UNIVERSITY OF CALIFORNIA, SAN DIEGO

Can You Infect Me Now? A Treatise on the Propagation of Malware in a Cellular Phone Network

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Christopher Brian Fleizach

Committee in charge:

Professor Geoffrey M. Voelker, Chair
Professor Stefan Savage
Professor Alex C. Snoeren

2007

The thesis of Christopher Brian Fleizach is approved:

Chair

University of California, San Diego

2007

DEDICATION

No one can write a thesis by himself and I want to acknowledge a special friend. That man is Mr. Griffin. Comedian Eddie Griffin, your acerbic anti-white humor was a constant source of inspiration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This project would not have been possible without the insightful ruminations and pontifications of my advisor, Professor Geoffrey M. Voelker. He also helped with writing and re-writing major sections of this thesis. Michael Liljenstam provided a crucial component in the network topology generator, without which the project could not have continued, as well as a detailed discussion found in Section 3.4. I am also thankful for the comments and ideas provided by Per Johansson and András Méhes. Priya Mahadevan was kind enough to help explain characteristics of our address book topologies. I am also appreciative that David Liben-Nowell was kind enough to provide the LiveJournal.com data.

ABSTRACT OF THE THESIS

Can You Infect Me Now? A Treatise on the Propagation of Malware in a Cellular Phone Network

by

Christopher Brian Fleizach

Master of Science in Computer Science

University of California San Diego, 2007

Professor Geoffrey M. Voelker, Chair

The prolific spread of mobile phones through all corners of the globe has only been matched by their rapid increase in computing power. As cellular phones become further integrated into the fabric of everyday life, their value to attackers will rise accordingly. As a result, the widespread debilitating outbreak of self-propagating malware in the cell phone environment is a matter of *when*, rather than *if*. Although self-propagating malware is well understood in the Internet,

mobile phone networks have very different characteristics in terms of topologies, services, provisioning and capacity, devices, and communication patterns. To understand the propagation of malware in this new environment, we have developed an event-driver simulator that captures the characteristics and constraints of mobile phone networks. Key elements of the simulator are a network topology generator (RACoON), which creates realistic topologies and provisioned capacities of the network infrastructure, and a social network topology generator, which models address books and the resulting contact graph that would be used by propagating malware. Using the simulator, we evaluate the speed and severity of random-contact worms in mobile phone networks, characterize the denial-of-service effects such worms would have on the network, investigate techniques that malware writers could use to accelerate the rate of infection, and, finally, explore various methods network operators could take to defend against such attacks.

1

Introduction

As rapidly as mobile phones have proliferated, their capabilities have also evolved from a simple communications device into a full-fledged computing platform capable of running general-purpose applications. Recent estimates put the number of active mobile phone devices over 1.8 billion as of 2005 [49]. While the growth of mobile phone use has been astounding, their capabilities have improved equally dramatically. As battery life has steadily increased since cellular networks and mobile phones appeared in the mid 1980's, their processing capability has benefited from Moore's Law as much as computers. The inevitable consequence is that mobile phones have evolved into a viable computing platform for daily needs. Most phones include calendaring software, chatting, Internet connectivity, games, two-way radio, multimedia messaging, and Bluetooth connectivity. Moreover, many of these mobile platforms support Java or other familiar environments [35], meaning a well-established base of programmers already exist that can develop applications

for the devices.

As the differences between mobile phone and desktop computer fade, their successes and weaknesses are bound to be shared. The growth of the Internet has allowed consumers unprecedented ease in ordering and obtaining information and goods, but also with the risk of privacy and security violations. Although 190.9 billion dollars in consumer and business transactions were done on the Internet in 2006 [44], the Federal Trade Commission of the United States reports that identity theft and other related fraud account for 48 billion dollars in losses annually [66]. Flaws in major operating systems, like Windows XP, have allowed nearly 250 million computers [28] to be transformed into zombie computers, under the control of criminals intent on stealing personal information, using the machines to pump billions of spam messages [19] onto the Internet everyday, or launching denial-of-service attacks [22]. The mobile phone industry, with its even larger base of users, greater mobility and more personal nature, is clearly a valuable target for hackers ranging from script kiddies to organized crime. Furthermore, the relentless drive towards adding new features and functionality has a tendency to introduce security vulnerabilities.

Although there have been relatively few pieces of malware specifically targeting mobile phones to date [61, 62], many pundits concur that the cell phone is too lucrative a target to be ignored in the future [12, 18]. What are the motivations that will incentivize malware creators in their attempts to compromise phones? As with the Internet, motivations will likely range from vandalism, identity and informa-

tion theft, loss of consumers through denial-of-service, public relations disasters, and ultimately reduced revenue for mobile phone providers. Attackers could even coordinate large mobile phone botnets for launching distributed denial-of-service attacks, mobile phone spam, etc. How will such attacks be accomplished and through what means? Mobile phones can communicate through a variety of technologies and protocols that allow a diverse set of communication to other mobile devices, and even the Internet at large.

To explore the range of possibilities of malware propagation on mobile phone networks, we have developed an extensive event-driver simulation environment that captures the characteristics and constraints of propagation in this new environment. Since modeling the network is critical to understanding malware propagation behavior, we have developed a Radio Access and Core Operator Network (RACoON) topology generator that creates realistic topologies and provisioned capacities of the network infrastructure. To reflect the relationship of network deployment and provisioning to customer populations, we have also developed a tool to incorporate population census data into the topology generation. Mobile phone networks offer a range of communication services, each with different propagation characteristics and network support. We model two prototypical services, a voice over IP service in which malware can self-propagate by exploiting a vulnerability in the service implementation on the phone, and a Multimedia Messaging Service (MMS) in which propagation depends on user interaction. Finally, we have developed a social network topology generator that models mobile phone address

books and the resulting contact graph used by propagating malware. Together, our environment is capable of simulating malware propagation on realistic network topologies, capacity constraints, and address book contact graphs among millions of mobile phones.

We hypothesize that a worm outbreak on a cellular phone network could have the ability to spread rapidly and cause significant congestion and delays within the network. Using the simulator, we evaluate this proposition by examining. the speed and severity of random contact worms in mobile phone networks, characterize the denial-of-service effects such worms would have on the network, investigate techniques that malware writers could use to accelerate the rate of infection, and, finally, explore various methods network operators could take to defend against such attacks. We focus on malware designed to propagate as quickly as possible throughout a network, since this situation represents a worst-case scenario for both network providers and consumers due to the severe denial-of-service situations that occur. The results provided by the simulator contribute important insights into the severe potential for malware to propagate on mobile phone networks.

1.1 Overview of the dissertation

In the remainder of this dissertation, we look at related work in Chapter 2 that describes efforts in worm simulation and simulation of malware in the mobile domain. These efforts aid in guiding and motivating our exploration of communications-based malware within cell phone networks. We then explore the

requirements for building an accurate simulator in Chapter 3. In doing so, we explore network topologies that model the actual network hierarchy of a Universal Mobile Telecommunications System (UMTS). We also expore the field of social networks and discuss how the characteristics of address books contribute to the spread of malware. We then discuss the design, structure and operation of the simulator given the network topology and the social network. In Chapter 4, we present and analyze results obtained from the simulator. We study different methods in which malware can propagate, including Voice over IP (VoIP) and Multimedia Messaging System (MMS) scenarios. We look at techniques that malware authors could use to increase the rate of infection and find that despite heavy bandwidth and capacity constraints, malware could infect 90% of the population within a matter of hours if designed correctly. We then examine defenses that networks can use to control a malware outbreak. We show that network operators need to act quickly and aggressively to even slow down the spread. Finally, Chapter 5 concludes the study and summarizes the contributions of the dissertation.

2

Related work and motivation

Although there has been a long history of studying the effects of worm and virus propagation on the Internet, the mobile phone network, for the most part, has been given little attention. What little research has been done in the field has primarily focused on worms that spread through physical proximity, such as viruses designed to exploit Bluetooth. However, the internals of the mobile phone network should not be regarded as being similar to the Internet. In fact, there are a number of differences that affect communication patterns as well as worm propagation in a number of fundamental ways.

Computer hosts attached to the Internet have operated under the end-to-end principle [45], assuming that the network in between does not possess any intelligence. The mobile phone environment, on the other hand, offers a different perspective on networking. The intelligence often resides inside the core of the network, with smart routing stations controlling billing, providing differentiated

services and implementing authentication and access control. Traditionally, the end hosts in a phone network have been treated as "dumb terminals" due to their relative lack of functionality. However, this notion is being challenged as phones become more capable and powerful. Complexity exists within the network and at the edges. This has an effect not just on speed at which malware spreads, but also on the defenses deployed by network operators, who possess capabilities not available to the Internet community.

The notion of services provided by a network operator is another distinguishing factor. It dictates the attack vectors available to malware, while also contraining the malware. For instance, an attack that exploits messaging services will behave differently from an attack propagating through a signaling layer. Moreover, each service presents varying constraints in terms of bandwidth or capacity that Internet worms did not face until nearly peak spreading rates were achieved. A mobile phone network is cohesive and finely tuned, provisioned by a single operator so that current traffic levels represent a fairly significant portion of total capacity. Adding capacity and bandwidth is a measure taken only when financial needs dictate so. The Internet, in contrast, is comprised of many operators, which causes growth patterns to differ.

