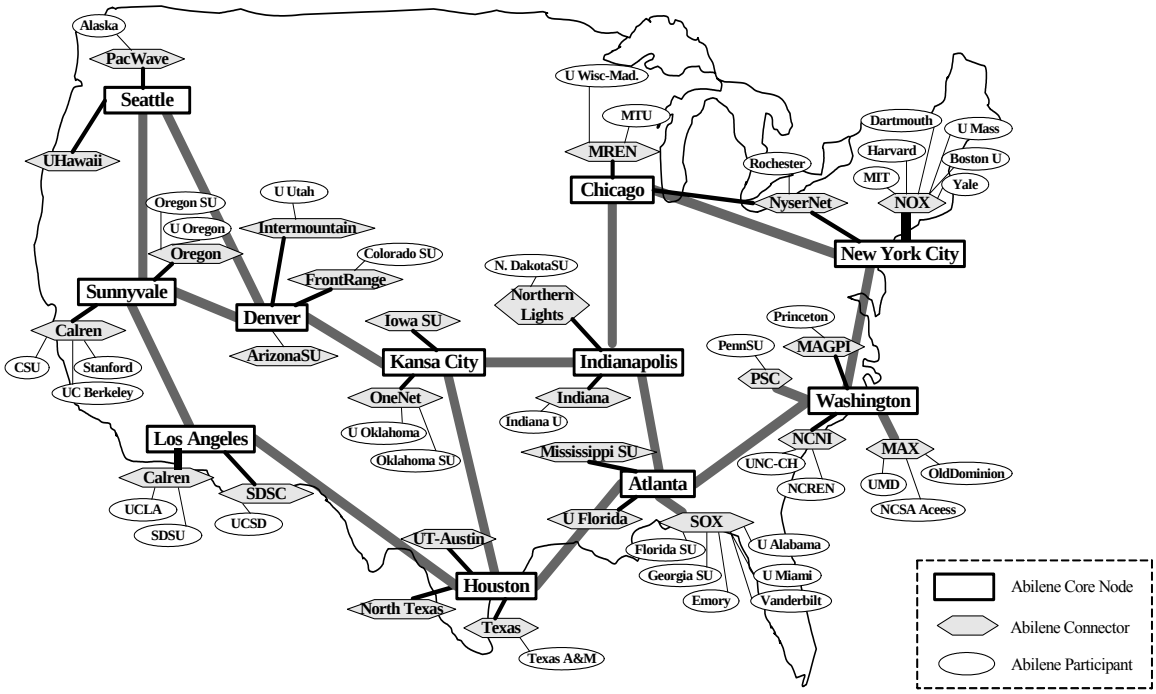


Figure 2.2: Abilene Network

Atlanta, Indianapolis}, and {Indianapolis, Atlanta, Washington, New York City, Chicago}.

The nodes that connect one bi-connected component with the other are *terminal nodes* and traffic through these terminal nodes are generally higher compared with other nodes. In figure 2.2 the terminal nodes are Sunnyvale, Denver, Kansas City, Houston, Atlanta, and Indianapolis.

In the next subsection, we present an algorithmic technique to recognize terminal nodes that is based purely on the topology of the network.

2.4.3 Terminal Network

The approaches presented for trace back problems all involve cooperation of intermediate routers in the network. Our main goal is to reduce the number of intermediate routers that participate in the trace back solutions. To this end, we identify a small number of set of nodes that are enough to complete the trace back based on the assumptions on the network and on its routing presented in section 2.4.1.

Given a network G, we first construct a set of connected components that does not exceed the given size (number of nodes) t.

Algorithm BCC (H, t)

Input: The network $H = (V, E)$ with nodes V and links E, and a size t.

Output: A set of connected components.

Begin

1. **If (**H is a tree) **Then**
2. **Return** H
3. **Else**
4. **If** ($|H| < t$) **Then Return** H
5. Test connectivity of H and let it be k.
6. Choose k nodes to make the network H disconnected.
7. Remove the k nodes (cut nodes) from H and let H_1, H_2, …, H_m be the connected components.

8. Add the k nodes to each of the connected component H_i, $1 \leq i \leq$ m with links (p, q), where p is a cut node, q is a node in H_i, and (p, q) is a link in E.
9. Mark the k nodes of each connected component to indicate that they are terminal nodes.
10. **Return** ({BCC(H_1,t), BCC(H_2,t) …, BCC(H_m, t)})
11. **EndIf**

End.

The main idea behind this is that the intruder resides in a connected component and searching only the terminal vertices would lead us to that connected component. Let H be k connected with $k \geq 2$. We need to identify k nodes whose removal will make the network not connected. The set of k vertices will be added to the set of terminal vertices.

We will remove these k nodes and apply the above algorithm on each of the remaining connected component until either each component is 1-connected or its size does not exceed t. The larger the size of t faster the above algorithm will terminate. If t is equal to n, the number of nodes in the network, then the entire network will become a terminal network and hence more network resources have to be committed for the trace route problem. On the other hand if t is smaller, then the number of bi-connected components identified will be large and so will the number of terminal nodes. Polynomial-time algorithms exist to test the connectivity of the network and to find cut vertices.

Definition 2.3 (*Terminal Network*): A *terminal network* $TG = (V', E')$ of a network G is an edge weighted network that contains nodes $\{u, v\} \in V'$, where u and v are terminal nodes of G and link $(u, v) \in E'$, if and only if, nodes u and v belong to the same connected component that results after the execution of the algorithm BCC. The weight on the link $(u, v) \in E'$ is the shortest distance between vertices u and v in G. ■

The terminal network of the network shown in figure 2.3 is shown in figure 2.4.

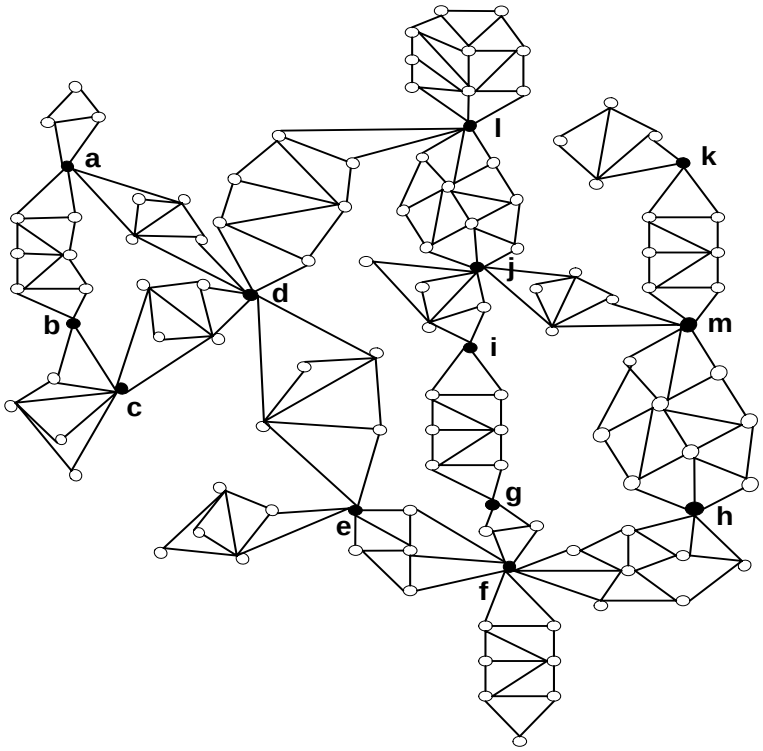


Figure 2.3: A network with terminal nodes highlighted and labeled.

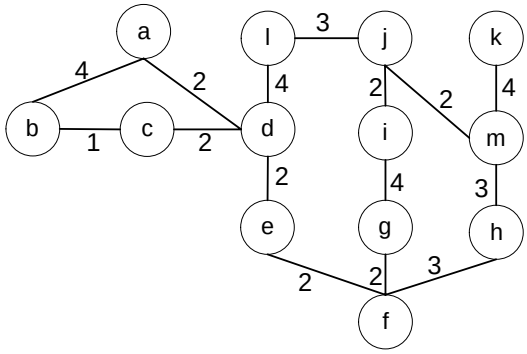


Figure 2.4: The terminal network of the network in Figure 2.3. The weights on the edges are the distances in terms of number of hops between the marked vertices in the network shown in Figure 2.3.

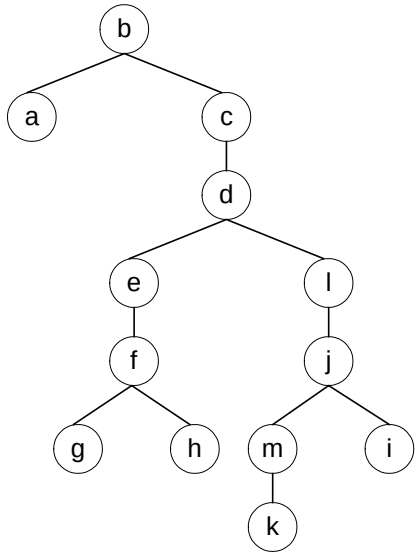


Figure 2.5: The single source shortest path tree with node '*b*' as the root of the tree.

2.5 Detection Algorithm

After the terminal network is constructed the single source shortest path tree starting with the terminal node that is closest to the victim is constructed from the terminal network. A single source shortest path tree is shown in figure 2.5.

Considering the tree network as shown in figure 2.5, let node *b* be the router that is closest to the victim. Assume that we are dealing with a continuous attack from a single source. A straight forward track back approach works as follows. Assume that router *k* (in figure 2.5) is forwarding the attack packets. Node *b* without any knowledge first requests node *a* to mark the packets. This operation will be termed as *placing an alarm* at node *a*. If the marked packets are not the attack packets, then it can request node *c* to mark the packets. Node *b* now will recognize that the attack packets are from node *c* ("ringing" node) and it will initiate a track back request to node *c* (for the subtree rooted at node *c*). Node *c* will perform similar operations to that of node *b* until node *k* is reached.

Clearly, this straight forward approach will require that all nodes mark the packets at one time or the other and hence is a severe drain of network resources. If we request more than one node to mark the packets simultaneously, then we can speed up the process of trace back, but it does not improve resource usage efficiency. In the case of a distributed DoS attack, there will be more than one ringing node that is on different paths from root to leaf nodes in the shortest path tree. For such cases, the trace back will be applied to each subtree rooted at the ringing nodes. In summary, for the continuous attack scenario, a given set of alarms is *incrementally* placed at more than

one location and depending on the ringing and non-ringing of the alarms the old alarms are removed and new alarms are placed. Instead of working with the network in figure 2.5, we can apply the same approach as above on the centroid tree. Since the depth of the centroid tree is no more than $O(\log n)$, the trace back can be completed in $O(\log n)$ time. This concept is explained in section 2.5.1.

2.5.1 The Centroid Approach

Our proposed algorithm requires a *centroid decomposition* technique on a tree network. Every tree T has a centroid consisting of either one vertex or two adjacent vertices [52]. For each vertex $v \in T$ of degree 2 or more, count the number of vertices in each of the subtrees emanating from v, and let n_v be the maximum of these numbers. If the tree has n vertices it can be shown that either there is just one vertex v for which $n_v \leq (n-1)/2$ or there are two adjacent vertices v and w for which $n_v = n_w = n/2$. We can determine a centroid of the tree T by repeatedly removing nodes of degree one until either a single vertex remains or an edge remains. A *centroid decomposition* is the process of repeatedly finding the centroids on subtrees obtained by removing every edge incident on the centroid. Given a n-node tree the centroid decomposition can be completed in $O(n)$ time [53].

Observation 2.1 [in 53]**:** Given a tree T with n nodes, the size of each connected component obtained by the removal of the centroid is no more than $n/2$. ■

Observation 2.2: Based on Observation 2.1, it can be clearly seen that the depth of the centroid tree T_C is $O(\log n)$. ■

A *centroid tree* T_C of a tree T is obtained using the algorithm CentroidTree.

Algorithm CentroidTree (T)

Input: The tree network T.

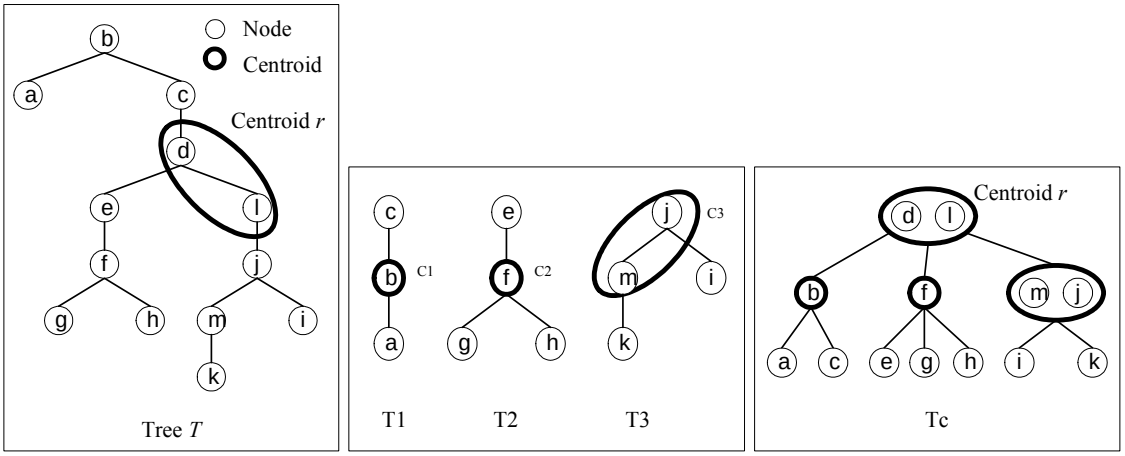
Output: The Centroid Tree of T_C.

Begin

1. The centroid of T is the root r of the tree T_C.
2. let subtrees T_1, T_2, …, T_k be obtained by removing r from T; the centroids c_1, c_2, …, c_k of the subtrees T_1, T_2, …, T_k, respectively are the children of r,
3. each node x in T_C serves as the root of the centroid tree of the subtree T_x.

End.

The centroid decomposition process on a tree network and its corresponding centroid tree are illustrated in figure 2.6.



(a) (b) (c)

Figure 2.6**:** (a) Find a centroid r in a tree T, (b) Find subtrees T_1, T_2, and T_3 by removing r and the centroids c_1, c_2, and c_3 from each subtree, (c) construct a centroid tree T_c with node 'r' as the root of the tree

2.5.2 The Algorithm

The algorithm to perform the trace back assumes that preprocessing has been completed and the terminal network has been constructed. Once the victim is identified, then the router closest to it in the terminal network is chosen as the root of the single source shortest path tree as explained previously. The alarm is placed on the centroid or its neighbors as explained below. Depending on the 'ringing' and 'non-ringing' of the alarm the new set of alarms is placed on the centroid of the sub trees that remain after the previous centroid is removed. This process is continued until the terminal router closest to the intruder is identified.