The differences between the mobile phone domain and the Internet provide unique challenges for malware creators, infected mobile phone users and the network operators trying to defend and contain outbreaks. Malware creators need to be aware of bandwidth and capacity constraints, while trying to avoid duplicating

effort by contacting already infected phones. Mobile phone users will experience significant delays and congestion. Users that become infected may now possess a phone that is no longer capable of communication, due to the malware, and needs to somehow be disinfected. The network operators are burdened with the need to quickly recognize that an outbreak is occurring, as well as fashioning a method for protecting against an attack that can work faster than the virus, much like the Internet community. However, they are in a stronger position than Internet operators, in that they entirely control the network and are able to restrict access and limit rate and bandwidth usage. A non-technical issue, but one that is still vitally important, is the public relations and customer service problems that arise from a worm outbreak that infected many phones and disrupted service.

With these differences in mind, we first survey existing research that models and explores computer virus propagation. We then examine initial efforts to study viruses operating within the mobile phone environment and highlight the new contributions that our research offers.

2.1 Traditional computer virus propagation

Understanding the behavior of worms and viruses has long been a goal of researchers in the field of computer security. The characteristics that describe their patterns of propagation and infection are pertinent for a number of reasons, chief amongst them is the ability to "predict failures of the global network infrastructure" and to "use them as an early detection mechanism" [46]. A number of tools

are available for researchers, such as experimental testbeds, real-world experiments, simulation and mathematical modeling. However, modeling and simulation both provide numerous advantages over testbeds and real-world experiments — of note is the absence of legal liability for real-world experiments. Thus, researchers have focused on both modeling and simulation as a means for providing important insights into the spread of computer worms and how they differ from traditional, organic viruses.

Although computer viruses had been around for a number of years — the first known virus appearing in the wild in 1982 [42] — studying the spread of these viruses only started to gain momentum in the early 1990s after Kephart and White [20] discussed a viable modeling method. Using techniques from epidemiology, they employed a Susceptible-Infected-Susceptible model, where agents moved according to a pattern based on random directed graphs. They also recognized that deterministic equations used for years in biology did not account for the vagaries of chance. Their probabilistic modeling showed that the sparse connections through which computer viruses had been spreading limited their infection potential. However, within a few years it was clear to researchers the sparseness barrier had been removed with the connectivity present in the Internet.

As the Internet was adopted by many as a means of communication, it was only a matter of time before traditional computer viruses that percolated through floppy disks found a new method of propagation. These new computer "worms" necessitated a greater clarity into virus propagation that did not follow traditional models

of human interaction. Although the first wide-spread Internet worm appeared in 1988, it did not spur substantial research on the topic [41]. Instead, it took over a decade for the outbreak of another Internet worm to stimulate the research community towards a more comprehensive understanding of the phenomena. In 1999, the Melissa worm appeared, which automatically spread by contacting the email addresses found in a user's email address book. It was perhaps the first virus to gain widespread media attention as it affected many ordinary PC users and appeared at a time when Internet usage was skyrocketing [27]. Shortly thereafter, in 2001, an automated worm named Code Red infected more than 350,000 computers within a 24-hour period [36]. Its rapid spread, coupled with the estimated $2.6 billion in damage it caused to thousands of organizations, became a cause célèbre for researchers interested in understanding the effect that computer worms could have on an infrastructure as complicated as the Internet. Models and simulations soon followed that attempted to explain and predict worm behavior. Subsequently even more virulent worms appeared that exploited flaws in servers and home computers, labeled with monikers like Nimda, Sircam, Blaster, and SQL Slammer [9]. To understand the epidemiological characteristics of these worms, a variety of methods were proposed and implemented [5, 57, 58]. However, a common complaint often discussed was the lack of realism that simulations offered, since the parameters were often much smaller than the actual Internet. Weaver et al. attempted to "scale-down" the Internet to simulate worms [60], while Serazzi et al. [46] modeled advanced behaviors including bandwidth saturation and incorporated the AS

topology into their formulas.

The next generation of worms appeared in early 2003 and the speed and efficacy at which they infected systems was prolific. Consequently, an interest arose to determine how fast a worm could compromise a significant portion of the susceptible hosts on the Internet. The seminal work of Staniford, Weaver and Paxson [48] examined various methods that worms could use to spread in the quickest manner possible. Simulations and analysis showed that within tens of seconds, the entire population of susceptible hosts could be compromised. They had the insight that the sooner the infection rate becomes exponential, the faster the Internet would be compromised. They also conjectured that a careful construction of the worm along with an intelligent deployment strategy could confer a substantial speed advantage. They proposed concepts such as hit-lists that concentrated on initially infecting the most capable hosts in terms of bandwidth and power. Furthermore, they described how worms could use "partitioned permutation scanning," in which each new infection becomes responsible for an ever-decreasing block of addresses. The strategy reduces unnecessary probing of already infected hosts due to uncoordinated random scanning done independently at each host. The combination of these techniques enabled a so-called "Warhol worm" to reach epidemic threshold in literally less than a minute. Further work continued to push the speed limits of "flash worms" to sub-second durations [11, 47, 69].

2.2 Cell phone malware

In the years following the outbreaks of the series of well-publicized fast spreading worms, the desire by malware authors to create worms for the sake of spreading appears to have diminished. Today's malware and worms usually arrive in the form of software packages aimed at handing the complete control of a computer over to another [3], and collecting such hosts under the control of a single owner into so-called "botnets." One consequence of this change is that there is not as strong an impetus to spread rapidly, since the goal is to continue adding members to the botnet without being detected. Even so, an attacker possessing the right motivation and corresponding exploit could potentially take over and control millions of servers and desktop computers.

However, the explosive growth of the cell phone market, along with the commensurate increase in cell phone computing power and generality, has begun to change the landscape of malware. Thus, the profit and motive for attackers to explore malware aimed at cell phones will continue to burgeon [12, 18]. Moreover, there is a growing consensus [16, 23, 54] that the technological platforms on which mobile phones are built, such as Bluetooth, have inherent security weaknesses that may lend themselves to being compromised. Translating the lessons learned from how worms spread on the Internet to that of the mobile phone domain can thus become a valuable tool for planning and mitigating the effects and aftermath when such an attack occurs.

Recent studies have begun to look at worms and viruses in the context of mobile

phones, mostly by examining their different capabilities. The notion of how quickly a piece of malware code can spread through the network has been explored on a few levels. Interestingly, there has been little attention paid to worms that spread through the communication functions in a phone, which has in turn motivated this study. Most efforts thus far have focused on the spread of Bluetooth-enabled malware. Bluetooth, as a proximity-based wireless service, has the potential to act much like traditional viruses. Malware can spread only when two entities come in close enough proximity (usually less than 10 meters), for a long enough period of time (more than a few seconds), to engage the other device and send the payload. Already there have been reports of some viruses that use Bluetooth as a means to propagate, such as Cabir [61] and Commwarrior [62]. However, they also rely on social engineering techniques to infect the device.

To understand more closely how a Bluetooth-capable outbreak in the wild would behave, some approaches have focused on studying human movement patterns since such patterns determine the contact proximity necessary for propagation. Perhaps the most in-depth study has been from Su et al. [50], who went to actual locations and measured various statistics about Bluetooth usage and duration of contact. A key result was that approximately half of the phones they encountered were in contact long enough to establish a connection and transfer enough data for a virus to copy itself. Kostakos et al. [21] instrumented a city-wide deployment of Bluetooth monitoring equipment in Bath, England. They found that only 8% of their users had discoverable Bluetooth devices and that the

duration of contact time was scale-free.

Mickens and Noble [30] took a different approach and focused on modifying traditional analytic models to create a probabilistic queuing technique that accounted for movement and traffic patterns over various time durations. Since it accounts for the mobility properties of mobile phone users, their model was better suited to accurately model mobile phone malware propagation than the standard Kephardt-White model. Zheng et al. [67] also studied modeling proximity-based malware propagation on mobile phones. They focused on the population distribution density, Bluetooth radius and node velocity. Their results point to a variety of quarantine methods that could greatly reduce the virulence potential.

However, the interest in Bluetooth propagation has perhaps overestimated the threat of the proximity-based infection vector. Of course, malware could certainly use Bluetooth, and it has done so in the past [62], but its spread would be severely limited by human movement patterns. More importantly, though, are the severe restrictions most phones have on Bluetooth use. For instance, any Bluetooth-enabled worm would need to discover the address of the device in order to communicate. If the address is unknown, attempting to correctly guess Bluetooth addresses through brute-force is an option available only to dedicated adversaries who are in lengthy physical presence with the device [4]. The addressing difficulty is further compounded by the observation that very few mobile phone users (8–10%) keep Bluetooth enabled in discoverable mode [21]. Bluetooth drains battery life more quickly and most phones leave it off by default. Some phones cannot even

stay in discoverable mode permanently. These problems suggest that malware that spreads through Bluetooth is unlikely to spread quickly enough to affect a large proportion of the user base.

Beyond Bluetooth, there are a number of other avenues that malware can utilize to infect and impact mobile phones and their supporting infrastructure. Racic et al. [43] showed how an attacker could surreptitiously drain the battery power on a user's cell phone much faster than normal rates. Their investigation also revealed that the MMS service leaks a significant amount of information about the device. Traynor et al. [55] studied a form of denial-of-service attack in which a single cable-modem was capable of producing enough SMS messages to effectively deny cellular service to a large number of customers in a service area. They note that since SMS messages use the same control channels as voice data, there is contention that can be exploited to deny access to other services.

These studies, however, have ignored a vital component of the mobile phone environment — namely that malware would be able to communicate with other devices through the network. Two such examples of communication that we focus on include the Multimedia Messaging Service (MMS), a service that expands upon the text-based Short Messaging Service (SMS), and Voice over IP (VoIP), enabled through adoption of the IP Multimedia Subsystem (IMS), a new service that is currently in the early stages of deployment. Subsequently, malware using communication channels would be subject to capacity constraints and bandwidth limits as it propagates through a cellular network, a point that past research efforts leave

unaddressed.

With these considerations, it is our goal to understand the characteristics of a communication-based worm as it spreads through a cellular network. The differences between a cellular network and the Internet would cause the features of worm propagation to alter drastically. The consequences of these differences are two-fold. First, clever malware creators would use different strategies to maximize infection potential. Second, and more importantly, defenses aimed at slowing down and ultimately stopping the spread would need to be cognizant of economic constraints, such as the need to quickly alleviate congestion for customers, lest the mobile network operator experience significant financial loss.

To answer these questions, our simulator models a real mobile phone network that accounts for bandwidth and capacity constraints, while operating on realistic population statistics and geographical considerations. It is not our goal to show that such a worm is possible through vulnerabilities — we take this as a given — but rather to realize that as mobile phones become more pervasive and powerful, they will experience the same problems that the Internet has had in the past in dealing with malware that spreads through communication mechanisms. It is our hope to explore the relatively unknown domain of malware propagation in mobile phone networks, taking into account as much realism as possible, and to use these results to explore viable counter-measures.

2.3 Feasibility of a communications-based attack

For the most part, we ignore the specifics of how an attack might occur. We do, however, assume that the population of mobile phones are generally smartphones and that they have the ability to run general purpose applications and that they allow those applications to perform a wide range of network-enabled actions, such as programmatically initiating calls. Although smartphones comprised only 3.8% of the U.S. cell phone market at the end of 2006, this number is expected to grow rapidly [53]. Further, the market is predominantly reliant on a single operating system, reminiscent of the homogeneity of Windows on the Internet. As of the first quarter of 2007, 72% of smartphones ran Symbian operating systems, followed by BREW and J2ME. Microsoft's WindowsCE has a nearly 7% market share, but is growing rapidly [52]. While J2ME applications are usually restricted, Symbian, BREW and WindowsCE open an API to developers that allow them to, for example, send MMS messages, make phone calls and send email messages [51]. The implication is that most smartphones on the market will have a feature-rich set of abilities malware could make use of to spread. A caveat, however, is that most platforms require some form of code signing. As a result, previous malware outbreaks have entirely relied on user intervention to start the infection process [61, 62],

3

Methodology

To develop a realistic simulator, we first investigated two corollary issues that impact the spread of malware. The network topology defines how the components in a Universal Mobile Telecommunications System (UMTS) cellular phone network are interconnected. To create the network topology, we reference information regarding population densities, geography and leverage knowledge from deployed networss. Next, we study the role of address books in determining the spread of malware. Social network topologies have been well-researched and we investigate a number of likely scenarios and their characteristics. With this information, we proceed to create a simulator capable of using the network topology, while creating a social network topology that connects mobile devices. To underpin the discussion, we start by explaining how a UMTS network functions.

3.1 The Universal Mobile Telecommunications System

To capture the effects of communication within a mobile phone network, we focused on the forward-looking Universal Mobile Telecommunications System (UMTS), a third-generation mobile phone system standardized by the the 3rd Generation Partnership Project (3GPP) as the successor to the Global System for Mobile Communications (GSM). Although its deployment in the U.S. has been slow compared to other regions, such as Europe, there has been at least one major carrier that has announced plans to begin roll out of UMTS in the near future [13]. UMTS defines a hierarchical set of nodes that interact to provide seamless voice and data services to handheld devices. We present a simplified version of this network and describe the changes we have introduced when we simulate a malware outbreak within such a system.

Figure 3.1 illustrates a simplified version of a UMTS for one network carrier. We have assumed, for simplicity, all communication is packet-based and that signaling and control channel effects are not significant when compared to packetized bandwidth limits. In the system, a mobile device connects through a radio interface to a Node B — essentially a radio tower. The Node B is responsible for transmitting and receiving radio signals between the mobile devices. Each user is typically bandwidth-limited, while total bandwidth for a Node B is also capped. The Node B then forwards data to a *radio network controller* (RNC), a control-

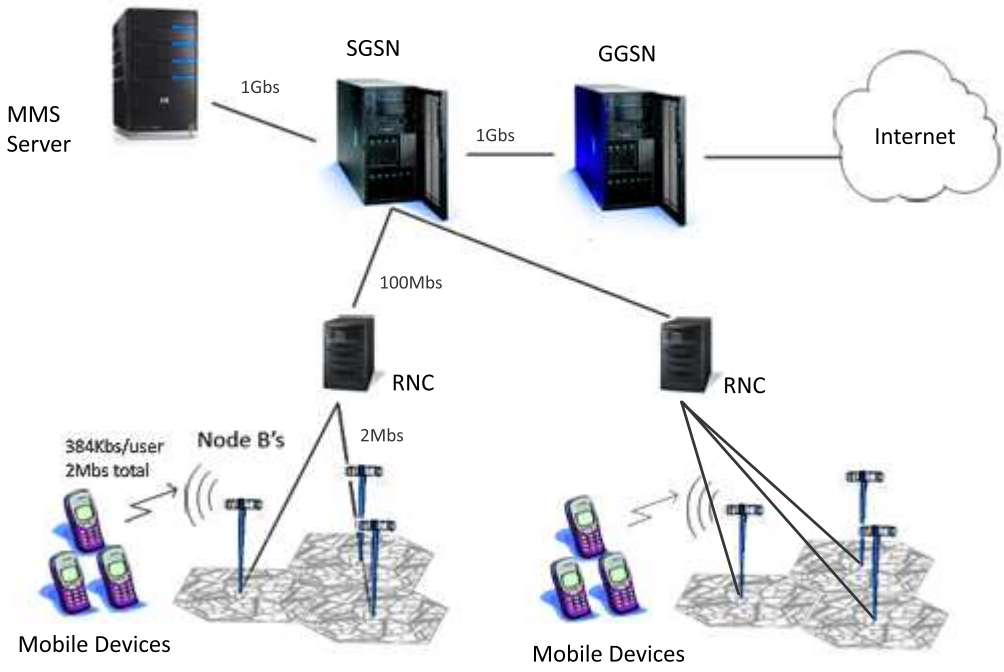

Figure 3.1: A UMTS network is comprised of handheld devices, which transmit and receive from Node Bs. The Node Bs are connected to RNCs, which are connected to SGSNs. SGSNs can route to other SGSNs or to GGSN's, which connect to the Internet. MMS servers connect to the SGSN backbone.

ling element in the network that is responsible for radio resource management including hand-over and admission control, amongst other duties. The RNC is connected to a *serving GPRS support node* (SGSN), which handles routing, authentication and charging functionality for a region. Typically, there are very few SGSN's in a network compared to the number of Node B's. We have not shown the backbone between multiple SGSN's, which connect together to form a routable network. The hierarchy depicted in the figure is the path through which packets

follow in our simulation. We also depict a Multimedia Messaging System (MMS) Server connected to a SGSN. There may exist relatively few MMS servers for a very large region. In our simulator, we have simplified MMS communication by introducing only one MMS server that is connected to one SGSN. Finally, we also show a Gateway GPRS Support Node (GGSN), which is responsible for bridging data between different interfaces, such as the radio network and the IP network, including the Internet. It also authenticates and performs some billing functions. We do not model GGSN nodes since the scenarios we examine in our simulator focus primarily on communications within the radio network.

The following sections describe how our simulator operates using a network topology generated by modeling a UMTS.

3.2 Network modeling

The design and implementation of an accurate simulator required the investigation of a number of related issues having a significant impact on the correctness of the malware propagation. Our initial step to recreate realism was to obtain and produce a representation of actual population densities for a region of land large enough to provide a meaningful analysis. With that geographic information, we then generated a corresponding topology of the infrastructure of a mobile phone network. The topology determined the paths through which data would travel, which dictated how the malware would propagate. It also determined the capacity of the network, which affected the rate at which infections occurred. Using real

data to inform decisions about simulations is a critical factor in achieving realistic insights. Others have also followed this path. For example, Su et al. [50] used data from the Reality Project [14] to help simulate interactions between people moving in a physical space. In an orthogonal fashion, data sources can be used to derive the characteristics of distribution models, such as Zou et al.'s [69] use of data from Yahoo! email groups to model the degree distribution of contacts within email address books.

Another important consideration for achieving realism was choosing the vectors through which malware would spread. A locally transmissible virus would use Bluetooth to propagate, raising questions of how often and at what times people came in contact with each other. As Bluetooth-enabled viruses have been examined heavily in previous studies, we did not focus our efforts on proximity-based worms. On the other hand, if the virus used remote communication methods, such as MMS or VoIP, then its rate of infection would depend on the phone numbers listed in the address book and the constraints of the underlying network.

3.3 Population and geography

To create a realistic infrastructure of a mobile phone network, we used the U.S. Census data from 2000 for county subdivisions [56], the smallest unit of measurement the U.S. government produced, to determine population densities and distributions for regions which could be subdivided into radio cells in a grid. The county subdivisions contained the latitude and longitude, land area and total pop-

ulation for each county. This non-standard format was not particularly amenable to processing and, moreover, our goal was to create a region of land that was broken into standard-sized grid cells on a finer-grained scale than provided by the census data. We therefore created a program to convert the census data into an XML format, described in detail in Appendix A, that specifies the location and population for each cell in the grid.