Algorithm Detect_Intruder

Input: The network and the router closest to the victim

Output: The router closest to the intruder

Begin

1. First construct the terminal network and find the single source shortest path tree T of the terminal network with the node in the terminal network closest to the victim as the root.
2. Find the centroid of T and place an alarm either on the centroid or the neighbors of the centroid as specified in the more detailed description below.
3. Determine the subtrees obtained after removing the centroid and if attack packets are discovered in a node v on which an is alarm is placed, then execute step 2) on a subtree T_v containing the node v. If there is more than one node alerting an attack, then choose the one that is *farthest* from the root of the tree. The above process is continued until the intruder is detected.

End.

Based on observation 2.2 and the above algorithm, it is evident that the number of alarm placement steps is bounded by $O(\log n)$ for a n-node tree. Step 1) of the above algorithm can be constructed using Dijkstra's shortest path algorithm in $O(m+n\log n)$ where m and n are the number of links and nodes of the terminal network.

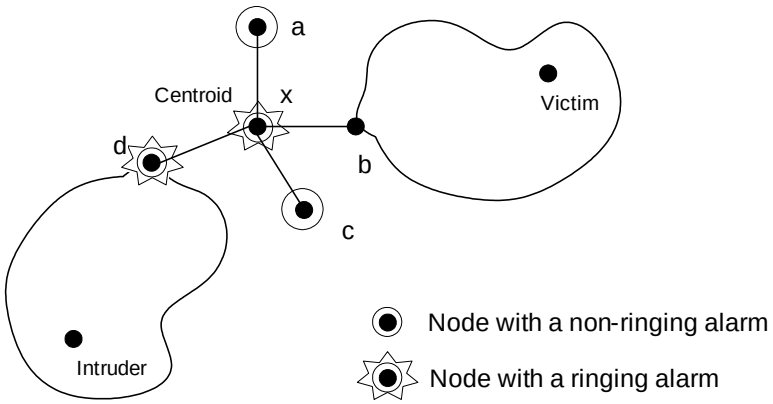


Figure 2.7: Alarms are placed on nodes *a*, *c*, *d*, and *x* and the ringing ones are *d*, and *x*. Node *d* is the closest to the intruder and the tree rooted at *d* is processed next.

For step 2) of the algorithm, as pointed out earlier the centriod can be found in $O(n)$ time and after we find the centroid we place the alarms as follows. Assuming that we have a large number of alarms available, we should place the alarms on the centroid and its neighbors.

Figure 2.7 illustrates alarms and shows the ringing alarms based on the location of the intruder. For example, in figure 2.7, based on the position of the intruder the alarms placed on node *x* (the centroid) and nodes *d* will ring. Node *d* is farthest from the victim and hence the subtree rooted at *d* will be chosen next and its centroid will be found. This process will continue until the terminal node closest to the intruder is found. If more than one such node *d* exists due to attacks from multiple sources, then each tree rooted at such *d*'s will be processed as above.

2.6 Concluding Remarks

We have introduced several techniques for tracing internet packets with spoofed source addresses back to their origin of attacks. Even though those several proposals are worthy attempt to be applied in different tracing techniques for different attack types, there exist some practical limitations for implementation such as resource bound, modification of Internet infrastructure, and so on.

In this chapter we have presented a simple and efficient algorithm for detecting the source of attack in a network. The algorithm uses the dynamic centroid decomposition technique to select routers for monitoring packets to identify the one with signatures of an attack packet. To simplify network topology and reduce the amount of network resources required to perform trace back, we need a scheme which transfers a general network topology to a terminal network of the network.

In summary, we ensure that many of the existing trace back techniques can be used in our algorithm in order to obtain the minimal number of network entities on which alarms (or monitors) are placed to determine the source of attacks under more sophisticated attacks. In other words, our proposed solution for the trace back problem involves a very small fraction of routers wherein logging, marking, or IPSec association is performed. In addition, our proposal could serve as the basis for future research work on quarantine of potential sources for large distributed attacks.

Chapter 3

Measurement and Analysis of Worm Propagation on Internet Network Topology

3.1 Introduction

In the area of virus and worm modeling, many studies have employed simple epidemiological models to understand general characteristics of worm's propagation. Epidemiologic propagation models have traditionally been used to understand and model the spread of biological infectious diseases [9, 10]. The time required for finding the target node to be infected and the rate of infection were assumed to be a constant in many propagation models proposed in the literature [5, 8]. A constant infection rate is

reasonable for modeling epidemics but may not be valid for real Internet viruses and worms. The reason is that most classical epidemic models are homogeneous, in the sense that an infected host is equally likely to infect any of the susceptible hosts while Internet is non-homogeneous. In addition, current propagation studies have not considered the real Internet topology data and exploited characteristics of the network topology.

3.1.1 Immunization Defense of Worms

Previous works on worm modeling neglect the impacts of multiple worm outbreaks on our computer networks. Nowadays, new network worms will continue to be created while the strains of old worms will continue to circulate around the Internet. Recently, the Blaster worm, known as MSBlast or LoveSAN, has infected an average of about 2,500 new systems hourly running Microsoft operating systems that are unpatched for the so-called RPC vulnerability [16]. It is noted that a huge number of infected hosts is a substantial rate of infection, though the several hundred thousand hosts may be still infected by other old Internet worms including Slammer, Code Red and Nimda. In other words, many new viruses and worms come out every day, though most of them die away without infecting many computers due to human countermeasures including using antivirus software, patching susceptible computers, disconnecting network services and so on. Thus, any proposed defense mechanism must be evaluated in handling many active worms simultaneously. Wang et al [12] investigated the immunization defense on different network topologies including hierarchical and

clustered. Immunization can be thought of as effective packet filtering. Immunization from one worm does not guarantee protection against other forms of the worm. Wang et al [12] considers permanent or static immunization where a node once immunized is permanently protected. In reality, immunization must be taken as temporary due to multiple worm outbreaks since a computer being recovered from a certain worm can be reinfected by other worms immediately. In other words, any computer could not be permanently immune to many Internet worms.

3.1.2 Characteristics of Worm Spreading

In order to defend against future worms, we need to understand the network characteristics of worm spreading. Clearly the following characteristics of worm must be well understood before the model of Internet worm propagation could be developed.

1. The rate and pattern of infection,
2. The effect of factors on underlying network topology, and
3. The human countermeasures in the network

If such characteristics were known, mechanisms might be developed to detect an on-going, network wide infection. Certain nodes of the Internet are well protected compared with the others. Moreover, at certain vital installations the rates at which infections are cured are higher compared with others. To model this real world phenomenon we have taken into account in our simulations variable infection rates and variable cure rates.

Also in this chapter, with real Internet topology data, we find that there are two effective factors that influence worm propagation: temporary immunization time and network delays. We note that our simulation results can explain how fast a virulent worm can spread and suggest effective mechanisms to monitor and defend against the propagation of worms. It also shows that we can find location(s) in the network that when quarantined would slow down the rage of spread.

The rest of the chapter is organized as follows. Section 3.2 reviews the analytical methodologies of Internet worms. In Section 3.3, we give a brief review of the classical epidemic models and point out their limitations to model Internet worm propagation. In Section 3.4 and 3.5, we show the simulation results based on different network topologies. We conclude the chapter with an outline of our future work in section 3.6.

3.2 Analytical Methodologies of Internet Worms

Classical Epidemic model

In epidemiology research, there exist several deterministic and stochastic models for virus spreading. About ten years ago, Kephart and White [5] presented the Epidemiological model to understand and control the prevalence of viruses. This model is based on biological epidemiology and uses nonlinear differential equations to provide a qualitative understanding of virus spreading. They assumed that classical epidemic models are all homogeneous, which means that an infected host is equally likely to infect any of other susceptible hosts. Though at that time the model assumptions were considerably accurate because they considered that infection takes

place when hosts share their disks, but with the spreading on the Internet such assumptions are no longer valid. They also introduced an analytical model called SIS model in which infected hosts become susceptible once being cured of the infection.

Two-factor Worm Model

The Code Red worm incident of July 2001 has been investigated to model and analyze Internet worm propagation. Zou et al [11] introduced that there were two factors affecting Code Red propagation: one is the effect of human countermeasures against worm propagation; the other is the slower worm infection rate due to Internet congestion caused by Code Red worm. Based on the classical epidemic models, they derive a new general Internet worm model called *two-factor worm* model, which matches the observed Code Red worm data of July 19th 2001 with their simulation results and numerical solutions.

Active Worm Model

Chen et al [13] present a model, referred to as the Analytical Active Worm Propagation (AAWP) model that characterizes the propagation of worms that employ random scanning. They compare their mathematical model with the Epidemiological model and Weaver's [14] simulation results which use hit list scanning. The AAWP model shows that the model can be applied to monitoring, detecting and defending against the spread of active worms. The AAWP model can be also extended to Local AAWP model to understand the characteristics of the spread of worms that employ local subnet scanning effectively.

Applying Infection Delay in Worm Epidemic

Wang et al [12] introduced an analytic model to capture the impact of underlying topology in computer viral propagation. The simulations are conducted to attempt to answer the question – how a virus propagates in real network. They assume that an infection rate for each edge and a cure rate for each infected node are constant. In addition to the spread of a virus in real network, Wang and Wang [20] investigated the model extending the classical epidemic model by including two specific parameters: infection delay and user vigilance time. The infection delay is a period of time between the arrival of a virus on certain node and further infection from that node. The user vigilance time is the immune time. The model of capturing the effective of infection delay and user vigilance was validated by simulation analysis based on the homogeneous SIS epidemic model. In real networks however, the worm infection rate and cure rate are not likely to be a constant.

We also examined several major characteristics of infection, including the variant rate and pattern of infection through the different network topologies and the rate of re-infection at each host during an attack. We use a discrete time model and deterministic approximation to describe the spread of Internet worms.

3.3 Worm Propagation Models

The epidemic propagation models for the study of biological infectious diseases have been applied on modeling the propagation of computer viruses [5, 8]. The propagation of a real worm on the Internet is a complicated discrete event process. In this section

we consider only continuous process and use the continuous differential equations to describe it, which means that a worm on an infectious host continuously tries to find and infect other susceptible hosts. We introduce two classical deterministic epidemic models and an extension of one of models, which are the basis of our experimental design. We also point out their limitations when we try to use them to model Internet worm propagation.

3.3.1 Definition

In classical epidemic model, it is defined that a host is called an *infectious* host at time *t* if it has been infected by virus before *t*. A host that is vulnerable to virus is called a *susceptible* host. By infection and cure rate, we mean the probability with which an infectious host send infective messages to its neighbors and the probability with which an infectious host will be cured of the infection once it received infective messages from its neighbors, respectively. In addition we define that the *temporary immunity* is a temporary hold on a worm spreading, which means that many hosts will be susceptible or infected by new worm outbreaks at time *t* though they are already immune to old worm that came out before time *t*.

3.3.2 Classical simple epidemic model

In classical simple epidemic model, each host stays in one of two states: susceptible or infectious. Each susceptible host becomes an infectious one at a certain rate. At the same time, infectious hosts are cured and become again susceptible at a different rate.

Notation	Definition
N	Size of total vulnerable population
$S(t)$	Number of susceptible hosts at time t
$I(t)$	Number of infectious hosts at time t
$R(t)$	Number of removed infectious hosts at time t
β	Infection rate
δ	Curing rate on an infectious host
λ	Removal rate on an infectious host
μ	Re-susceptible rate on a removed host
ρ	Epidemic threshold

Table 3.1 Notations of Worm Epidemic Models

This model system where having the infection and being cured does not confer immunity. This model is called the SIS model, because hosts move between the *S* (Susceptible) and *I* (Infectious states). Using the terms defined in table 3.1, the differential equation for the SIS model is

$$\frac{dI(t)}{dt} = \beta I(t)[N - I(t)] - \delta I(t) \qquad (3.1)$$

where $I(t)$ is the number of infectious hosts at time t; N is the size of population; β is the infection rate; and δ is the cure rate.

We assume that at beginning, t =0, one host is infectious and the other ($N-1$) hosts are all susceptible. Let $S(t) = N - I(t)$ denote the number of susceptible hosts at time t. Replace $I(t)$ in equation (3.1) by $N - S(t)$ and we get

$$\frac{dS(t)}{dt} = -\beta S(t)[N - S(t)] + \delta[N - S(t)] \tag{3.2}$$

Therefore we say the SIS model is defined by:

$$\frac{dI}{dt} = \beta SI - \delta I \tag{3.3}$$

$$\frac{dS}{dt} = -\beta SI + \delta I$$

The solution to the equation (3.1) is

$$I(t) = \frac{I_0(\beta N - \delta)}{I_0\beta + (\beta N - \delta - I_0\beta)e^{-(\beta N - \delta)t}} \tag{3.4}$$

We conclude that, as $t \to \infty$,

$$I_\infty = N - \rho \tag{3.5}$$

where $\rho = \frac{\delta}{\beta}$ and I_0 is the initial number of infectious hosts. Therefore, not absolutely all the population gets infected. This shows that each infectious host infects others with an average value of β per unit time.

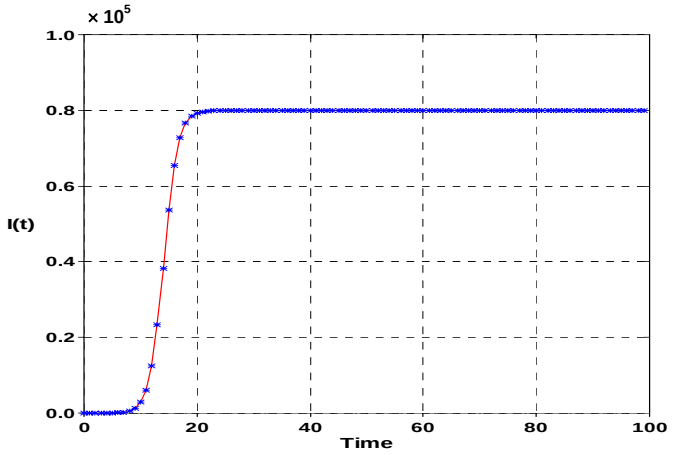


Figure 3.1: Classical simple epidemic (SIS) model

However, the probability that a host becomes infected is not the same for every host because it is a function of their connectivity and the infection characteristics with a certain cure rate. We note that the probabilities per unit time of infection and of cure are independent. Once a host is cured, it is immediately capable of being re-infected. Figure 3.1 compares the number of infectious hosts as a function of time as obtained from equation (3.4). The graph contains 100,000 hosts and the infection and cure rates are $\beta = 1.0$ and $\delta = 0.2$, respectively. It shows that the number of infectious hosts is nearly exponentially increased from $t = 0$ to $t = 20$.

The number of infections stops increasing when about 80% of all susceptible hosts have been infected. The SIS model does not take into account the possibility of host's removal due to death or immunization which would lead to the so-called *Susceptible-Infectious-Removed* (SIR) model [9]. It also does not model secondary effects such as reduced infection rate due to network congestion when many hosts are infected [11].

3.3.3 Kermack-Mckendrick model

In epidemiology modeling, Kermack-Mckendrick model considers the removal process of infectious hosts [9]. This model is called the classical SIR epidemic model. Kermack-Mckendrick model can be described as shown in figure 3.2. Each host is assumed to be in one of three states: *Susceptible* (S) meaning vulnerable to the virus, *Infectious* (I) meaning infected and actively infecting other hosts, and *Removed* (R), which corresponds either to immunity from the virus, or death at a constant rate.