The program first takes the latitude and longitude of each county subdivision and places it into a two-dimensional array. It then uses the land area of the subdivision to, in effect, draw the entire region around the specific coordinates specified by the census data. Each grid within the two dimensional array was the equivalent of 1/100 of a degree of latitude and longitude. The program then determines the population for each cell by dividing the population from the county by the number of these cells, and uniformly assigning the average to each one. Finally, the program iterates over each position and coalesces regions together to produce a map made of square grid cells one mile on a side. [1] If two smaller regions overlapped, the populations were averaged together for the final result. The tool then outputs an XML representation of this in-memory data structure as the final step. In addition to the resulting XML file, it can generate a visual representation to confirm the accuracy of the transformation. Figure 3.2 depicts the results of transforming the U.S. Census data from 2000.[2] Areas with high

[1] In future work that involves other propagation vectors, such as Bluetooth, a higher resolution grid is required. In that case it would be reasonable to have grid cell dimensions on the order of 10–100m, considering the transmission range of the Bluetooth radios.

[2] The elongated shape of the United States land mass occurs because we have ignored the effects of the curvature of the Earth.

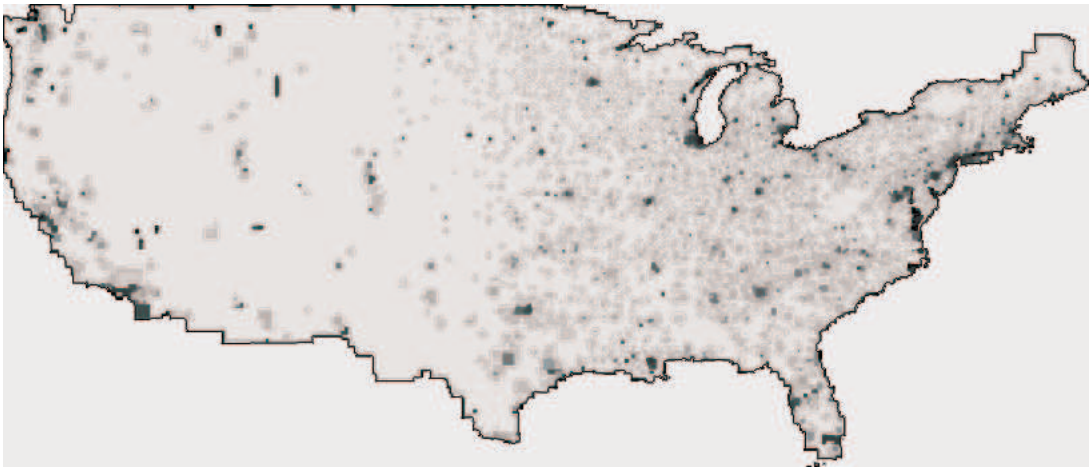

Figure 3.2: Visual representation of geographic census data.

population densities appear in darker shades of gray, whereas areas with lower population densities contain a lighter shade. The light regions represent areas without substantial population.

3.4 Cell phone network topology

To create a realistic foundation to communication between devices, we developed the Radio Access and Core Operator Network (RACoON) topology generator to create the topology of a UMTS cellphone network, as described in Section 3.1.

3.4.1 Radio Access and Core Operator Network Topology Generator (RACoON)

To model the communication between cell phones, we created a coarse model of the capacity of the system to determine how potential capacity bottlenecks affect the propagation dynamics of malware. Since the capacity of the system is not only determined by the capacity constraints of the technology and components, but to a large part by the dimensioning of the system, the network topology generator attempts to mimic the network planning and design process in a high-level fashion. It should therefore first be emphasized that a tremendous amount of complexity and detail is abstracted away, and consequently ignored, in our model. Instead, we focus on capturing the characteristics of the network topology and capacity that are essential for modeling malware propagation.

When network carriers design a real cellular network, they typically use as input data a combination of land use data of a region, optionally supplemented with population data; they also consult telephony traffic workloads available from previous deployments, if available. They estimate the number and typical distributions of subscribers, and the typical traffic mix (voice, data, other services), and use it as input to the planning process. Much of the complexity in the network planning and design lies in dealing with radio propagation issues. The system cost needs to be minimized while ensuring adequate coverage and capacity exists. The planning may start through a cell plan — a hexagonal layout — that is adjusted by hand to improve performance. However, due to the many practical constraints

related to obtaining land access to sites for base stations, the starting point is more likely to be an approximate site plan that has already been defined based on the topography, rough population patterns, and site access constraints. The radio cell planning process then consists of tuning the system and verifying that it meets the coverage and capacity goals. In our case, we omit practically all of the complexity related to radio propagation, and instead base our model on meeting the capacity demands of the population base in the region in a coarse fashion. The only aspect of the radio propagation included in the model is the maximum transmission range, which defines the maximum radio cell size.

Since our intended malware propagation scenarios were defined to either propagate through MMS or a software-related Voice over IP vulnerability, we decided to focus on packet data traffic and ignore circuit switched traffic in the network. We recognize, however, that the signaling involved in establishing packet data communications requires communicating with nodes in the circuit switched domain of the network.

3.4.2 Operation and data flow

As input for generating the network topology we chose to work from publicly available census data to create a coarse population distribution, as described in Section 3.3. This distribution represents the average population density or, alternatively, the peak population density expected in different parts of the simulated region, such that we can capture the presence of population centers vs. sparsely

populated areas. From the population distribution we generated a plausible network topology that would handle the expected user traffic in the region. We then output the topology in a format that the simulator could easily load and parse. The simulator also requires the population density information, so RACoON also includes that information in the topology information file as well.

The output from RACoON is similar to Internet topology generators [8, 29] and we considered using previously defined file formats. However, since the node types and the hierarchy for the radio access network and operator core network differ greatly from the two-tier Internet model, adopting an existing format without fundamental changes was not possible. Moreover, these tools tend to use various ad-hoc text formats with hand-crafted parsers. Since an XML-based format would be easier to implement and more likely to be used in other studies, we defined a new XML-based file format described in greater detail in Appendix B.

3.4.3 Population density representation

To simplify the topology generation, we divide a simulated region — the *population grid* — into grid cells. The grid cell size should be chosen such that the population resolution is useful even in the most densely populated areas. Unfortunately, due to the lack of resolution in the census population data, a high resolution in the population grid was not possible. Thus, the grid cells used to build the topology were set to be 1x1 square miles.

Although we would have liked to use the entire United States in simulation,

practical computing resource concerns limited our simulation to a specific region. We chose to study a metropolitan area and its surroundings such that the population grid contains a mixture of densely and sparsely populated areas. Due to these differences, it could be useful to use a *multi-resolution-inspired* approach, specifically a non-uniform size of grid cells. This approach would also have the added benefit that the population density representation aligns with the way the radio access system is built to match different population densities in different areas. However, for simplicity of implementation, we decided to use a uniform grid for our study.

3.4.4 Network generation

The topology generator uses a bottom-up strategy to construct the network, starting by placing the radio cells and Node B's, and then proceeding to construct the (fixed) transport network from the radio access edge to the core network. Network nodes are added by connecting them to previously added nodes as each layer is added to the hierarchy, such that they obey capacity constraints of the nodes. The exact types of constraints may vary from node to node. For example, the maximum bandwidth and/or maximum number of users might be referenced to add a node so that the capacity constraints are obeyed to the greatest extent possible. The actual process that network operators use to design a network is similar in this respect. However, they also use a tremendous amount of additional detail in terms of performance constraints and the required capacities in different sys-

tem traffic channels expected to arise from the assumed load and traffic mix. To dimension the links interconnecting the nodes, a *typical* bandwidth was chosen to represent the interconnection of each type of node. This approach also represents a significant simplification of the real process.

To outline, the topology generator takes the following steps:

1. Place radio cells and Node B's in a grid cell.

2. Add RNCs by grouping a number of adjacent Node B's together and connecting them to an RNC.

3. Add SGSNs by grouping a number of adjacent RNCs together and connecting them to an SGSN.

4. Interconnect SGSNs to form a core network.

5. Create a MMS-C node and attach to the network.

To algorithmically deal with the non-uniformity of population and capacity demands, the generator creates a *multi-resolution* aggregation data structure to group nodes. Starting from the original population grid, RACoON creates several tiers by aggregating population grids with successively larger quadratic cells. Each level aggregates four cells in a tier below to form larger cells, as illustrated in Figure 3.3. Rather than requiring that the original grid contains a power of two number of cells, RACoON adds *null cells* to the edges to create the illusion of even powers of two.

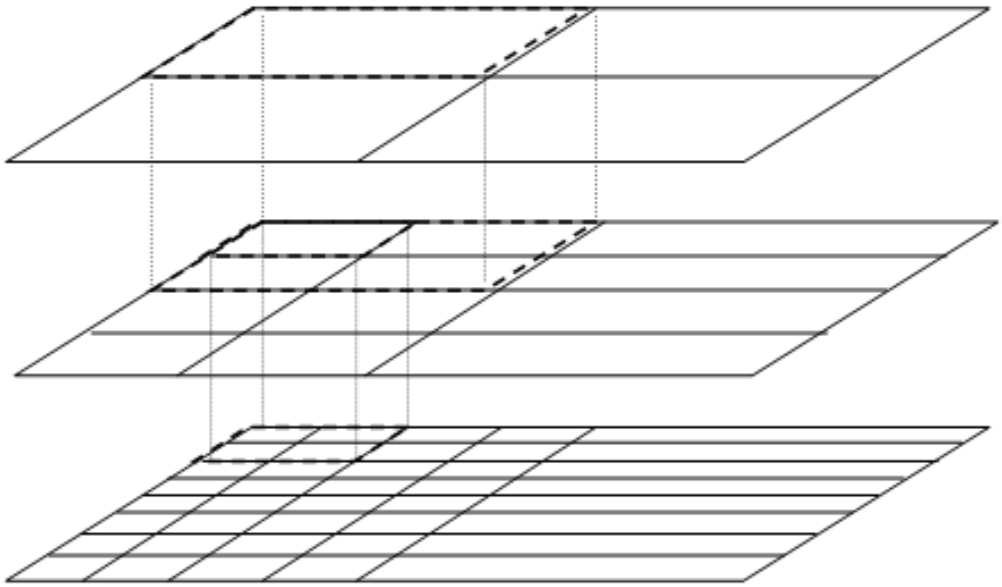

Figure 3.3: Aggregation of population grid data by a multi-resolution data structure.