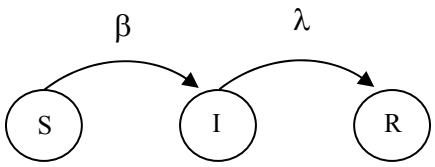


Figure 3.2: The SIR model in which hosts move between three states: *Susceptible* (S), *Infectious* (I) and *Removed* (R) with infection rate β and removal rate λ.

In this model, the assumptions are that susceptible hosts become infected by contact with infectious hosts, infectious hosts either die or recover at a constant rate, and the total population is constant. The sizes of the susceptible and infectious populations therefore evolve according to the following equations based on the simple epidemic (SIS) model.

$$\frac{dI(t)}{dt} = \beta S(t)I(t) - \lambda I(t)$$

$$\frac{dS(t)}{dt} = -\beta S(t)I(t) \qquad (3.6)$$

$$\frac{dR(t)}{dt} = \lambda I(t)$$

where β is the infection rate; λ is the rate of removal.

The Kermack-Mckendrick model improves the SIS epidemic model by considering that some infectious hosts are immune, are placed in isolation, or have died. However, this model is still not suitable for capturing the effect of multiple worm propagation simultaneously. First, in the Internet, many new viruses and worms come out every day though most of them disappear due to human countermeasures including using antivirus software, patching susceptible computers, disconnecting network service from the infectious hosts and so on. In other words, many hosts will be susceptible or infected by new virus outbreaks at time *t* though they are already immune to recovered old virus that came out before time *t*. But in Kermack-Mckendrick model once infectious hosts recover, they will not be infected again by any virus and stays in the "removed" or "immunized" state forever. The link delays required for the infection to travel to the hosts are captured in the aggregate value called infection rate. While such gross estimates are correct for long lasting worms, it does capture neither the short lived ones nor the vulnerability of nodes which are reachable quickly. In this chapter, we consider that the propagations of most Internet worms are topology dependent and

need to be modeled by considering the properties of the underlining topology, which will be discussed in a later section.

3.3.4 An extension for the SIR model

We assume that a more general case, allowing for loss of immunity that causes recovered hosts to become susceptible again. In other words, a portion of the removed hosts a time t, $R(t)$, due to loss of immunization join the susceptible population at time $t + \tau$, $S(t+\tau)$. Therefore a portion of population dynamically changes from susceptible to infectious, to removed and back to susceptible. Model that describes such an epidemical cycle is referred to as SIRS model. If hosts in the R state are only temporarily immune, the diagram becomes,

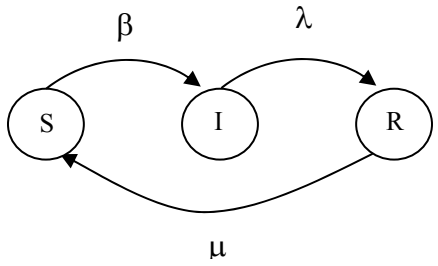


Figure 3.3: The SIRS model; with conferring a temporary immunity, it can move from the R state to the S state

Our model is a generalization presented in [10], allowing hosts recovering from the infective to go into a temporarily immune state rather than directly back into the

susceptible state. Let μ be the rate at which removals loose the immunization and becomes susceptible. Using the same notation as the SIR model we obtain the following deterministic SIRS model:

$$\frac{dS(t)}{dt} = -\beta I(t)S(t) + \mu R(t)$$

$$\frac{dI(t)}{dt} = \beta I(t)S(t) - \lambda I(t) \tag{3.7}$$

$$\frac{dR(t)}{dt} = \lambda I(t) - \mu R(t)$$

Also, we have $S(t) + I(t) + R(t) = N$, $\forall t \geq 0$. We can supply the same initial conditions as with the SIR model and numerically solve the SIRS model. Let $\rho = \lambda/\beta$ be the epidemic threshold if re-susceptible rate, μ, is less than removal rate, λ, and I_0 and S_0 are the initial fraction of infectious hosts and of susceptible hosts, respectively. For the epidemic to occur, we must have:

$$\frac{dI}{dt}\Big|_{t=0} > 0 \rightarrow \beta S_0 I_0 - \lambda I_0 > 0 \rightarrow S_0 > \frac{\lambda}{\beta} \tag{3.8}$$

Clearly S_0 must satisfy this condition for the epidemic to occur. The equation (3.8) indicates that no epidemic occurs if the initial number of susceptible hosts is smaller than the epidemic threshold, $S_0 < \rho$. This important result of the threshold effect is the same as what was already discovered by Kermack and McKendrick [9]; the population must be "large enough" for a disease to become epidemic.

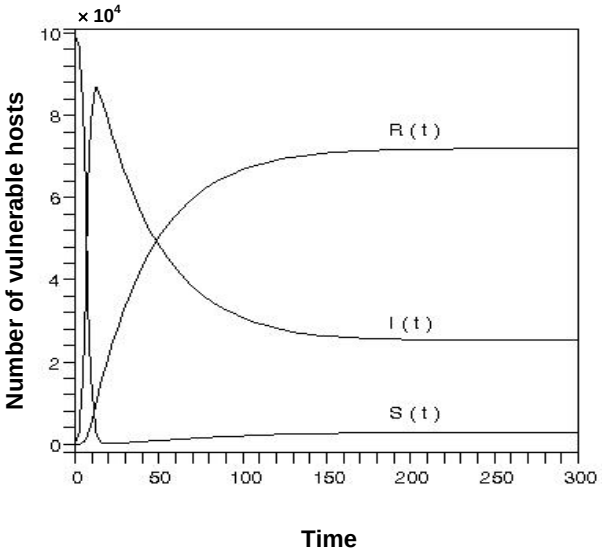


Figure 3.4: SIRS epidemic model; it shows the number of infectious, susceptible, and removed hosts as a function of time

Figure 3.4 compares the number of infectious, susceptible, and removed hosts as a function of time as obtained from equation (3.7). We attempt to solve this model using the numerical capabilities of MAPLE (mathematics software) without finding an explicit function-formula for the number of susceptible, infectious and removed hosts. The graph contains 100,000 hosts and the infection, removal and re-susceptible rates are $\beta = 1.0$ and $\lambda = 0.2$, $\mu = 0.07$ respectively. It shows that the number of infectious hosts is initially exponentially increased up to about 80% of total population and then decreasing the growth of infection population. It is also observed that the infection growth will reach a stable equilibrium after an amount of time passes.

While there is a vast literature covering models in which the "temporary immunity" step is not considered (i.e., SIS models and SIR models), comparatively little work has

been done to understand how the nature of the $R \rightarrow S$ transition affects the dynamics of an epidemic of Internet worms. With regard to the loss of immunity we consider two different types of worm behaviors, depending on parameters: (i) periodic epidemic outbreaks and (ii) one or more extended outbreaks followed by extinction of the epidemic due to stopping spreading of old worms.

We note that instead of acquiring infinite immunity to a specific epidemic, infected hosts in this extended model spend a constant number of time steps in a generalized immune state before returned to the susceptible population. We have to investigate the SIRS model with immunity lasting non-constant time step since hosts can be significantly delayed in the removed state by mechanisms such as a large constant period of temporary immunity.

3.4 Simulation and Analysis

In this section we describe our experimental design and validate the simple epidemic (SIS) model of computer virus introduced by Kephart [5] using the results of our simulation. We also present measurements of worm infections in two different network topologies with random rates at which an infectious node attempts to infect its neighboring nodes and random rates at which it protects itself or remove viruses itself. These experiments provide insight into the characteristics of infection propagation on computer networks and they also serve as the basis for future research work on quarantine of virulent Internet worms.

3.4.1 Random transit stub model without topology constraint

Our experiments have been conducted using a simulation environment that is capable of simulating hundreds of thousands of computing nodes with random network topology and any viral epidemic model. The network topology that is used in this simulation is constructed by Transit Stub model that produces hierarchical graphs in a different way by consisting of interconnected transit and stub domains [17]. A connected random graph is first constructed; each node in that graph represents a transit domain. Each node is then replaced by another connected random graph, which represents the backbone network topology of one transit domain. Next, for each node in each transit domain, a number of connected random graphs that represent the stub domains linked to that node are generated. Finally, certain number of additional edges is created between pairs of nodes, one from a transit domain and one from a stub domain, or one from each of two different stub domains. Clearly, if the random graphs generated are all connected, an entirely connected graph is constructed by the above procedure. Figure 3.5 shows the example of Transit-Stub model.

As shown in figure 3.5, transit domain represents the backbone of the Internet and each backbone node in a transit domain connects to a number of stub domains through nodes, called gateway, in the stub domain. In this experiment we do not consider the topology constraint such as infection delay time when infective messages are able to reach a susceptible node. Instead, the infection process was simulated by varying the connectivity of topology, the number of nodes, and the rate of infection β and cure δ.

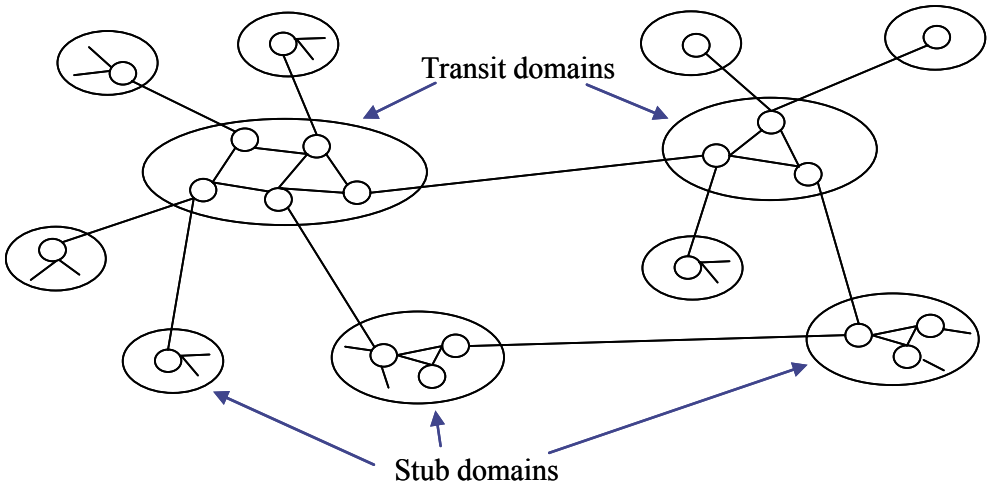


Figure 3.5: Example of Transit-Stub Model

3.4.2 System models

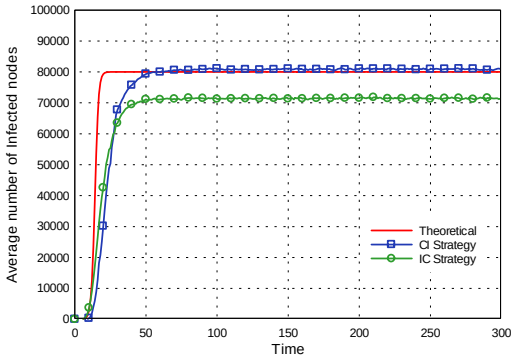
We consider a network with 100,000 nodes and two simulation scenarios. The first one is cured and infection case (*CI strategy*), the same as the one used in the classical simple SIS model, in which an infectious node determines whether it can be cured of infection or not before infecting any of susceptible nodes connected to it. The second one is infection and cured case (*IC strategy*) where an infectious node determines whether it can be cured or not after infecting any of susceptible neighboring nodes. We also analyzed the worm epidemic model with two different infection and cure rates: one is constant infection/cure rate at which an infectious node is equally likely to infect any of other susceptible nodes and to be cured of infection. The other one is variant infection/cure rate at which certain infectious nodes are likely to infect more susceptible nodes than other infectious nodes do. In addition, the infection rate, β, is

associated with each of edges. Similarly, the rate of cure of infection, δ, is related to each node.

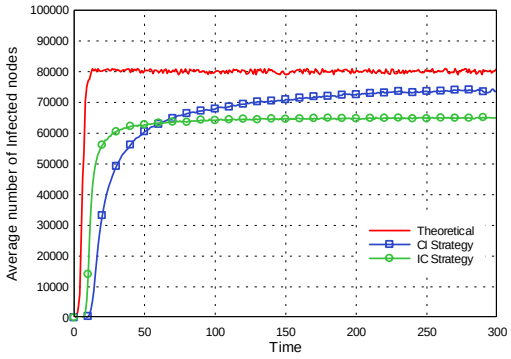
A few assumptions and simplifications were made to ensure feasibility of our experiment. First, a single initially infected node is randomly selected to release worm in each trial and we performed 500 simulation runs using same parameters. Second, a desired random graph has average degree of 5 on each node. It means that the average number of infectious messages that an infected node can generate is five. In other words, an infected node selects on the average five neighboring nodes to infect. Finally, to simulate the model, time is divided into a number of discrete steps, and on each step the population of hosts in each state is altered according to the different rules such as different rate of infection and recovery, which means that hosts move between the *S* (Susceptible states) and *I* (Infectious states) at a certain rate. In addition, relevant data is recorded per unit time and simulation stops when some desired state is reached, such as all nodes are infected or the completion of simulation time.

3.4.3 Initial results

Figure 3.6 shows the total number of infectious nodes averaged across the 500 runs of the two different types of simulation models. Note that in the case of constant rate the number of infectious nodes quickly reaches almost 80% of the total population, and the infection growth slows down after that point. This implies that after almost 50 units of time, the worm may spend much of its time trying to infect nodes that had already been infected.



a) With constant Infection/Cure rate

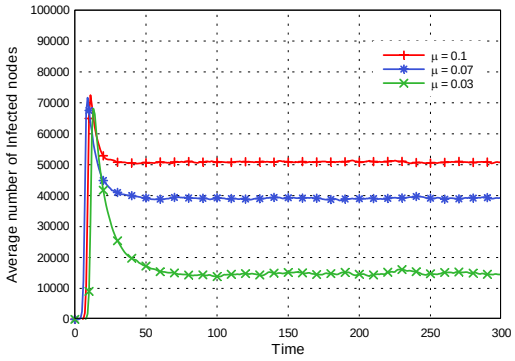


b) With variant Infection/Cure rate

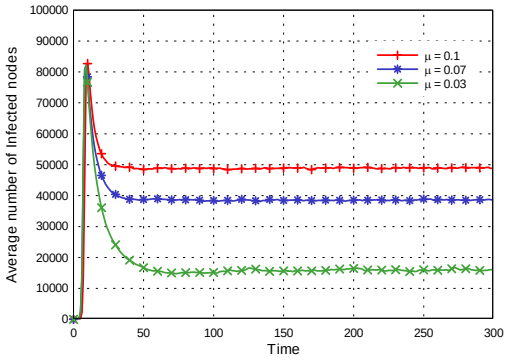
Figure 3.6: Comparison of the average number of infected nodes as a function of time in two different epidemic strategies; N = 100,000, (a) with constant rates, $\beta = 1.0$, and $\delta = 0.2$ (b) with non-constant rates

In addition, in the case of variant infection rate the number of infectious nodes infects other susceptible nodes with different infection rate at each time.

This result is consistent with the results in simulation presented by Kephart [5]. Also a considerable fraction of the nodes in a transit stub network remains uninfected for long periods of time due to their connectivity. Comparing the two different epidemic strategies between constant infection/cure rate and non-constant infection/cure rate, we note that there is a slight difference between these two strategies as shown in figure 3.6. For example, for IC strategy, it takes almost 50 time units for the infection growth to slow down in the model with constant rate in figure 3.6 (a) while it takes 40 time units in the model with variant rate in figure 3.6 (b). For CI strategy, it takes 70 time units to reduce the spread of worm with constant rate while it takes longer time with variant rate.



a) CI strategy with three different μ values

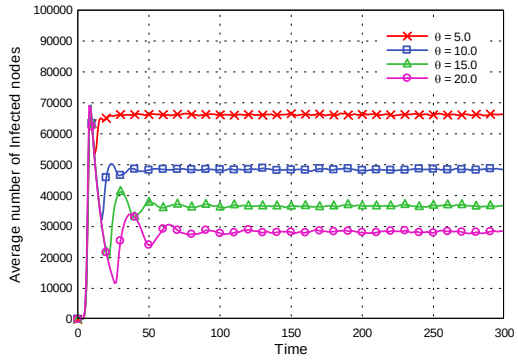


b) IC strategy with three different μ values

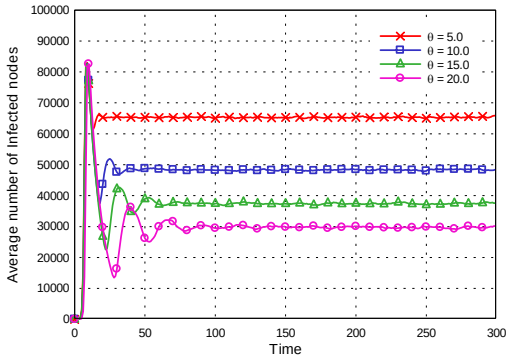
Figure 3.7: Comparison of the number of infectious nodes as a function of time in two different epidemic strategies. All cases are for 100,000 nodes, an average infection rate β of 1.0, a removed rate λ is 0.2, and a re-susceptible rate μ of 0.07

Clearly, it shows that CI strategy has more number of infected nodes than IC strategy does in both the types of simulation models.

Figure 3.7 shows the total number of infectious nodes averaged across the 500 runs of the two different types of simulation models. Note that in both cases the number of infectious nodes increases almost exponentially from time $t = 0$ to $t = 10$, and then the rate of infection growth decreases. This result is consistent with the numerical solution obtained from SIRS mode while there is little difference in comparison of two simulation models. It is also observed that the number of infectious nodes increases as the re-susceptible rate increases in a stable equilibrium stage. It is intuitive that we can expect a lower rate of infectious propagation when the rate of $R \rightarrow S$ transition is lower, because the number of susceptible hosts decrease as the worm propagates.



a) CI Strategy with various temporary immunization times



b) IC Strategy with various temporary immunization times

Figure 3.8: Extinction between two different epidemic models with variation of temporary immunization time θ. All parameters assigned in this experiment are the same as those given in Figure 3.7

Figure 3.8 shows the average number of infectious nodes as a function of time for two simulation models (CI and IC strategies) with various temporary immunization times. As defined in section 3.3.4, we account for the temporary immunization time as the time period to protect the same infectious messages from infected nodes until new worm comes out during a worm infection. In other words, once an infected node has been immune to a worm it will not be again infected by the same worm. However, it might be susceptible to or infected by new worm outbreak which arrives after immunization. We note that the difference of time between two worm outbreaks is referred to as the temporary immunization time. For instance, if an infectious node is cured of infection and takes 10 unit temporary immunization times at time t, it could be susceptible or re-infected by other worms at time $t + 10$.

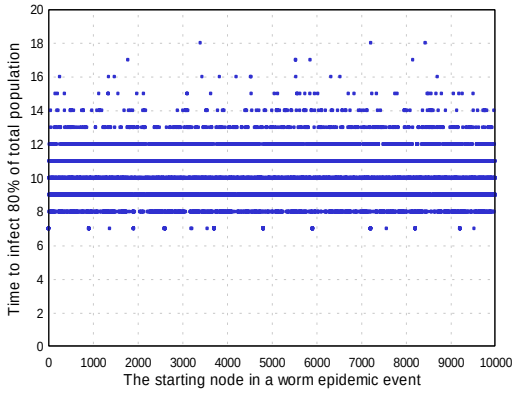


Figure 3.9: Comparison between the total times to infect 80% of total population vs. the starting node in a worm spreading; N = 10,000, infection rate $\beta = 1.0$, and cure rate $\delta = 0.2$

In addition, compared with the two simulation models, there is significantly less variation in the average number of infectious nodes during infection. Results of these experiments show that for a given epidemic model the longer temporary immunization time, the wider will be the variation in infection growth. It is also observed that the infection growth of any type of propagation will reach to a stable equilibrium after an amount of time passes.

In figure 3.9 assuming a particular chosen node as the first infected node, the graph shows the total amount of time to infect at least 80% of the entire population. The fastest time to infect 80% of the population was 7 time units and the longest time was 18. The results of figure 3.9 points out that for a given topology and a given infection model, nodes that are in certain critical locations spreads the worms much faster than

the others. This clearly points to the fact the aggregate infection rate using in many previous works are insufficient for a concrete analysis.

3.5 Worm Propagation with Topology Constraint

We extend our simulation methodology to include a realistic network model and evaluate the impact of topological constraints. After infecting a susceptible node, a worm attempts to infect other susceptible nodes with infection delay time which is the time to find its target nodes; it may attempt to only infect a small number of other susceptible nodes corresponding to network topological criteria, such as connectivity of network.

In addition, we focus on the behavior of Internet worm propagation in response to multiple worm outbreaks. We model the impact of multiple active worms by specifying the temporary immunization time under which an infected node could be immune to the same type of worm after being cured. If many worms are active though they could be removed without infection then the temporary immunization time will be a small, which means that any node in "removed" state as described in our terminology could be susceptible or infectious by new worm outbreaks at time $t + \theta$ although being already cured or recovered at time t. We recall that the time θ is the temporary immunization time which is a measure of how many measurement intervals it will take before the new worm comes out in the Internet. On the other hand, the time θ is long enough for an infected node to be immune to the worm unless new worm conquers the entire network at its high rate of growth.

3.5.1 Network model

In this section, we describe the experimental network model of Internet worm infection using real Internet data set (round trip time (RTT) data) called topology constraint. As the Internet has grown, it is difficult to accurately model the topology and structure of hundreds of thousands of interconnected networks. As a result, it is infeasible to model the full Internet topology for our experiments, so we, instead, look for developing a smaller topology without loss of network characteristics. Our model of the network topology defines the latency (RTT) data for infection delay time and the paths that a worm can follow when propagating. We note that this does not necessarily mean either a fully-interconnected topology or an infection path along every network link. Our interest lies in the network model used to obtain real-time information about the Internet topology measurements around the Internet.

In this study, we obtain network topology data (*e.g.* RTT data and traceroute) from the NLANR Active Measurement Program (AMP) [21]. AMP provides measurements of forward IP path and graphical analysis for Internet usage on undertaking site to site measurement across the HPC networks, which has currently more than 140 active measurement monitors deployed to measure currently round trip times, topology and loss. Given the current network topology, this work is designed to compliment the measurements taken by Abilene network [22] that is an advanced backbone network that connects regional network aggregation points as shown in figure 3.10. It supports the topology measurement services to universities participating in Internet2 and also complements other research networks across the country.

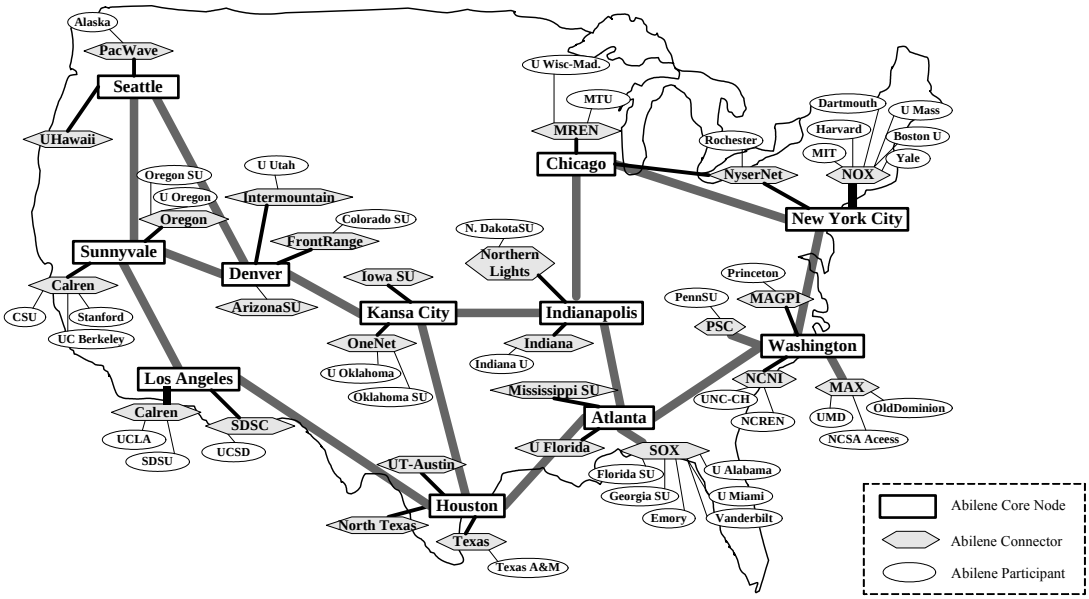


Figure 3.10: The Abilene Network Topology including Abilene core nodes, connectors and some of participants [22]

Our network model consists entirely of 130 active measurement nodes provided by AMP. Each node is connected to the global network shown in Abilene network topology. The global network represents the Abilene core nodes which are connected together and can address and forward packets to each other directly. It is likely that some core nodes contain multiple measurement nodes that frequently communicate among themselves.

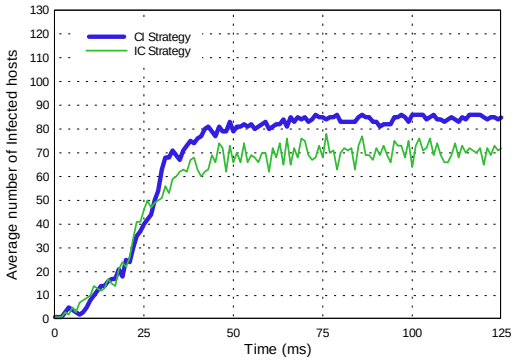
For our experiment, we make a simulation across 500 runs of two different types of simulation model (CI and IC strategies) as described in previous section. In addition to these two types of strategies, we also analyzed two different rules of infection rate: constant infection/cure rate and non-constant dynamic infection/cure rate. Finally, we

note that the round trip time is too fast to capture the rate of population of infected hosts though large RTTs have values greater than 400ms and occasionally as great as 500ms in our observed data. We therefore consider that a valuable mechanism for converting a continuous system time such as RTT into a discrete simulation time, which means that the system time is divided into a number of discrete steps (or time unit), and on each step the population of hosts can be observed according to two states at a certain rate: S (Susceptible states) and I (Infectious states) respectively.

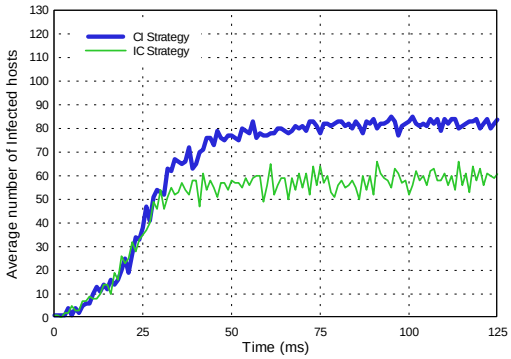
3.5.2 Simulation Results

We simulate a simple, relatively small Internet network model consisting of 130 active monitors connected to each other and located around United States as presented in Abilene network. We examined the performance of the worm epidemic model with topology constraint using the classical epidemic model. For our simulation, we set the discrete interval time into one millisecond (ms), the maximum simulation time for trial to 125 ms corresponding to the maximum RTT value observed from AMP. For Internet worm epidemic model we assume that the infection rate β and the cure rate δ are the same as what are applied in the classical epidemic model.

The total average number of infected hosts over time for two different types of infection/cure rate is shown in figure 3.11. In each case of model we analyze the effectiveness of two different simulation strategies (CI and IC strategies) described in section 3.4.2. Considering the results of figure 3.11 we see that the infection growth stops increasing after the maximum number of infectious hosts reaches almost 80 out



a) With Constant Infection/Cure rate



b) With Non-constant Infection/Cure rate

Figure 3.11: Comparison of average number of infectious nodes as a function of time in two different simulation models

of total population in CI strategy while 60 or 70 nodes in the IC strategy.

Moreover, it takes longer for the rate of growth of worm propagation to be in equilibrium. On the other hand, in classical epidemic model the growth of the curve become quickly stable as shown in figure 3.6. The result of these experiments is undisputed because the growth of infection propagation would be slower if the epidemic model includes topology constraint referred to as infection delay time. Comparing the two different epidemic strategies between constant infection/cure rate and non-constant infection/cure rate, we note that in both case more rapid propagation of worm infection were observed in the CI strategy.

Figure 3.12 shows the result for the comparison of the total 60% infection times as the starting node in Internet worm propagation, obtained from 500 runs of the simulation for Abilene network topology model. We note that some variation exists in the time to infect a large portion of the network.

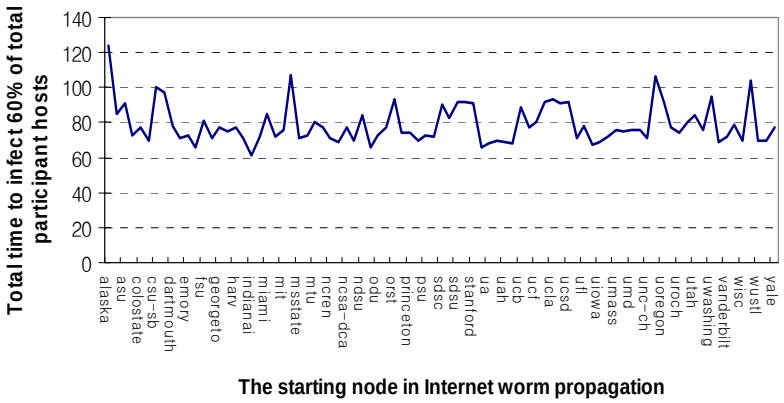


Figure 3.12: Comparison between the total times to infect 60% of total participants vs. the starting node in a worm spreading with constant infection and cure rates

The fastest total time to infect 60% of total participant hosts is 61ms when Indiana University is infected first and the longest was 125ms when University of Alaska was infected first. As shown in Figure 3.10, Indiana University is considered as the center of Abilene backbone network as well as one of Abilene connectors connecting directly to the Abilene network while University of Alaska is just participant located away from the central point of the Abilene network.