At each level, each aggregated cell holds information about all network nodes in that region. It also stores key measures used in the topology building process, such as the population size in the aggregated cell and the aggregated bandwidth required. Starting with the addition of RNCs, then proceeding with SGSNs, the generator starts from the top of the aggregation hierarchy and checks, for each cell, if the capacity constraints for the node are met at the cell. If they are, the node is created and all nodes at the next lower layer in the node hierarchy are connected to it. If the constraints cannot be met, the region is split by proceeding downwards to the next lower level in the aggregation hierarchy, and attempting to add one node for each cell at that lower level. This procedure has been motivated

in part by R-trees [15].

The placement of radio cells in the generator corresponds to the cell planning stage of a real network design. However, in reality, neighboring radio cells overlap in order to support hand-over of connections between cells. Our model uses non-overlapping radio cells so that choosing a base station for a phone to connect with is well-defined. The radio cell placement strategy is likewise kept simple. The goal for placing radio cells is to map one or more population grid cells to a radio cell, such that the required traffic can be supported by the system and that the maximum transmission range is not exceeded. However, due to the limited resolution of the population grid, the current radio cell placement is further simplified to map one radio cell to each population grid cell. This mapping assumes that the population grid cell size does not exceed the radio transmission range. As a result, we achieve full radio coverage for the simulated region. Moreover, we assume omni-directional cells used throughout, rather than the sectorized radio cells that are often used for more densely populated regions. Hence, one omni-directional radio cell and one Node B is deployed for each population grid cell. The maximum packet data bandwidth available to one user in a radio cell is assumed to be 384 Kbps, and the total available bandwidth in a radio cell 2 Mbps. These values have been derived after discussions with engineers responsible for provisioning networks and represent current practices in the field.

The next step is to connect the Node B's in a cell to a Radio Network Controller (RNC) that manages that area. The capacity criterion used for the RNC is defined

by a maximum number of Node B's that the RNC can handle. For the model we assumed this to be on the order of a couple of hundred, choosing 256 for a 16x16 square mile region of Node Bs. Each Node B has a 2-Mbps link to its RNC.

Similarly, RNCs are grouped together in quadratic regions and connected to Serving GPRS Support Nodes (SGSNs). The capacity constraint chosen for the SGSNs is the maximum number of Simultaneously Attached Users (SAUs). We assume that each SGSN can handle up to 10,000 SAUs. Note that the generator makes a conservative assumption that all users in an area may need to be connected at once. However, this does not mean that they all necessarily send or receive traffic at once. It means only that they are attached to the network and can send packet traffic communication. Additionally, state for each phone is stored in the network. For normal Web browsing and sending and receiving of MMS messages, maintaining per-phone state is typically not done. But for some applications, like push email clients, it is common for the terminal to stay attached to the network continuously. Hence, it is plausible that this type of device behavior will become more common over time. In terms of bandwidth, the maximum bandwidth for all the cells is simply summed. Similarly, this assumes that the maximum capacity may be used simultaneously for the whole service area, which is also conservative.

When geographically placing the RNCs and other network nodes, it is plausible that these nodes will tend to be closer to population centers. Therefore, we chose a simple approach to placement. The generator calculates each node's position as the population centroid for the grid cells handled by the nodes in the lower tier.

Once all the nodes have been created, the SGSNs need to be interconnected to form the mobile packet core network. Lacking any significant empirical data regarding the preferred topologies for the core network, we chose to use a model similar to the Waxman model [59], a distance-biased random topology. Thus, nodes that are geographically closer to each other will have a higher probability of having a direct link between them than nodes that are further apart. In Waxman's model, nodes are placed at random in a two-dimensional plane, and then connected pairwise by links using the probability distribution function:

$$P(u, v) = \beta * e^{\frac{-d(u,v)}{L*\alpha}}$$

where $d(u, v)$ is the distance between nodes u and v, L is the maximum distance between nodes, β is a parameter to control the link density, and α is a parameter to control the mixture of short to long links. For a probability distribution function in our model, we simply used the inverse of the distance between the nodes

$$P(u, v) = \beta * \frac{1}{d(u,v)}$$

where, again, β is a tunable parameter to control the link density.

Since this procedure does not guarantee that the graph will be connected, the generator performs a final breadth-first-traversal of the graph to check for and ensure the graph is connected, adding links if necessary.

Finally, for the scenarios focusing on malware propagated by MMS messages, a MMS-C node needs to be added to the network. In reality, the MMS-C node would normally be placed in a service network located on the Gi interface of the GGSN

node, i.e., the side connecting to the rest of the Internet. In the model, we have simplified this placement somewhat so that an MMS-C node connects to one of the SGSN nodes (the one covering the largest user population). Given the typical capacity constraints of the MMS-C node itself, compared to the transport network connecting to it, this simplification should not make any significant difference. The capacity of a single MMS-C node is on the order of hundreds of messages per second (MPS) [6]. Although this capacity appears quite small, given that the expected traffic load at busy hour is assumed to be significantly less than 1 message/user/busy-hour, a single node is able to serve millions of subscribers. Again, these values have been assigned based on current network provisioning knowledge.

Figure 3.4 shows an example of a network topology for a 200x200 square mile grid from the northwest United States. The southeastern part of the region has a higher population density than the western part, as evidenced by a higher concentration of SGSN nodes.

Our model was sufficient to simulate packet network mediated malware propagation in a single provider network. Future work could further extend the model to include multiple providers, gateway nodes between their separate networks and even different system deployments, such as GSM.

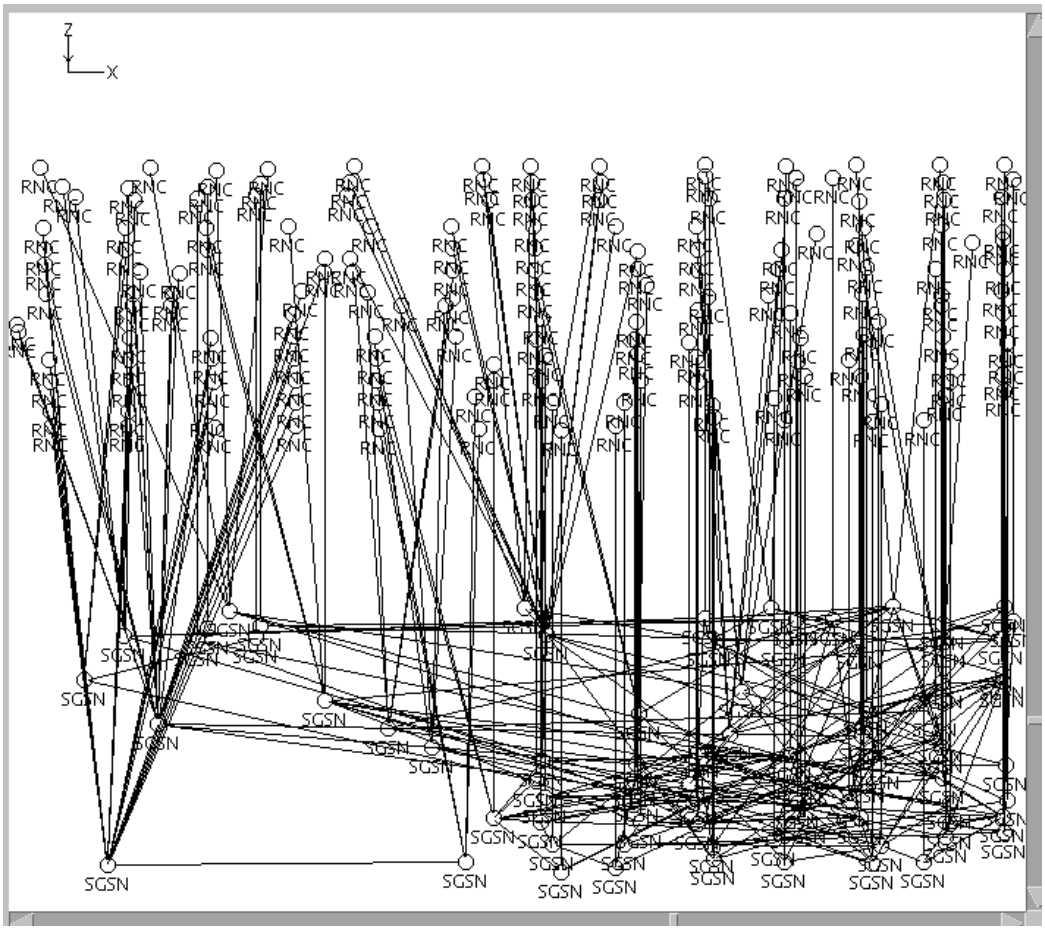

Figure 3.4: Example of a generated topology. The X-Y plane corresponds to geographic position. The Z-axis is used to separate nodes in the hierarchy. Node B's are not shown.