We also measured the total number of re-infections that each participant host experienced during a worm infection in order to validate whether the structure of network topology has great influence on infection propagation. Figure 3.13 shows the result for the number of re-infections for each of the 130 participant belonging to Abilene network. We see that some number of the hosts is re-infected much more than others.

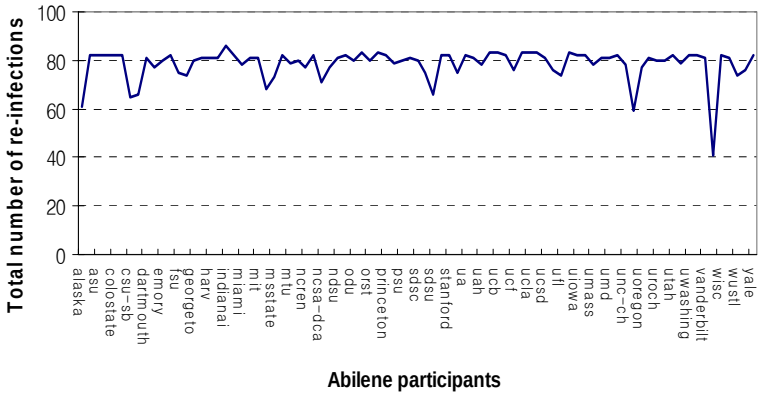


Figure 3.13: Counting the total number of re-infections at each participant host

For example, Indiana University is attacked 86 times on average, while Wisconsin university is attacked 41 times only. It may be pointed out that for a given topology we might slow down the growth of Internet worm infection if we find critical locations where some nodes are more prone to being attacked more than others. Moore et al [18] investigated the containment system using address blacklisting and content filtering to minimize worm propagation in the Internet. The simulation system we have performed could identify addresses to be blacklisted.

3.6 Concluding Remarks

In this chapter we have presented measurements of worm infections in two different network topologies with constant or non-constant infection and cure rates. We extended our simulation methodology to include a real Internet network model and evaluated the impact of topology constraints. As a first step, in this chapter we have described the two classical simple epidemic models and pointed out their limitations. In

addition to those models, we presented an extended model, allowing for loss of immunity that causes recovered hosts to become susceptible again. Our simulation has been conducted using two simulation processes including Infection and Cure per time unit during the spreading of a worm infection. We considered the IC and CI strategies for worm infections and showed that the CI strategy causes more rapid propagation. Furthermore, as part of our ongoing work we are working on accurate analytical models to capture the spread of worms on the Internet. We are also working on the development of effective quarantine techniques using the knowledge of worm propagation.

Chapter 4

Optimal Control of Treatment Costs for Internet Worm

4.1 Introduction

The task of detection and prevention of worms has become more difficult on our existing computing infrastructure. Previous works on worm modeling have been investigated to model and analyze Internet worm propagation [11, 12, 13] and have introduced an analytic model to capture the impact of underlying topology in computer viral propagation [19, 64]. However the control of infectious worm has been one of the most important issues of computer networks. In this chapter, we attempt to investigate a new approach to such optimal control problems with two costs to be minimized.

There is now a large body of work on the cost-benefit analysis of infectious disease control in the public health literature. Goldman and Lightwood [55] presented the cost

optimization problem of minimizing the value of costs incurred from both disease and treatment in biological epidemiological model (SIS). This model uses nonlinear differential equations to provide a qualitative understanding of virus spreading. However, we do not try to find an optimal closed form solution for an economic cost. Instead, we start with the optimal control problem for the cost, say network delay, as derived by Pontryagin's maximum principle [56, 57].

Optimal control can be regarded as one of the possible methodologies of the control system's design. Its role in general theory is unquestionable, but direct practical applications for Internet worm have so far been scarce. Optimal control is well established in some areas, like trajectory planning in the aerospace field and robotics, or model predictive control in chemical industry and furthermore, increasingly many new industrial applications of optimal control have been introduced.

The rest of the chapter is organized as follows. Section 4.2 presents some of important issues in computer networks. Section 4.3 compares our work with several related works. Section 4.4 gives a brief review of the classical epidemic models and a modification of SIS model with treatment to control Internet worm propagation. The analysis of optimization problems is given in Section 4.5. Our numerical solution of optimal controls is also introduced in this section. Section 4.6 shows the simulation results based on a network topology. We conclude the chapter with an outline of our future work in section 4.7.

4.2 Statement of Problems

Infection Cost vs. Treatment Cost

This chapter considers the optimal control problem of minimizing the value of two costs: first is the infection cost which can be interpreted as the node delay due to infection caused by reduced system performance, and increased network delay due to congestion in the network, and the second is treatment cost also referred to as node delay incurred by a certain level of filtering. In other words, both the infection cost and treatment cost are referred to as variations of the nodal processing delay. Our optimal control approach is proposed which enables tradeoffs between the infection cost of compromised systems and the treatment cost of defensive countermeasures, with respect to time.

How many nodes needed to filter?

We have presented a notion of optimal number of nodes to obtain filtering treatment at certain infection rate, providing both a mathematical model of the control factors affecting how many nodes to filter, and collecting empirical data to compare the numerical solution of our analytical model with the results of our simulation. For worm propagation model, we apply the classical SIS model [5, 8]. With this model, the assumptions are that during each period the infected nodes can deal with treatment that will increase their rate of recovery and it has no preventive properties upon recovery. The treatment will also be assumed to exist in discrete time.

When to start a filtering treatment?

"When to start filtering?" presents a serious problem to the security administrator because there are unnecessary treatment cost incurred by forcing the administrator to start filtering as soon as possible and also to delay a filtering treatment so that it takes a more time to fix a security problem. We note that our simulation results can explain when to start the filtering treatment to prevent virulent worms from spreading and suggest effective mechanisms to monitor and defend against the propagation of worms. It should also be noted that we are *not* considering the issue of where to serve a filtering treatment within computer networks. Our model considered only the question of how many nodes to filter and when to obtain filtering services to minimize the infection cost and reduce the spread of worm infection.

4.3 Comparison with Previous Work

In epidemiology research, there exist several deterministic and stochastic models for virus spreading. About ten years ago, Kephart and White [5, 6] presented the Epidemiological model (SIS) to understand and control the prevalence of viruses. This model is based on biological epidemiology and uses nonlinear differential equations to provide a qualitative understanding of virus spreading. They assumed that classical epidemic models are all homogeneous, which means that an infected host is equally likely to infect any of other susceptible hosts. Though at that time the model assumptions were considerably accurate because they considered that infection takes place when hosts share their disks, but such assumptions are no longer valid with the

spreading on the Internet.

The Code Red worm incident of July 2001 has been investigated to model and analyze Internet worm propagation [4]. Based on the classical epidemic models, Zou et al [58] presented mathematical analysis of three worm propagation models under this dynamic quarantine method. The analysis shows that the dynamic quarantine can reduce the speed of worm propagation, which can give us precious time to fight against a worm.

In the spread of a virus on a real network, Wang and Wang [20] investigated the model extending the classical epidemic model by including two specific parameters: infection delay and user vigilance time. The infection delay is a period of time between the arrival of a virus on certain node and further infection from that node. The user vigilance time is the immune time. The simulation study suggests that the most cost effective strategy will need to employ a combination of infection delay and user vigilance.

Kreidl et al [59] presented a feedback control autonomic defense system to improve survivability for a single *host* computer. The survivability objective is expressed as the minimization of a certain mathematical cost that quantifies a tradeoff between the failure cost of a compromised information system and the maintenance cost of ongoing defensive countermeasures. However, their system is mainly about how to detect a worm's process that is already running on a computer and then recover the computer from the worm. It cannot protect a computer from being infected at the first place.

Beattie et al [60] presented a notion of an optimal time to apply security updates, providing both mathematical models of the factors affecting when to patch, and collecting empirical data to give the model practical value. Huerta and Tsimring [61] analyzed the role of contact tracing as a part of the epidemics control strategy in complex networks. The simulation demonstrated that by applying this strategy, a major outbreak can be significantly reduced or even eliminated at a small additional cost.

Kim et al [64] introduced the extension of SIR model to simulate worm propagation in two different network topologies. Whereas in the SIR model once a node is cured after infection it becomes permanently immune, this model allows the immunity to be temporary, since the cured nodes may again become infected, maybe with a different strain of the same worm. The simulation study also showed that time to infect a large portion of the network varied significantly depending on where the infection begins. They extended the simulation methodology to include a real Internet network model and evaluated the impact of topology constraints.

In this chapter we examined several major characteristics of infection, including the variant rate and pattern of infection through the network topology and the rate of treatment at a router during a worm attack. We use a discrete time model and deterministic approximation to describe the spread of Internet worms.

4.4 The SIS Infection Model

We introduce two classical deterministic epidemic models which are the basis of our experimental design. In classical epidemic model, it is defined that a node is called an

infectious node at time *t* if it has been infected by virus before *t*. A node that is vulnerable to virus is called a *susceptible node*.

4.4.1 Infection without Treatment

In particular, the most common mechanism of infection is through contact with an infected node, and the mechanism of recovery is either deterministic or purely stochastic with a certain typical time of recovery. In the classical susceptible-infectious-susceptible (SIS) model [5, 8], a recovered node immediately becomes susceptible again, while in a more complicated susceptible-infectious-removed SIR model [9], cured nodes become immune and effectively excluded from further dynamics. In SIS model, each node stays in one of two states: susceptible or infectious. Each susceptible node becomes an infectious one at a certain rate. At the same time, infectious nodes are cured and become again susceptible at a different rate. In this model, having the infection and being cured, does not confer immunity. Infectious nodes have a constant probability of recovery in each period with treatment. There is no permanent immunity to the infection, so a cured node becomes susceptible again upon recovery. Using the terms defined in table 4.1, the differential equation for the SIS model without treatment is

$$\frac{dI(t)}{dt} = \beta I(t)[N - I(t)] - \delta I(t) \tag{4.1}$$

Notation	Definition
N	Size of total vulnerable population
$S(t)$	Number of susceptible nodes at time t
$I(t)$	Number of infectious nodes at time t
$Q(t)$	Number of treated infectious nodes at time t
β	Infection rate
δ	Curing rate on an infectious node
λ	Treatment rate on an infectious node
C_I	Infection cost
$C(Q(t))$	A function of treatment cost
φ	Adjoint variable
ε	Epidemic threshold

Table 4.1 Notations of SIS Infection Model

We assume that at beginning, t =0, one host is infectious and the other ($N-1$) nodes are all susceptible. Let $S(t) = N - I(t)$ denote the number of susceptible nodes at time t. Replace $I(t)$ in equation (4.1) by $N - S(t)$ and we get

$$\frac{dS(t)}{dt} = -\beta S(t)[N - S(t)] + \delta[N - S(t)] \qquad (4.2)$$

The solution to the equation (4.1) is

$$I(t) = \frac{I_0(\beta N - \delta)}{I_0\beta + (\beta N - \delta - I_0\beta)e^{-(\beta N - \delta)t}} \tag{4.3}$$

We conclude that, as $t \to \infty$,

$$I_\infty = N - \varepsilon \tag{4.4}$$

where $\varepsilon = \delta/\beta$ and I_0 is the initial number of infectious nodes. Therefore, not absolutely all the population gets infected. This shows that each infectious node infects others with an average value of β per unit time. However, the probability that a node becomes infected is not the same for every node because; it is a function of their connectivity and the infection characteristics with a certain cure rate. We note that the probabilities per unit time of infection and of cure are independent. Once a node is cured, it is immediately capable of being re-infected.

4.4.2 Infection with Treatment

We now present the optimization model that takes into account infection and treatment costs. Assume that filtering treatment is available. Infectious nodes can use a level of filtering during each period which will increase the probability of recovery. Higher the level of filtering more will be number of packets processed for infection and hence more will be delay.

Using equation (4.1) and (4.2), this can be expressed as

$$\frac{dI(t)}{dt} = \beta I(t)S(t) - \delta(I(t) - Q(t)) - \lambda Q(t) \tag{4.5}$$

where $\lambda > \delta > 0$, and $I(t) \geq Q(t) \geq 0$

Let $U(t)$ denote the number of untreated infectious nodes at time t. Then equation (4.5) can be defined by

$$\frac{dI(t)}{dt} = \beta I(t)S(t) - \delta U(t) - \lambda Q(t) \tag{4.6}$$

Where $U(t) = I(t) - Q(t)$

If $Q(t) = I(t)$ in each period, then every node which is infected obtains treatment for infection and equation (4.6) becomes

$$\frac{dI(t)}{dt} = \beta I(t)S(t) - \lambda Q(t) \tag{4.7}$$

Note that if the treatment is very effective, then it may be the case that $I = Q$, and the infection no longer is epidemic with full treatment, which is called the equilibrium state. The equilibrium of the model with full treatment is the same as that of the model without treatment if the recovery of infection is also very effective.

4.4.3 Definition

Infection cost

Assume that the worm infection brings about an infection cost, C_I, which can be interpreted as the node delay due to infection caused by reduced system performance. In other words, if hosts get infected, too many 'bad' (infectious) packets would be sent from the infected hosts. Therefore, unnecessary network delay may be incurred from network congestion and buffer overflow. We note that a router does not get infected while every individual host can get infected by computer virus and worm.

The node delay is referred to as router delay since many bad packets causes increased network delay at a router. Once hosts become cured of worm infection a router is also cured. For example, from figure 4.1, if every host in subnet A is cured of infection, a router A becomes cured. However, if one of hosts in subnet A remains infected, the router A is not able to be cured.

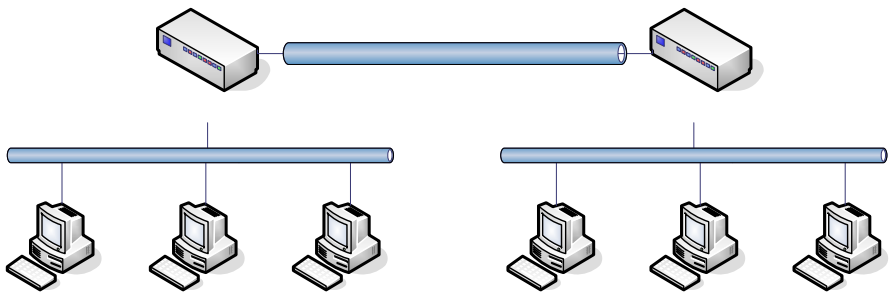


Figure 4.1: Two subnets connected by routers

In this point of view, we apply the classical simple SIS model in which each router stays in one of two states: susceptible or infectious. Each susceptible router becomes an infectious one by receiving infectious packets from infected hosts. At the same time, infectious routers are cured and become again susceptible if no infectious packet received. In this model, having the infection and being cured, does not confer immunity.

Treatment cost

The treatment cost, $C(Q)$, is a function of the total number of infectious nodes treated per period. It can be also referred to as node delay incurred by filtering packets at a router. Similarly, we consider the node delay as the router delay since the router suffers from processing delay due to a certain level of packet filtering mechanism. With the SIS model, the assumptions are that filtering treatment is available and during each period the infected routers can deal with treatment that will increase their rate of recovery and protect their subnet hosts. However, it has no preventive properties upon recovery. The treatment will also be assumed to exist in discrete time.