3.4.5 Topology information

The topology we used in our experiments was based on a 100x100 square-mile grid of cells centered around the Boston metropolitan area. It contained 7,234,667 people initially, which was scaled down in each cell in accordance with the penetration of cell phones in the United States. At the end of 2006, cell phones had reached 77.7% of the U.S. market [34], thus the population we used in

Table 3.1: Overall network topology statistics

Network Topology Characteristics	
Topology size	100x100 sq. mi
Population	5,621,336
# of Radio Cells and Node B's	9616
# of RNCs	49
# of SGSNs	49
# of MMS servers	1

our simulations was 5,621,336. Table 3.1 lists detailed information regarding the settings of the network topology used in our simulations.

3.4.6 Communication in the network topology

Malware that makes use of communication facilities within a mobile phone, such as SMS, MMS, Voice over IP or even Internet applications, will necessarily rely on the underlying cellular infrastructure to carry its data. The rate at which malware can spread, therefore, is a function of the latency and bandwidth of those underlying connections. As mentioned in section 3.6.3, there are a number of components in a modern cellular network that are involved in transmitting packetized data. Our simulation made use of these connections when two mobile phones contacted each other or when a message was sent to an MMS server.

The simulator uses the infrastructure to inform the communication process between two phones so that the initiator and the receiver each check the available bandwidth for sending data. If there is not enough bandwidth available at that time on either end, such as if other mobile phones have consumed all the bandwidth,

then the initiator essentially hangs up and waits until the next time step before performing another operation. This models how a mobile phone might react in the case that it could not successfully send data.

If there exists sufficient bandwidth, the phones negotiate the maximum amount of bandwidth it can use based on available bandwidth and any user limits. The phones then reserve that amount of bandwidth for the duration of sending the malware payload. Both parties release the bandwidth when the payload delivery completes. The simulator calculates the duration based on the available bandwidth and the latency for each component. The data packets would typically follow a path that starts at the radio cells of the initiator and moves to the Node B, then the RNC of the initiator, the SGSN of the initiator, intermediate SGSNs, the SGSN of receiver, the RNC of receiver, the Node B of receiver and finally wireless transmission at the radio cell.

The SGSNs form the backbone of the mobile phone infrastructure, although they are not all directly connected to each other. In fact they will route packets to the appropriate SGSN through a series of intermediate hops. The source routes between SGSNs are encoded for each SGSN within the network topology file so that the simulator is aware of the correct paths to follow so that the simulator allocates bandwidth and models latency during the propagation of the malware. The simulator calculates source routes using Dijkstra's shortest path algorithm. Table 3.2 lists the bandwidth and capacity constraints used in our network topology. Note that bandwidth and capacity constraints are not full duplex — they

Table 3.2: Bandwidth and capacity network topology statistics

Network bandwidth and latency characteristics	
Max. user bandwidth at radio cell	384 Kb/s
Total bandwidth at radio cell	2 Mb/s
Latency at radio cell	0.2 s
Bandwidth from Node B to RNC	2 Mb/s
Latency from Node B to RNC	0.2 s
Bandwidth from RNC to SGSN	100 Mb/s
Latency from RNC to SGSN	0.3 s
Bandwidth between SGSN nodes	1 Gb/s
Latency between SGSN nodes	$<$ .001 s
Bandwidth between SGSN and MMS server	1 Gb/s
Latency between SGSN and MMS server	0.1 s
# of messages/s per MMS server	100

must be shared by inbound and outbound data.

3.5 Social network topology

An important consideration for malware that propagates through remote environments is the manner in which it spreads. One possible method of propagation would be to randomly dial phone numbers and hope that the targeted phone is both a cell phone and powered on. A smarter worm would make use of the address book within the phone to be aware of already existing phone numbers. Of course, they may not all be mobile devices, but as cell phones become more pervasive, the percentage of cell phones within address books will rise accordingly. Thus, modeling the number of contacts, as well as who the contacts are, is an important

aspect towards understanding the spread of a worm.

The social networking field has been well studied and we first look at related work in regards topological structure and topology characteristics before describing the social networks that we use in our simulations.

3.5.1 Address book degree distributions

The topological structure of social networks is a phenomenon that has long been studied in sociology, physics, medicine and computer science. Understanding and programmatically generating these networks has often taken a few forms, including random, small world, or scale-free networks. A random network assigns, at random, connections between nodes. Randomness comes in terms of who is connected together and the number of connections per node. Although random topologies can be useful for sanity checking, real-world networks often exhibit characteristics that are more well-defined.

A small world network implies that any two individuals are connected through a short chain of links. These types of networks are pervasive in many real scenarios. For example, Newman [38] has found them present in the structure of scientific collaborations. To generate such a topology, Huang et al. [17] suggest assigning $2n$ links to n entities and then randomly re-wiring the nodes.

Other studies have recognized that real-world phenomena can often be classified as scale-free networks, such as the Internet AS topology [7, 26], professional links between movie actors [1], email address books [65] and file sizes on computers [32].

A scale-free network is one in which a few nodes act as highly connected hubs, while the remaining majority of nodes have relatively few connections.

A characteristic of scale-free networks, and occasionally small world networks, is that they follow a power law distribution in terms of the degree of contacts per node. A power law graph has the form of $y = ax^k$, while its distribution can be modeled as $p(x) \propto x^{-a}$. When plotted, the distribution appears as a straight line on a log-log plot. This line is often referred to as a *long-tail*, indicating that while most nodes have very few connections, the degrees of the remaining nodes fall within a wide range. In lieu of a power-law distribution, occasionally studies will argue that a log-normal distribution provides a better fit for an instance of the long-tail phenomena [24]. Mitzenmacher [33] relates the history of the question, noting that as early as the 1950s, there were arguments as to whether income distribution followed a power law or log-normal distribution.

Despite the wealth of information on social networks, there is scant information explicitly regarding address books in mobile phones. A first step in exploring the domain then is to recognize that, although not completely similar, email address books might share many of the same characteristics as mobile phone address books. Zou et al. [68] collected data from the size of Yahoo! email groups and found that there was a wide range in the size of these email lists, from just a few users to hundreds of thousands. In aggregate, the distribution matches a power law distribution well. They used this data to model the spread of an email-based worm. When compared to other possible topologies (small world and random), malware

propagates faster on a scale-free network. The key insight in understanding worm propagation in scale free networks is that once an infection reaches a highly connected node, the spread of the worm is amplified throughout the network.

In a similar vein, Newman et al. [39] obtained address book data from a large institution and measured the in and out degree distributions of contacts. They discovered a small difference between in and out degree — most studies assume undirected connections — and, although they could not corroborate Zou's power law distribution, they observed it was still heavy-tailed. Their data pointed to a faster decaying distribution that was better modeled by a simple exponential $p(x) \propto e^{-j/j_0}$ where j represents degree and j_0 is a constant. The differences between in and out degree pointed to a question about reciprocity between connections. They refer to their topology as semi-directed, as only 23% of nodes shared bi-directional edges. This result is somewhat surprising and has the potential for altering the results of previous studies that have made such a simplifying assumption. However, despite these findings we have used only bi-directional links and follow the lead of other studies.

Liben-Nowell [24] collected another source of data offering more promise than email address books in terms of cell phone address book topologies. He used the "friends" of LiveJournal.com users and found the in and out degrees of each user. Since the Web site required active participation and manual addition of friends, he reasoned it was a valid indicator of real-world social networks. The data showed a long-tailed distribution that was better characterized by a log-normal distribution.

In contrast with a power-law distribution, a log-normal distribution's *tail* does not go on forever. It also has a higher concentration of nodes near the end of the tail, whereas the power-law distribution is more evenly spread across the range of values.

However, there have not been any efforts that have explicitly addressed the distribution of contacts for mobile phones. One obstacle is the inherent difficulty in obtaining the data. Mobile phones are personal devices controlled by the owner (opposed to email addresses to which administrators sometimes have access) and call logs are closely guarded by network providers. After reviewing the literature on social network topologies, we were concerned that a scale-free distribution was not an accurate representation of a cell phone social network topology. For one, there is often a hard limit in the number of contacts stored in a cell phone. For example, the LG enV cell phone has a phone book capacity of 1,000 [35]. It also seems unlikely that the large majority of users would have only two or three contacts; anecdotally, many people have a much higher degree of connectivity. To explore this further, we sent email solicitations to the Computer Science Department at UCSD and to Ericsson research in Sweden, asking how many contacts were in people's address books. There were a total of 51 responses from UCSD and 22 from Ericsson. We then rounded the number of contacts down to the nearest multiple of ten and plotted the data, shown in Figure 3.5. The results appear to look more like a shifted Gaussian than a scale-free distribution, although admittedly the number of samples is small.