Once every subnet host becomes cured of infection, filtering treatment will be stopped at that router. In other words, infectious routers have a constant probability of recovery in each period with treatment. There is no permanent immunity to the infection, so a cured router becomes susceptible again upon recovery. It has been determined that the treatment can reduce the level of infection and prevent the prospects of the spread of infection in the future.

Finally, Higher the level of filtering more will be number of packets processed for infection and hence more will be delay.

4.5 The Analysis of Optimization problems

In this section we consider (time dependent) optimal control strategies associated with infection and treatment cost based on classical SIS model. To determine the appropriate number of nodes to filter, we need to develop a mathematical model of the potential costs involved in infection and treatment at a given time. We will develop cost functions that systems administrators can use to help determine an appropriate level of treatment. Goldman and Lightwood [55] introduced the cost optimization problem of minimizing the two cost of disease; a constant per period economic cost of disease and per period cost function of treatment. Our problem is to minimize the total cost of infection and treatment over the finite period time.

Then the our objective functional to be minimized is

$$\int_0^T \left[C_I I(t) + C(Q(t)) \right] dt \tag{4.8}$$

where the control function, $Q(t)$, represents the fraction of total infected nodes consuming treatment (to reduce the number of nodes that may be infectious),

subject to the infection equation:

$$\frac{\partial I(t)}{\partial t} = \beta I(N-I) - \delta(I-Q) - \lambda Q, \qquad I(0) = I_0 \tag{4.9}$$

where β, δ, and λ are know positive constants and I_0 is the known initial infected node.

4.5.1 Necessary Conditions for Optimization

Our objective functional balances the effect of minimizing the cases of implementing the filtering treatments and minimizing the total cost of infection. The necessary conditions that an optimal control variable must satisfy come from Pontryagin's Maximum Principle [56, 57]. In order to derive the necessary conditions we introduce the adjoint variable φ and the Hamiltonian equation, H. This principle converts equation (4.8) and (4.9) into a problem of minimizing a Hamiltonian, H, for the optimization problem:

$$H = C_I I + C(Q) + \varphi[\beta I(N-I) - \delta(I-Q) - \lambda Q] \tag{4.10}$$

Furthermore, there exists an adjoint function, φ(t), such that

$$\frac{\partial \varphi(t)}{\partial t} = -\frac{\partial H}{\partial I} = -C_I - \varphi(N\beta - 2\beta I - \delta) \tag{4.11}$$

where the state problem has initial values $I(0) = I_0$ and the adjoint problem has final values $\varphi(T) = 0$.

We then have to minimize H over $0 \le Q \le I$, that is,

$$\frac{\partial H}{\partial Q} = C'(Q) + \varphi(\delta - \lambda) = 0 \tag{4.12}$$

We denote $C'(Q)$ as the marginal cost of treatment then say $C'(Q) = \alpha\ Q$ where α is equal to the marginal value of an additional unit of the treatment.

Suppose Q^* is an optimal control for the above problem and I^* is the corresponding trajectory so that from equation (4.12) the solution for the optimal control is

$$Q^*(t) = \frac{(\lambda - \delta)}{\alpha}\ \varphi^*(t), \qquad 0 \le t \le T \tag{4.13}$$

Substituting equation (4.13) into equation (4.9) gives

$$\dot{\mathbf{I}}^{\,*} = \beta I(N - I) - \delta I - \frac{(\lambda - \delta)^2}{\alpha}\ \varphi^*, \quad I^*(0) = I_0 \tag{4.14}$$

$$\dot{\varphi}^{\,*} = \varphi^*(2\beta I - N\beta + \delta) - C_I, \qquad \varphi^*(T) = 0, \qquad 0 \le t \le T \tag{4.15}$$

The optimal control is determined by equations (4.13)-(4.15), that is, we must solve equations (4.14) and (4.15) for optimum trajectory and an adjoint variable.

Next, we discuss the numerical solutions of the optimality system and the corresponding optimal control pairs, the parameter choices, and the interpretations.

4.5.2 Numerical Results

In this section, we study numerically an optimal treatment strategy for minimizing the total cost of infection since the full dynamic solution of the control problem is usually very difficult and an explicit function-formula does not exist except for very special cases [55, 56, 62]. The optimal treatment strategy is obtained by solving the optimality system, consisting of two differential equations from the state and adjoint equations presented in previous section.

Figure 4.2 (a) shows that the average number of infected nodes is plotted as a function of time. The graph contains 1000 nodes and the infection, cure and treated rates are $\beta = 1.0$ and $\delta = 0.2$, $\lambda = 0.8$ respectively. For the figure 4.2, we assume that the cost weight factor, C_I, associated with the number of infected nodes $I(t)$ is less or equal to marginal cost of treatment, α, which is associated with a control $Q(t)$. In figure 4.2, the set of the cost weight factors, $C_I = 200$ and $\alpha = 500$, is chosen to illustrate the optimal treatment strategy.

Note that with treatment the number of infected nodes eventually reaches almost 50% of the total population, and the infection growth slows down after that.

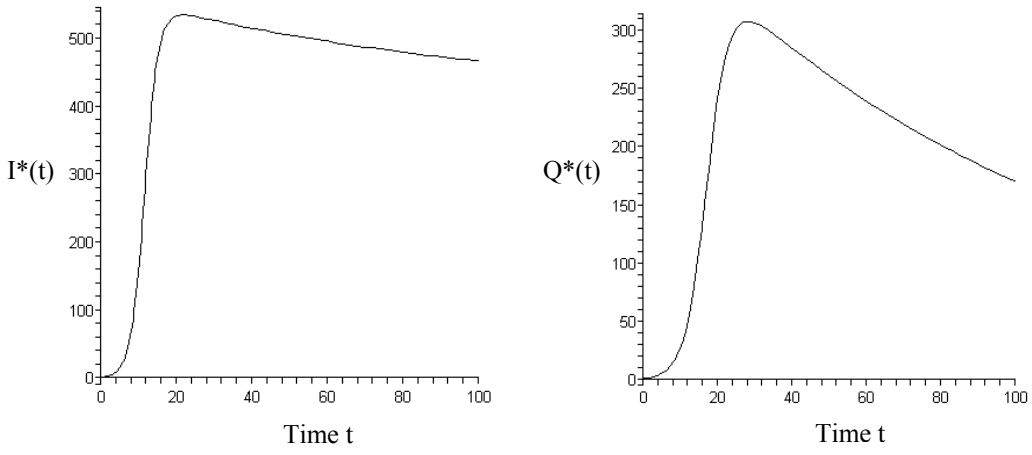


Figure 4.2: Optimal control strategy constructed using Maple

However, without treatment the number of infected nodes reaches almost 80% of the total population. Figure 4.2 (b) shows that for the optimal treatment strategy the control $Q^*(t)$ is plotted as a function of time with the same parameters of figure 4.2 (a).

This is an expected result because the number of infected nodes consuming a filtering treatment, $Q(t)$, is decreased as the number of infected nodes, $I(t)$, is reduced. In conclusion, the control $Q(t)$ that follows this optimal strategy can effectively reduce the spread of infection and minimize the total cost of infection consisting of both infection and treatment costs.

4.6 Simulation Experiments

In this section we describe our experimental design and compare the numerical solution of the optimal control problem with the results of our simulation. We also present measurements of Internet worm infections in two different strategies (with treatment and without treatment) with random rates at which an infectious node attempts to infect its neighboring nodes and random rates at which it cures itself or is treated with a filtering treatment. In addition to measurements of worm infection, we attempt to answer the following question:

1. What is the optimal level of treatment which should be chosen to minimize the total cost of infection?
2. When to start filtering to minimize unnecessary treatment cost?
3. Is there a relationship between treatment cost and infection rate?

These experiments provide insight into the characteristics of infection propagation on computer networks and they also serve as the basis for future research work on quarantine of virulent Internet worms.

4.6.1 Network Model

Our experiments have been conducted using a simulation environment that is capable of simulating thousands of computing nodes with random network topology and a viral epidemic model. The network topology that is used in this simulation is constructed by

Transit Stub model that produces hierarchical graphs in a different way by consisting of interconnected transit and stub domains [17].

We assume that the population consists of N nodes whose connections to one another form a fixed random graph. A node n is said to have a degree $k(n)$ if it is connected to k other nodes. In case of random graphs the degree distribution is Poisson with a certain mean degree $K = \langle k(n) \rangle$. In our simulations with random graph based networks, we typically built networks with average degree $K = 5$ and 1000 nodes. Our simulation is an event-driven simulation which is the most accurate method to simulate the propagation of a worm.

Simulation proceeds in steps of one time unit. During each step, every infectious node I attempts to infect each of its neighbors j with infection rate β. In addition, every infectious node I is subject to a curing attempt with cure rate δ. If the curing of I occurs before the infection attempt, then I does not send out infections to j. In this experiment the infection process was simulated by varying the connectivity of topology, the number of nodes, and the rate of infection β, cure δ and treated λ..

4.6.2 System Model

We consider a network with 1000 routers and a limited buffer is assigned to each router for storing packets which need to be sent and received. A few assumptions and simplifications were made to ensure feasibility of our experiment. First, a single initially infected node is randomly selected to release worm in each trial and we performed 100 simulation runs using the same parameters.

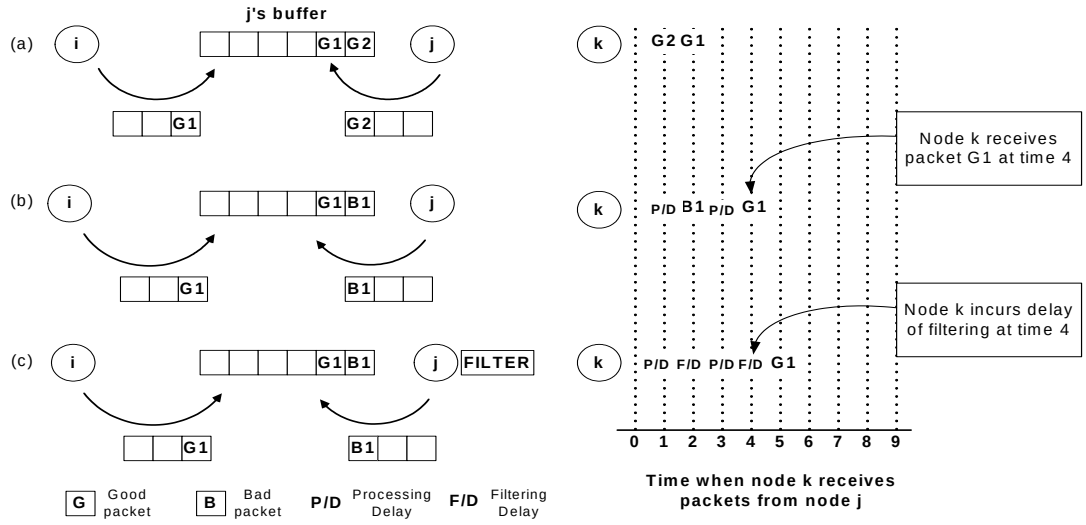


Figure 4.3: Three cases of network delay: (a) No infection occurs (b) Infection without treatment (if node j is infected) (c) Infection with treatment (if node j is infected)

Second, for measuring the network delay we randomly choose some pairs of a source and destination. Each source creates several good (no infection) packets and stores them into its buffer per time unit. We use an all-pair shortest path routing technique to send good packets from source to destination.

Figure 4.3 provides more details about the network delay. Assume that each link has a delay of 1 time unit. A processing delay of 1 is assigned to a host when it is infected, since the system performance is reduced by the infection. We also assign an additional delay of 1 for the packet filtering on a treated node. In case (b) a node *k* which is the next hop from node *j*, receives a packet G1 at time 4 while in case (c) it takes 5 time unit for the same node to receive a packet G1 due to packet filtering system in which only good packets are forwarded to next hop and bad (infectious)

packets are dropped. Finally, to simulate the model, time is divided into a number of discrete steps, and on each step the population of individuals in each state is altered according to the different rules such as different rate of infection and recovery, which means that nodes move between the *S* (Susceptible) and *I* (Infectious) states at a certain rate. In addition, relevant data is recorded per unit time and simulation stops when some desired state is reached, such as the completion of simulation time.

4.6.3 Simulation Results

In this section, we present a set of simulation results that demonstrate the accuracy of our analytical models in describing optimal treatment strategy on random transit-stub networks with infection and treatment costs. Figure 4.4 shows the total number of infectious nodes averaged across the 100 runs of the two different types of simulation models (with treatment and without treatment). Note that without treatment the average number of infectious nodes increased exponentially and eventually reaches almost 80% of the total population, and the infection growth decayed after that. But in the other case (with treatment) the average number of infectious nodes reached almost 60% of the total population and the spread of infection was decreased. For the number of infected nodes obtaining treatment, $Q(t)$, we choose 40 % of total infected nodes at each time. It notes that a certain level of treatment for infection can effectively reduce the spread of worm infection.

Figure 4.5 shows that for the optimal treatment strategy the control variable $Q^*(t)$ and state variable $I^*(t)$ are plotted as a function of time with the same parameters of

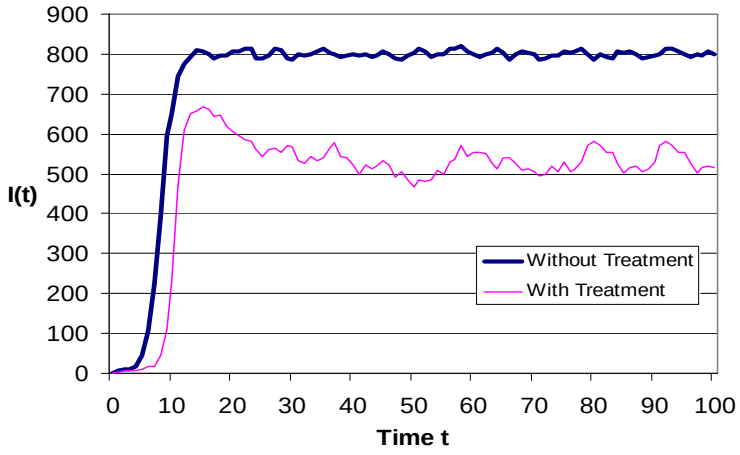


Figure 4.4: Comparison of the average number of infectious nodes as a function of time in two different epidemic strategies. For the cases above we used 1000 nodes, an average infection rate β of 1.0, a cure rate δ of 0.2, and a treatment rate λ of 0.8

figure 4.2. These two curves show that simulation results are consistent with our numerical results of the optimization problem as described in section 4.5.2. It has been determined that higher the level of a filtering treatment, which means increasing the treatment rate, less will number of infectious nodes spread for infection.

However, since a filtering treatment presumably bears a significant cost, an optimal choice of the treatment rate, λ, for a given infection rate is an important issue. From figure 4.5 we conclude that for a given infection and cure rates if one follows the optimal control $Q^*(t)$ trajectory then the spread of infection can be significantly reduced or even eliminated at a small additional cost. This implies that the optimal treatment strategy derives its value from reducing the current infection rate and from reducing the prospects of the spread of infection.