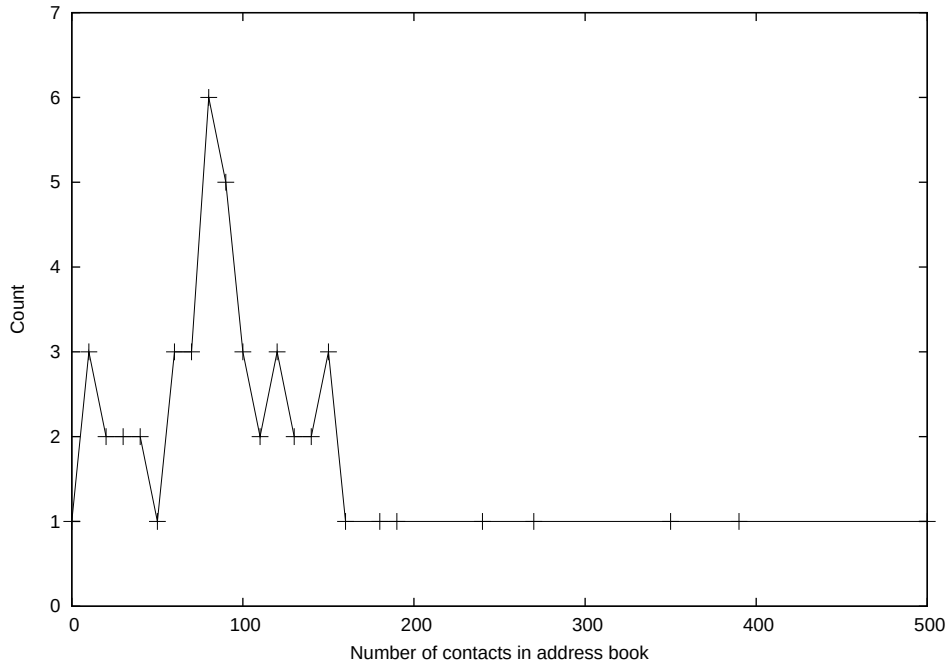

Figure 3.5: The degree distribution of mobile phones users collected at UCSD and Ericsson.

In fact, we found that an Erlang distribution provided a good fit for the data. An Erlang probability distribution is characterized by the formula

$$P(x) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}$$

where λ provides the rate of change and k defines the shape of the curve [63]. We found that if we used a $\lambda = .04$ and $k = 3$, we could create a distribution with an overall average of 65 contacts, which followed our anecdotal evidence as well as survey data.

In our experiments, we evaluated networks with different degree distributions — log-normal, power-law and Erlang — ranging from 1 to 1,000, and observed the effect that it had on the propagation of malware. The details are further

Table 3.3: Graph statistics from an example Erlang degree distribution

Example address book information	
Avg. node degree	65.67
Max node degree	377
Diameter of graph	5
Avg. distance of graph	3.02

discussed in Section 4.1. However, we settled on using the Erlang distribution for the majority of experiments, because it made more intuitive sense based on our survey data. Table 3.3 presents statistics on a sample Erlang address-book contact topology generated by the simulator.

Figure 3.6 shows the probability density of the Erlang distribution as given by the formula described above (dotted line), the degree distribution generated by the simulator (points marked with a "plus") and our survey data, scaled appropriately (dashed line with marks). The generated counts have been normalized to show the correlation with the formula. Figure 3.7 displays the absolute value of counts from the Liben-Nowell's LiveJournal.com data set and the normalized data generated by our simulator when generating a log-normal distribution. Despite their use of different scales, the similarity is apparent. Figure 3.8 shows the probability distribution function of a power law degree distribution and the distribution as generated by the simulator. The distribution uses -1.7 as the power law exponent, a value obtained from Zou et al.'s study of Yahoo email group distributions.

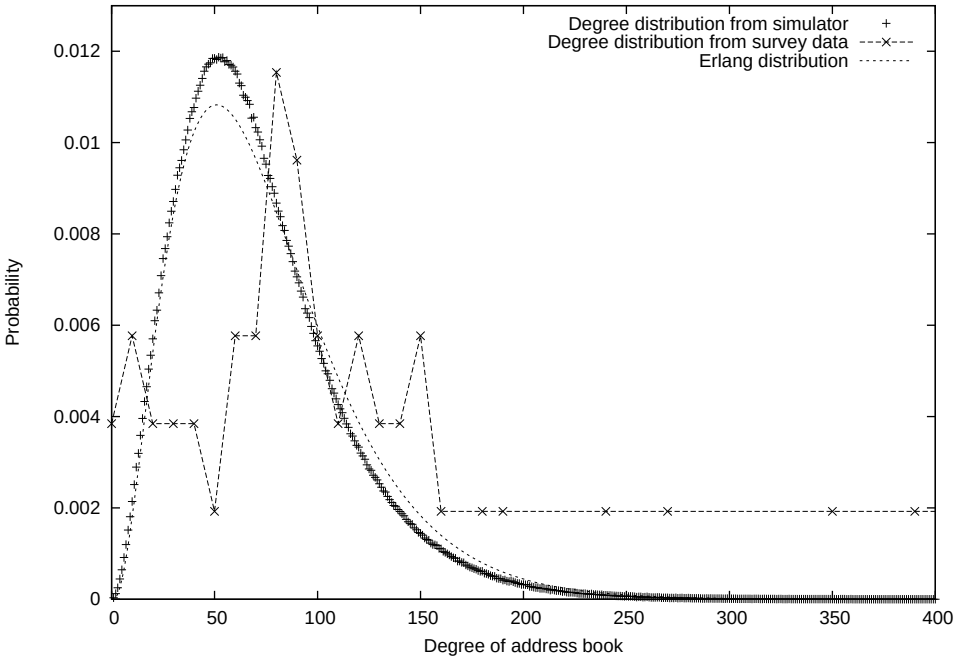

Figure 3.6: Erlang degree distribution created by formula, programmatically and scaled from survey data.

3.5.2 Node attachment

Another principal concern when generating network topologies is determining which nodes are connected to each other. In scale-free networks, the topologies are incrementally generated so that the highly connected "hubs" are chosen more often than other nodes. One way to generate these hubs is by assigning degree distributions randomly and then randomly connecting nodes to each other. However, a slightly more advanced technique uses a model of attachment so that nodes "want" to connect to popular nodes. The method is called preferential attachment and can be important when nodes have a global sense of popularity, such as routers on the Internet. Bu and Towsley [7] explore this idea and create an

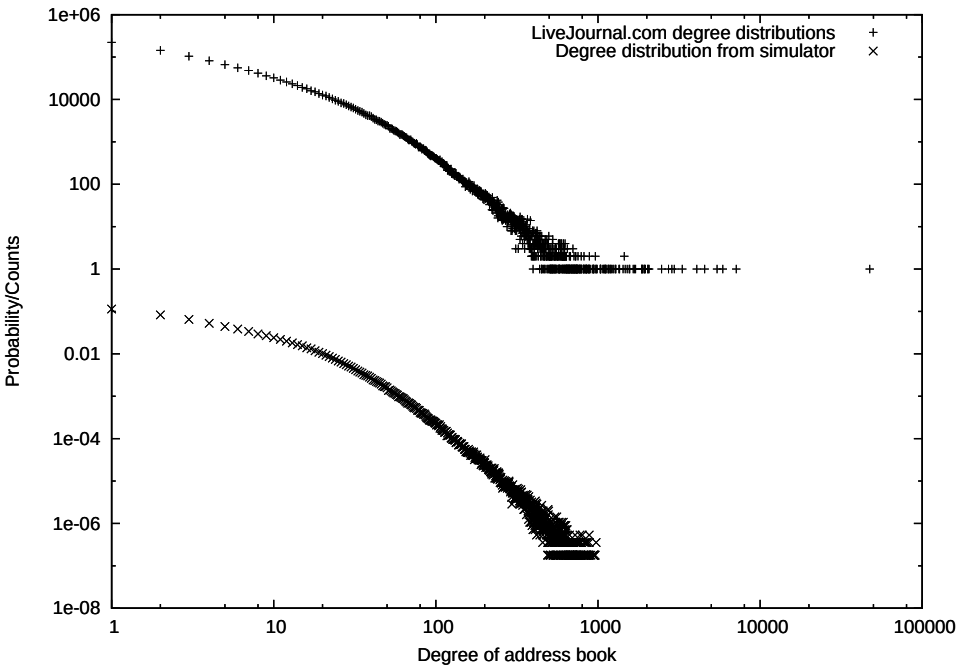

Figure 3.7: Log-normal degree distribution from LiveJournal.com data and created programmatically.

Internet topology that was more accurate than previous models. Thus, if node attachments are randomly wired together, the result may be an unstructured, and undesirable, topology. Alderson et al. [2] discuss the need to incorporate further practical constraints and attachment models when generating Internet AS topologies. They pictorially demonstrate the difference between topologies with the same degree distribution that, at the same time, have vastly different structural qualities. However, we reason that the preferential attachment model, a factor that helps to create hubs and structured topologies, may not be appropriate for cell phone networks. Realistically, a single person cannot be acquainted in daily life with hundreds of thousands of people, meaning there are no true "superhubs" that

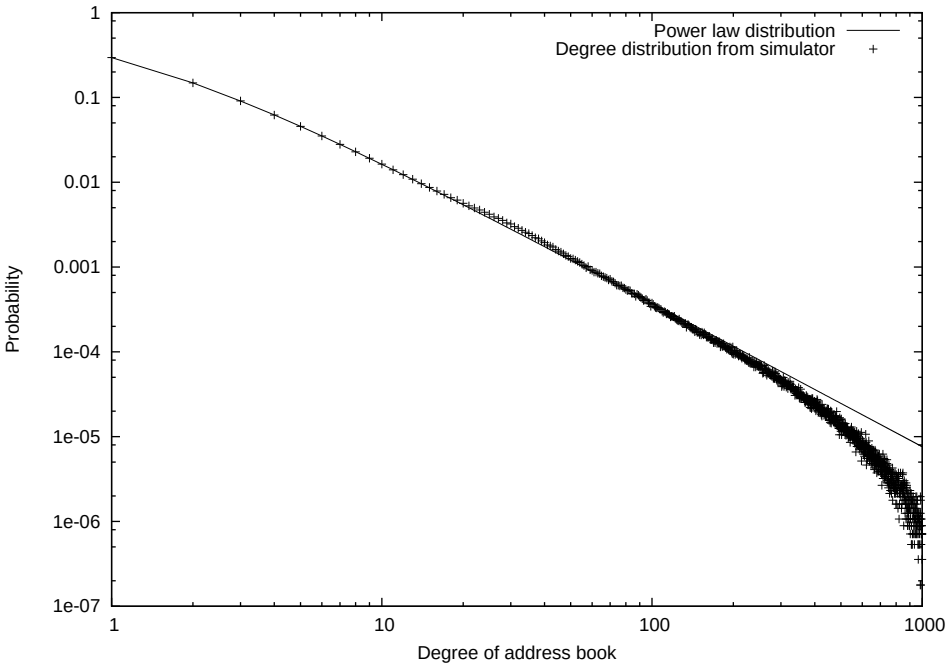

Figure 3.8: Power law degree distribution created by formula and programmatically.

have several orders of magnitude more contacts than others.