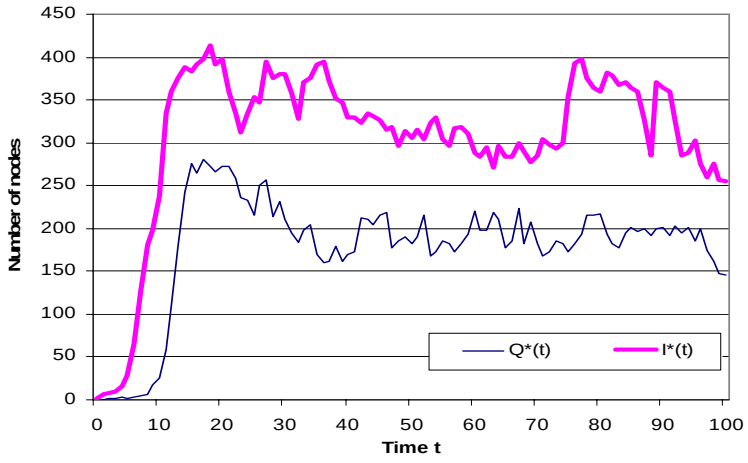


Figure 4.5: Optimal control $Q^*(t)$ and $I^*(t)$ are plotted as a function of time for N = 1000, $\beta = 1.0$, $\delta = 0.2$, $\lambda = 0.8$

	Q(t) = 0%		**Q(t)** = 10%		**Q(t)** = 20%		**Q(t)** = 30%		...	**Q(t)** = 90%		**Q(t)** =100%	
t	**I(t)**	**C(t)**	**I(t)**	**C(t)**	**I(t)**	**C(t)**	**I(t)**	**C(t)**		**I(t)**	**C(t)**	**I(t)**	**C(t)**
0	1	**1**	1	**1**	1	**1**	1	**1**		1	**1**	1	**2**
1	6	**6**	4	**4**	4	**5**	5	**6**		5	**9**	5	**10**
2	9	**9**	6	**6**	7	**8**	8	**10**		7	**10**	7	**14**
3	10	**10**	9	**9**	10	**12**	8	**10**		8	**13**	9	**18**
4	18	**18**	12	**13**	17	**20**	9	**11**		9	**14**	8	**16**
5	46	**46**	28	**30**	44	**52**	9	**11**		11	**16**	10	**20**
6	108	**108**	78	**85**	103	**123**	19	**24**		13	**26**	15	**30**
7	226	**226**	181	**199**	210	**252**	48	**62**		33	**65**	34	**68**
8	394	**394**	344	**378**	368	**441**	139	**180**		67	**112**	62	**134**
9	596	**596**	573	**630**	524	**628**	279	**362**		78	**140**	100	**200**
10	752	**752**	737	**810**	666	**799**	454	**590**		122	**219**	111	**222**
...													
96	898	**898**	717	**788**	567	**680**	417	**542**		512	**1003**	556	**1112**
97	898	**898**	721	**792**	572	**686**	419	**544**		487	**895**	623	**1123**
98	896	**896**	721	**792**	551	**661**	415	**539**		468	**925**	687	**1374**
99	896	**896**	736	**809**	542	**650**	423	**549**		645	**1192**	765	**1530**
100	897	**897**	736	**809**	516	**619**	425	**552**		687	**1212**	825	**1650**

Table 4.2 Determination of optimal control $Q^*(t)$ and $I^*(t)$

Table 4.2 shows how to obtain optimal control $Q^*(t)$ and $I^*(t)$ from our simulation results. First, we perform 100 simulation runs using the same parameters and select a minimum value of total infection cost, $C(t)$, at each time t. From table 4.2 one curve line indicates the optimal trajectory for total infection cost. In next step we can determine the control variable $Q(t)$ and state variable $I(t)$ according to the total infection cost $C(t)$ at each time. For example, from table 4.2 we select a value of 4 for a minimum infection cost $C(t)$ at $t = 1$, then we can find that the number of infected nodes, I ($t = 1$), is 4 and the number of infected nodes getting treatment, Q ($t = 1$), is 10 % of I ($t = 1$).

The average delay of good packets over infection rates for two different types of simulation models is shown in figure 4.6. For this experiment, we make a simulation across 100 runs as described in figure 4.4. All parameters assigned in this experiment are the same as those given in figure 4.5. In addition to these two types of strategies, we also analyzed the average delay of the good packets generated on only a set of source nodes. As seen from the figure 4.6, without treatment the average delay increases exponentially as the epidemic becomes saturated while with treatment the average delay is not drastically increased with the same increase of infection rates. It implies that a filtering treatment of worm infection is very effective for reducing the spread of infection and minimizing the total infection costs as referred to network delay. It is further observed that the treatment of worm propagation in its early stage is not an optimal time to minimize the total cost of infection since there is little difference in the average delay between the two cases (with treatment and without treatment) until the infection rate reaches 0.4.

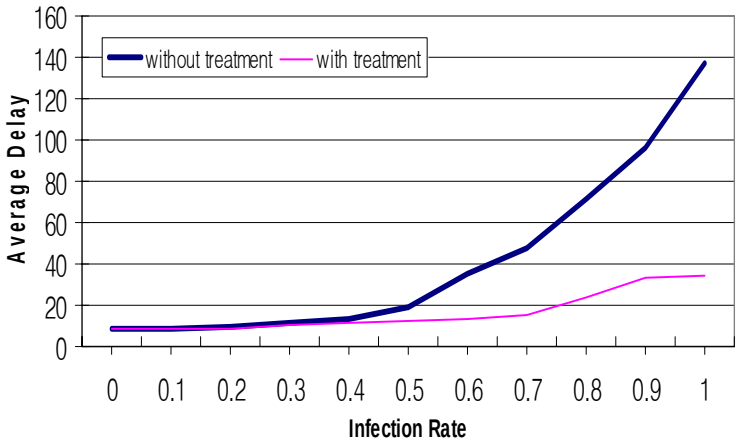


Figure 4.6: Comparison of the average delay as infection rates in two different epidemic strategies

For example, figure 4.6 shows that the filtering treatment of worm infection could be started as late as when the infection rate is 0.4 without having more than a marginal effect on the total cost of infection.

There is a security issue for the security administrator to find an appropriate time to start filtering since there are unnecessary treatment costs incurred by filtering too early and also delaying a filtering treatment. In particular, many security administrators feel that it is imperative to start a filtering treatment *immediately*. This, however, is just representing those sites that have very high risk of penetration and have ample resources to do an entire treatment. Our intent in this study is to provide guidelines to those who do not have sufficient resources to immediately detect and filter everything, and must choose where to allocate scarce security resources. We have used the empirical data to arrive at concrete recommendations for delaying a filtering treatment until there is assurance that the treatment is not likely to cause unnecessary costs.

4.7 Concluding Remarks

In this chapter we have presented the optimal control problem of minimizing the total cost of infection which can be interpreted as the network delay incurred by both infection and treatment. This chapter focuses on the application of optimal control theory to minimizing the value of two costs. We have derived the necessary conditions for our optimal control problem which is solved numerically.

We also validated the numerical solution with our simulation results. It has been determined that by applying this optimal control strategy, one can very effectively reduce the spread of infection and minimize the total cost of infection for the case of random graphs. However, we could wait for a certain ratio of the total number of nodes to be infected before starting treatment, and this would prevent an unnecessary network delay which would happen if treatment were started earlier.

This chapter considered only the question of how many nodes to filter and when to obtain filtering services to minimize the infection cost and reduce the spread of worm infection. We are also working on the development of effective quarantine techniques using the knowledge of cost optimization problem for worm infection.

Chapter 5

Conclusion and Future work

In this chapter, dissertation contributions as well as future work will be summarized. First, a brief introduction about the problem we are trying to solve is presented. Then, an organization of dissertation contribution is discussed followed by the appropriate category including a brief summary of each chapter. At the end, future work directions are presented and discussed.

5.1 Problems

Many early Internet protocols were designed without a fundamentally secure infrastructure and hence vulnerable to Internet attacks such as denial of service (DoS) attacks and worms. DoS attacks attempt to consume the resources of a remote host or network, thereby denying or degrading service to legitimate users. In fact, many attacks can be launched readily from anywhere in the world masquerading the location of the

attacker. *Network forensics* is an emerging area wherein the source or the cause of the attacker is determined using IDS tools. The problem of finding the source(s) of attack(s) is called *trace back problem*. Lately, Internet worms have become a major problem for the security of computer networks, causing considerable amount of resources and time to be spent recovering from the disruption of systems. In addition to breaking down victims, these worms create a large amount of unnecessary network data traffic that results in network congestion, thereby affecting the entire network. However, the task of detection and prevention of worms has become more difficult on our existing computing infrastructure.

5.2 Organization of Proposals

In this dissertation, we have analyzed some security issues in Internet attack through an investigation of Internet worm propagation models and an identification of intrusion source. Also we attempt to solve an optimal control problem of minimizing the total cost of infection in terms of network delay. This analysis of security issues took two categories. First, to develop appropriate tools for thwarting quick spread of worms, we are trying to understand the behavior of the worm propagation with the aid of epidemiological models and to provide mathematical models of control factors that influence Internet worm propagation called *Infection Pattern*. The second category is motivated by the fact that we have to react to Intrusions, be a worm based intrusion or others. Reacting to intrusions has two kinds of actions; one is intrusion source

identification, the other is network defense with an optimal level of treatment. Chapter 2 and 4 describe about the latter category while chapter 3 is for the former category.

5.3 Infection Patterns

In chapter 3, we presented the classical SIS model and a modification of SIR model of Kermack-Mckendrick to understand the behavior of the worm propagation with the aid of epidemiological models. The analytical models that we provide are useful in determining the rate of spread and time required to infect a majority of the nodes in the network. We also present measurements of worm infections in two different network topologies and in one of the topologies we use the round-trip time collected by using the NLANR Active Measurement Program (AMP). Whereas in the SIR model once a node is cured after infection it becomes permanently immune, our modification allows this immunity to be temporary, since the cured nodes may again become susceptible or infected, maybe with a different strain of the same worm. Our simulation results on large Internet like topologies show that in a fairly small amount of time, 80% of the network nodes is infected. For example, on the Abilene Internet2 topology using real link delays we have shown that the worm can spread and infect 80% of the nodes in about 30ms. The simulation study also shows that time to infect large portions of the network vary significantly depending on where the infection begins. This information could be usefully employed to choose hosts for quarantine to delay worm propagation to the rest of the network.

5.4 Reacting to Intrusions

In chapter 2, we have introduced several trace back techniques and defined a trace back problem more formally. Our goal in this work is to determine the sources of intrusions or at least the routers closest to the intruders using a minimal amount of network resources. We have developed a novel algorithm to decompose a network into connected components and using high traffic routers on the connected components, we construct a terminal network. A centroid decomposition technique is applied on the terminal network. Based on the position of the victim in the network, our scheme selects only a small fraction of routers to monitor the traffic to identify packets that bear the signatures of the attack packets. From the information provided by these chosen routers, the network is pruned and another set of routers is chosen to identify the source of attack, until the source router is detected. The trace back can be completed in $O(\log n)$ steps, where n is the number of terminal nodes (routers) in the terminal network. Contribution of our work is to identify the set of routers that are requested to log, mark, or authenticate depending upon the type of attack. The number of routers identified for this task will be kept at a minimum yet sufficient to reduce the burden on the routers.

In chapter 4, we have presented an optimization model that takes into account the infection and treatment costs. We have two variables that we need to work with: delay caused by filtering of worms at routers, and the delay due to worms' excessive amount of network traffic. On one hand filtering causes delays at routers and on the other worm's packets overload the buffer at routers and this in turn causes additional delays

for genuine packets. Furthermore, we defined the objective of minimizing the total cost of infection and derived the necessary conditions for our cost optimization problem which is solved numerically. Using this model we can determine the level of treatment to be applied for a given rate of infection spread. We have devised a technique again borrowing from epidemic models to determine the optimal start point for filtering and optimal number of nodes in the network that should perform the filtering. The simulation study shows the optimal level of treatment which should be chosen to minimize the total cost of infection and to reduce the current infection rate, providing both a mathematical model of the control factors affecting how many nodes needed to filter, and collecting empirical data to compare the numerical solution of our analytical model with the results of our simulation. Finally, we noted that our simulation results can explain when to start the filtering treatment to reduce an unnecessary network delay incurred by filtering too early and also delaying a filtering.

5.5 Future Work

For future work in this area, we intend to develop an effective quarantine method using the knowledge of worm propagation and of cost optimization problem for worm infection. Due to the fast spreading nature and great damage of Internet worms, it is necessary to implement automatic mitigation such as dynamic quarantine. This information could be usefully employed to choose hosts for quarantine to reduce the prospects of the spread of infection in the future. In addition to development of quarantine techniques, we attempt to answer several questions:

- What is effective size of quarantine?
- How can we detect and monitor an unknown (zero-day) worm?
- How can we defend against the spread of unknown worms effectively?

It will be the most important task of dynamic quarantine defenses.

Bibliography

[1] Ed Tiley. *Personal Computer Security*. IDG Books Worldwide, Inc., Foster City, CA, 1996.

[2] Computer Emergency Response Team. smurf IP Denial-of-Service Attacks. *CERT Advisory CA-1998-01*, 1998.

[3] Jose Nazario, *Defense and Detection Strategies against Internet Worms*. Artech house, Inc., MA, 2004.

[4] D. Moore, C. Shannon, and J. Brown. Code-Red: a case study on the spread and victims of an Internet worm. *Proceedings of the 2nd Internet Measurement Workshop*, pages 273–284, November 2002.

[5] J. O. Kephart and S. R. White. Directed-graph epidemiological models of computer viruses. *Proceedings of the IEEE Symposium on Security and Privacy*, pages 343–361, 1991.

[6] J. O. Kephart and S. R. White. Measuring and Modeling Computer Virus Prevalence. *Proceedings of the IEEE Symposium on Security and Privacy*, pages 2-15, 1993.

[7] E. H. Spafford. The Internet worm incident. In *ESEC'89 2nd European Software Engineering Conference*, United Kingdom, pages 446-468, 1989.

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

[9] J. C. Frauenthal. *Mathematical Modeling in Epidemiology*. Springer-Verlag, New York, 1980.

[10] L. Edelstein-Keshet. *Mathematical Models in Biology*. Random House, New York, 1988.

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

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

[13] Z. Chen and L. Gao and K. Kwiat. Modeling the Spread of Active Worms. *IEEE INFOCOM*, 3:1890-1900, 2003.

[14] N. Weaver. Warhol Worms: The Potential for Very Fast Internet Plagues. 2001, http://www.cs.berkeley.edu/~nweaver/ warhol.html.

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

[16] eEye Digital Security. Press. New Critical Flaw in Microsoft Windows® RPC. http://www.eeye.com/html/Press/PR20030910.html.

[17] Ellen W. Zegura, Ken Calvert and S. Bhattacharjee. How to Model an Internetwork. *Proceedings of IEEE INFOCOM96*, 2: 594-602, CA, 1996.

[18] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. *Proceedings of IEEE INFOCOM*, 3:1901-1910, 2003.

[19] Yang Wang, Deepayan Chakrabati, Chenxi Wang, and Christos Faloutsos. Epidemic spreading in real networks: an Eigenvalue viewpoint. *Proceedings of 22nd International Symposium on Reliable Distributed Systems (SRDS03)*, pages 25-34, 2003.

[20] Yang Wang, Chenxi Wang. Modeling the Effects of Timing Parameters on Virus Propagation. *Proceedings of the 2003 ACM workshop on Rapid Malcode (WORM 03)*, pages 61-66, 2003.

[21] NLANR Active Measurement Program (AMP), http://watt. nlanr.net/

[22] Abilene backbone network, http://abilene.internet2.edu/

[23] L. Garber. Denial-of-service attacks rip the Internet. *IEEE Computers*, 33(4):12-17, 2000.

[24] NightAxis and R. F. Puppy. Purgatory 101: Learning to cope with the SYNs of the Internet. 2000. Some practical approaches to introducing accountability and responsibility on the public internet, http://packetstorm.securify.com/papers/contest/RFP.doc

[25] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on TCP. *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 208–223, 1997.

[26] G. Vigna and R. Kemmerer. NetSTAT: A Network-based Intrusion Detection System. *Journal of Computer Security*, 7(1):37-71, 1999.

[27] C. Meadows. A formal framework and evaluation method for network denial of service. *Proceedings of IEEE the Computer Security Foundations Workshop*, pages 4-13, 1999.

[28] O. Spatscheck and L. Peterson. Defending against denial of service attacks in Scout. *Proceedings of the 3rd USENIX/ACM Symposium on Operating Systems Design and Implementation (OSDI'99)*, pages 59–72, 1999.

[29] Computer Emergency Response Team (CERT). Denial-of-service developments. *CERT Advisory CA-2000-01*, 2000.

[30] R. Morris. A weakness in the 4.2BSD Unix TCP/IP software. *Computer Science Technical Report 117, AT&T Bell Labs*, 1985.

[31] J. Postel. Internet protocol. *RFC 791*, 1981

[32] Wenke Lee and Sal Stolfo. Data Mining Approaches for Intrusion Detection. *Proceedings of the 7th USENIX Security Symposium (SECURITY '98)*, pages 79-94, 1998.

[33] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422-426, 1970.

[34] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. *Computer Communication Review*, 30(4):295-306, 2000.

[35] D.X. Song and A. Perrig. Advanced and Authenticated Marking Scheme for IP Traceback. *Proceedings of IEEE INFOCOM*, 2:878-886, 2001.

[36] K. Park and H. Lee. On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack. *Proceedings of IEEE INFOCOM*, 1:338-347, 2001.

[37] CERT/CC, SANS. Institute, and CERIAS. Consensus roadmap for defeating distributed denial of service attacks. Feb. 2000. A Project of the Partnership for Critical Infrastructure Security, http://www.sans.org/ddos roadmap.htm

[38] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. *RFC 2827*, 2000.

[39] Hal Burch and Bill Cheswick. Tracing Anonymous Packets to Their Approximate Source. *Proceedings of the 14th USENIX Systems Administration Conference* (*LISA 2000*), pages 319-327, 2000.

[40] S. M. Bellovin. ICMP Traceback Messages. *IETF Internet Draft*, March 2001.

[41] A. Mankin, D. Massey, C. Wu, S. F. Wu, and L. Zhang. On Design and Evaluation of Intention-Driven ICMP Traceback. *Proceedings of IEEE International Conference on Computer Communications and Networks*, pages 159-165, 2001.

[42] E. Zwicky, S. Cooper, D. Chapman, and D. Ru. *Building Internet Firewalls*. O'Reilly & Associates, Inc., 2nd edition, 2000.

[43] H.Y. Chang, R. Narayan, C. Sargor, F. Jou, S.F. Wu, B.M. Vetter, F. Gong, X. Wang, M. Brown, and J.J. Yuill. DECIDUOUS: Decentralized Source Identification for Network-Based Intrusions. *Proceeding of 6th IFIP/IEEE International Symposium on Integrated Network Management*, pages 701-714, 1999.

[44] H.Y. Chang, P. Chen, A. Hayatnagarkar, R. Narayan, P. Sheth, N. Vo, C. L. Wu, S.F. Wu, L. Zhang, X. Zhang, F. Gong, F. Jou, C. Sargor, X. Wu. Design and Implementation of A Real-Time Decentralized Source Identification System for Untrusted IP Packets. *Proceedings of the DARPA Information Survivability Conference & Exposition*, 2:1100-1111, 2000.

[45] R. Stone. CenterTrack: An IP Overlay Network for Tracking DoS Floods. *Proceedings of 9th USENIX Security Symposium*, pages 199-212, 2000.

[46] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer. Single-Packet IP Traceback. *IEEE/ACM Transactions on Networking*, 10(6):721-734, 2002.

[47] Vern Paxson. End-to-end Internet path dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277-292, 1999.

[48] Craig Labovitz, G. R. Malan, and F. Jahanian. Internet Routing Instability. *IEEE/ACM Transactions on Networking*, 6(5):515-527, 1998.

[49] Bilal Chinoy. Dynamics of Internet Routing Information. *ACM SIGCOMM Computer Communications Review*, 23(4):45-52, 1993.

[50] R. Govindan and A. Reddy. An Analysis of Internet Inter-Domain Topology and Route Stability. *Proceedings of IEEE INFOCOM'97*, 2:850-857, 1997.

[51] A. Shaikh and L. Kalampoukas. Routing Stability in Congested Networks: Experimentation and Analysis. *ACM SIGCOMM Computer Communications Review*, 30(4):163-174, 2000.

[52] F. Harary. *Graph Theory*. Addison-Wesley, Inc., MA, 1969.

[53] G. S. Brodal, R. Fagerberg, C. N. S. Pedersen, and A. Östlin. The complexity of constructing evolutionary trees using experiments. *Proceedings of 28th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2076:140-151, 2001.

[54] SANS Institute. Egress filtering v 0.2. *Global Incident Analysis Center, Special Notice*, Feb. 2000. Available at `http://www. sans.org/y2k/egress.htm`.

[55] Steven M. Goldman and J. Lightwood. Cost Optimization in the SIS Model of Infectious Disease with Treatment. *The B.E. Journals in Economic Analysis and Policy*, 2(1):1-22, 2002.

[56] G. Knowles. *An Introduction to Applied Optimal Control*. New York, Academic Press, 1981.

[57] E.B. Lee and L. Markus, *Foundations of Optimal Control Theory*, New York, John Wiley & Sons, Inc., 1967

[58] C. C. Zou, W. Gong, and D. Towsley. Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense. *Proceedings of the ACM/CCS workshop on Rapid Malcode (WORM'03)*, pages 51-60, 2003.

[59] O.P. Kreidl and T.M. Frazier. Feedback Control Applied to Survivability: A Host-Based Autonomic Defense System. *IEEE Transactions on Reliability*, 53(1):148-166, 2004.

[60] S. Beattie, S. Arnold, C. Cowan, P. Wagle, and C. Wright. Timing the Application of Security Patches for Optimal Uptime. *Proceedings of LISA '02: 16th Systems Administration Conference*, pages 233-242, 2002.

[61] R. Huerta and L.S. Tsimring. Contact tracing and epidemics control in social networks. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 66(52):056115/1-056115/4, 2002.

[62] M. D. Intriligator. *Mathematical Optimization and Economic Theory*, Englewood Cliffs, N.J., Prentice-Hall, 1971.

[63] J. Kim, S. Radhakrishnan, and S. K. Dhall. On Intrusion Source Identification. *Proceedings of the 2nd IASTED International Conference on Communications, Internet, and Information Technology*, pages 7-12, 2003.

[64] J. Kim, S. Radhakrishnan, and S. K. Dhall. Measurement and Analysis of Worm Propagation on Internet Network Topology. *Proceedings of 13th International Conference on Computer Communications and Networks* (*IEEE ICCCN2004*), pages 495-500, 2004.

APPENDIX A

SIS Epidemic Model

The SIS model (where removals are ignored) is described by:

$$\frac{dS}{dt} = -kSJ + \lambda J \qquad \text{and} \qquad \frac{dJ}{dt} = kSJ - \lambda J$$

The initial conditions are: S(0) = S0 and J(0) = J0

To get (easily) an analytic solution, we observe that: S + J = N. Thus, S = N – J.

Substituting to the second differential equation we obtain:

$$\frac{dJ}{dt} = k(N-J)J - \lambda J \text{ which simplifies into } \frac{dJ}{dt} = (kN-\lambda)J - kJ^2$$

The differential equation could be solved using Maple as:

Note: Since I is used (reserved word in Maple) to denote: I = sqrt(-1); we use the variable J to denote the number of infected in the fixed population.

```
> restart;
> dsolve({diff(J(t),t)= k*(N-J(t))*J(t)-lambda*J(t), J(0) = J0}, J(t));
```

$$J(t) = \frac{J0\,(k\,N - \lambda)}{k\,J0 + \mathbf{e}^{(-(k\,N - \lambda)\,t)}\,k\,N - \mathbf{e}^{(-(k\,N - \lambda)\,t)}\,\lambda - \mathbf{e}^{(-(k\,N - \lambda)\,t)}\,k\,J0}$$

```
> simplify(%);
```

$$J(t) = \frac{J0\,(k\,N - \lambda)}{k\,J0 + \mathbf{e}^{(-(k\,N - \lambda)\,t)}\,k\,N - \mathbf{e}^{(-(k\,N - \lambda)\,t)}\,\lambda - \mathbf{e}^{(-(k\,N - \lambda)\,t)}\,k\,J0}$$

> collect(%,exp);

$$J(t) = \frac{J0\,(k\,N - \lambda)}{(k\,N - \lambda - k\,J0)\,\mathbf{e}^{(-(k\,N - \lambda)\,t)} + k\,J0}$$

> with(plots):

Warning, the name changecoords has been redefined

> N:= 100: S0:= 99: J0:=1: k:=0.8: lambda:=0.2:

> J:= unapply(J0*(k*N-lambda)/((k*N-lambda-J0*k)*exp(-(k*N-lambda)*t)+J0*k), t);

$$J := t \rightarrow \frac{79.8}{79.0\,\mathbf{e}^{(-79.8\,t)} + 0.8}$$

> plot([t,J(t), t = 0..10],t= 0..10, tickmarks=[5,5], labels=[`t`,`J(t)`]);

APPENDIX B

SIRS Epidemic Model

The SIRS model of infections was derived by Kermack and McKendrick (1927).

```
> restart;

with(DEtools):

with(linalg):

with(plots):
```

Warning, the name changecoords has been redefined

Define SIR equations, parameters and initial conditions

beta = rate from S -> I

lamda = rate from I -> R

mu = rate from R -> S (b=0 gives the SIR model)

```
> beta:= 0.07: lamda:= 0.02: mu:= 0.007:

        eqns := diff(S(t),t)=-beta*S(t)*J(t)+mu*R(t),

                diff(J(t),t)= beta*S(t)*J(t)-lamda*J(t),

                diff(R(t),t) = lamda*J(t)-mu*R(t);

        inits:= S(0)=9.9, J(0)=0.1, R(0)=0.0;
```

$$eqns := \frac{d}{dt}S(t) = -0.07\,S(t)\,J(t) + 0.007\,R(t),\ \frac{d}{dt}J(t) = 0.07\,S(t)\,J(t) - 0.02\,J(t),\ \frac{d}{dt}R(t) = 0.02\,J(t) - 0.007\,R(t)$$

inits := $S(0) = 9.9$, $J(0) = 0.1$, $R(0) = 0.$

Generate a numerical solution

```
> soln := dsolve({eqns,inits},{S(t),J(t),R(t)}, type=numeric, output=listprocedure):
```

Generate string for title

```
> rr := convert(beta,string):  aa := convert(lamda,string):
  bb := convert(mu,string):
  code := cat(`SIRS Model with rates beta = `,rr,`, lamda = `,aa,`, mu = `,bb);
```

code := *SIRS Model with rates beta = .7e-1, lamda = .2e-1, mu = .7e-2*

Define functions for the susceptibles - S(t), the infecteds - J(t) and the removed - R(t)

```
> s := subs(soln,S(t)): j:=subs(soln,J(t)): r := subs(soln,R(t));
```

r := **proc**(*t*) ... **end proc**;

```
> G:=plot({s,j,r},0..300,title="",color=black):
> T:=textplot({[200, 7.7, ` R ( t )`],[200, 0.8, ` S ( t )`],[200, 3, `I ( t )`]}):
> display({G,T});
```

APPENDIX C

Optimal Control Problem

Suppose Q^* is an optimal control for the optimization problem and I^* is the corresponding trajectory. The optimal control is determined by two equations below, that is, we must solve these equations for optimum trajectory and an adjoint variable.

$$\dot{\mathbf{I}}^* = \beta I(N - I) - \delta I - \frac{(\lambda - \delta)^2}{\alpha}\varphi^*, \quad \mathrm{I}^*(0) = \mathrm{I}_0$$

$$\dot{\varphi}^* = \varphi^*(2\beta I - N\beta + \delta) - C_I, \qquad \varphi^*(T) = 0, \qquad 0 \le t \le T$$

Next, we solve the numerical solutions of the optimality system and the corresponding optimal control pairs using Maple.

```
> restart:
> with(DEtools):
> with(plots):
```

Warning, the name changecoords has been redefined

```
> alpha:=500: beta:=1.0: delta:=0.2: lambda:=0.8: Cd:=200: N:=1000:
> CM_Model:=diff(J(t),t)=J(t)*(N*beta-delta-beta*J(t))-psi(t)*(lambda-delta)^2/alpha,
diff(psi(t),t)=psi(t)*(2*beta*J(t)-N*beta+delta)-Cd;
```

$$CM\_Model := \frac{d}{dt}J(t) = J(t)\,(99.8 - 1.0\,J(t)) - 0.0003600000000\,\psi(t),\ \frac{d}{dt}\psi(t) = \psi(t)\,(2.0\,J(t) - 99.8) - 200$$

> inits:=J(0)=1, psi(0)=1200;

$$inits := J(0) = 1, \psi(0) = 1200$$

> soln := dsolve({CM_Model,inits},{J(t),psi(t)},type=numeric, output=listprocedure):

> j := subs(soln,J(t)): p:=subs(soln,psi(t)):

> plot({j},0..100,title=Cost,color=black);

> plot({p},0..100,title=Cost,color=black);

dsolve and plots

dsolve(CM_Model,{J(t),psi(t)});

$$\left[\left\{ J(t) = \text{RootOf}\left(-\int^{\_Z} -\frac{5}{\sqrt{18\ \_a - 40\ \_a^{3} + 25\ \_a^{4} + 16\ \_a^{2} + 25\ \_C1}}\, d\_a + t + \_C2 \right), \right.\right.$$

$$\left.\left. J(t) = \text{RootOf}\left(-\int^{\_Z} \frac{5}{\sqrt{18\ \_a - 40\ \_a^{3} + 25\ \_a^{4} + 16\ \_a^{2} + 25\ \_C1}}\, d\_a + t + \_C2 \right) \right\},$$

$$\left\{ \psi(t) = -\frac{250}{9}\frac{d}{dt}J(t) + \frac{200}{9}J(t) - \frac{250}{9}J(t)^2 \right\} \Bigg]$$

> odeplot(soln,[J(t),psi(t)],0..100);