A more pertinent consideration than popularity, though, is the role of geography. Mobile phone users are more likely to communicate with people they are in regular contact with. In other words, the contacts in a user's address book will more likely be geographically nearby than distant. Liben-Nowell [25] correlated LiveJournal.com users with their geographic location based on zip code and compared that information to the location of their friends. In contrast to previous work, they found that the probability that two users were "friends" was proportional to the inverse of the number of people between them. Thus, when determining if two nodes are connected, the actual geographic distance is not as

important as the population that separates two nodes. Their results, based on U.S. population patterns, provide a more accurate picture of a social network. However, they also discovered this rule only held for about 70% of the relationships. The remaining 30% could not be characterized by the measure. We make use of both of these findings in our simulator to inform the choice of contacts found in address books. Thus, 30% of connections are made through random assignment. The other 70% are probabilistically assigned based on the number of people in between every cell. To programmatically accomplish this contact assignment, the simulator first chooses a grid cell using the inverse proportion rule. Then it chooses a mobile phone at random from that cell and assigns it to the address book of the cell phone. As a result, the address books of mobile phones are biased to include contacts to phones that are geographically nearby.

3.6 Simulator

To gain the greatest control over the experimentation process, we designed and implemented our own simulator capable of modeling the cellular phone network and the individual users connected to the network. It takes as input a file containing a list of variables that modify the behavior of the simulator and an XML file which specifies the topology of a cellular network. The simulator is single-threaded, object-oriented, written in C++ and comprised of over 4,500 lines of code.

3.6.1 Simulator design

The simulator is designed in a modular fashion that allows concrete "event producers," which model various scenarios, to generate various "events", which are maintained and scheduled by the simulator core. Each event producer maintains the necessary state associated with the specific type of event. For example, our simulator uses a communications event producer that is responsible for simulating the flow of data over communication channels offered by a cellular phone network. The use of event producers also allows for more than one virus to exist at a time, although we only model one virus for simplicity. In future work, we could also add a proximity based event producer that created events to model the interactions between mobile phones within a certain distance (e.g., via Bluetooth). The modular design also allows flexibility in how the simulator operates. For example, we could use actual trace data files to generate events at the same time that we are programmatically creating events for another type of scenario.

The simulator *runs* by querying all instantiated event producers, at each time step, whether there are new events available. The result of the query is a list of events which have occurred at that time in each event of the environments. Event producers can, of course, produce many events per time step. Events are basic objects which contain information that identifies the mobile device being infected and the device causing the infection. When retrieving an event, the simulator ensures that at least one device is infected with the malware (there may be events where two uninfected phones come in contact when, for example, they pass each

other in the street). Both phones may also already be infected, in which case it is a "non-event" and nothing further happens. However, if only one of the phones is infected, then the simulator processes the event by determining when the phone will actually become a member of the group of infected. Indeed, it is always the case that the victim phone will not immediately become infected. There exist various delays, such as bandwidth constraints, latency and user interaction, that cause the phone to actually become infected at a later time. These soon-to-be-infected phones are put in a queue and marked appropriately. When the infection has completed, the malware is essentially controlling the device and the phone is now able to infect others. At the end of each time step, we then record the number of infected phones, which is the primary output of each simulation.

Simulator configuration variables

The behavior of the simulator, and consequently the nature of the malware propagation, is controlled by a number of variables, outlined in Table 3.4.

Table 3.4: Simulator configuration variables

Variable Name	**Description**
scenarioFilename	The filename of the network topology to use when simulating communication between two devices.
timeLimit	The duration of the simulation in seconds.
timeIncrement	The length of each time step. A finer granularity can allow more events to happen in a shorter time span.
numberOfInitialInfections	The number of infections that are present when the simulation starts.
numberOfPhoneTypes	The number of phone types that exist within the simulation.
phoneTypeVulnerability	The probability that a certain type of phone is vulnerable to an exploit.
phoneTypeMarketShare	The percentage of all the phones belonging to a specific type.
probabilityOfUsersWhoIntervene	The probability a user will intervene to cause their phone to become infected.
probabilityPhoneHasAntiVirus	The probability that a phone already has anti-virus protection, and thus will never become infected.
payloadSize	The size of the malware in bytes.
spreadScenario	The mechanism through which the malware spreads between phones.
usesAddressBook	A boolean to determine if phones use their address book when attempting to contact devices.
topologyType	The type of address book topology to use in the simulation.
limitBW	A boolean to determine if bandwidth and capacity constraints should be used within the simulation.
randomDial	A boolean to determine if phones will randomly dial phone numbers in order to spread malware.

3.6.2 Simulator operation

When the simulator starts, it reads a configuration filename passed as a command line argument. This configuration file is a list of key-value pairs that specify how the simulator should behave under different circumstances. It parses the files and stores the variables in a hash data structure. The simulator then begins its initialization step. It begins querying these configuration parameters outlined in 3.6.1 to determine basic parameters, such as the time step granularity and the length of the simulation, as well as more advanced configurations, such as the probability that a phone contains an anti-virus application. It also loads the cell phone network topology, recreating the appropriate data structures in memory to model it. Section 3.4 discusses these components of the network topology in detail, but briefly, each of the components has a correlated data structure, which is connected with a link that specifies bandwidth and latency to another component.

The network topology file includes in it a representation of the underlying geography and population dynamics. The simulator uses these values to assign mobile phone users to different regions, as well as when creating the number of mobile phone users. When the simulator has loaded the population information it can begin to initialize each mobile phone with appropriate parameters and features. Many of the initialization parameters for the mobile phones require the use of a probability. For example, 30% of the market share may be held by an Ericcson phone. The remaining phones may originate from another manufacturer. To model this probabilistic parameter, the simulator chooses a random number that selects

which manufacturer created the phone. The simulator tests whether the random number is less than the assigned probability, in our example, 30%. Thus, if the random number was .27, then the mobile phone would be an Ericcson.

Perhaps the most important action when initializing the mobile phones is to assign the contacts that the user has in their address book. The address book plays a significant role in determining who the phone can communicate with when trying to spread malware. The simulator supports a number of address book topologies discussed in detail in Section 3.5.

The final initialized component is the malware. The malware object contains pertinent information that includes the size of the payload, the different vectors through which it can spread, and whether user intervention is required for propagation. Each simulation models only one malware component per execution. When a user is infected, the distinguishing characteristic is that they contain a pointer to the malware component. The malware component also determines how long an infection takes between two phones. This value depends on the data size of the malware and the maximum bandwidth which can be used through the vector. It is a general routine that can be used by any event producer that has a notion of bandwidth.

3.6.3 Event producers

Event producers are largely responsible for the interaction between mobile phone devices. We have currently implemented only one event producer — the

communication event producer. Although we experimented with a proximity-based event producer, which would have enabled malware to spread through Bluetooth for instance, we found that the orders of magnitude difference in spreading, coupled with relatively low usage of Bluetooth in the real world, made this a lower priority avenue to explore for our purposes (Section 2.2).

The communication-based event producer is intimately connected to the network topology. The foundation of all communication depends upon the hierarchy of the topology. When the simulator queries for new events at a time step, the event producer iterates over the list of infected phones idle at the time. Thus, it skips over phones that are currently communicating with another phone, as we assume there is no ability to multi-thread communication. We also check to see if enough bandwidth is available for communicating by validating the amount of bandwidth remaining at the radio cell level, the Node B to RNC connection and the RNC to SGSN connection. This check captures almost all cases of congestion so that we can short-circuit further processing, although there is still a chance that there exists congestion in the route between SGSNs.

When the communication producer identifies an idle phone with sufficient communication resources in its cell, the phone then chooses another device to contact. This device may come from their address book, in which case a random contact from that list is chosen. Of course, the malware will only try to reach a contact one time. The phone may also try to randomly dial a phone number depending on the configuration parameters. Our simulator allows the malware to communicate

through either a VoIP-like scenario or an MMS scenario.

VoIP event producer

In the VoIP scenario, the phone establishes a direct connection. If the target phone is busy communicating with another phone or if there is not enough bandwidth between the two phones through the provider network, a connection is not made. In this case, the infected phone attempting communication experiences a certain amount of delay that depends on the hierarchical connections within the network topology. This delay, in essence, reflects the need to look up and contact a support node within the hierarchy to determine the other phone's status. The infected phone would then be unable to perform any other operation until that amount of time had passed. However, if these conditions are met, the two phones establish a connection and determine the minimum bandwidth available to them. This value dictates the speed at which the malware will transfer and also marks both phones as being unable to perform other operations during infection. The communication producer then returns an event to the simulator indicating the phones involved in communication.

MMS event producer

In the MMS scenario, the communication producer performs a similar process to model infection, except that the infected phone sends a message to the central MMS server in the topology. The ability to send a MMS message at a time step depends on two factors: the bandwidth to the MMS server and the capacity of the

MMS server. In real network topologies, most MMS servers have a highly-limited capacity shared between millions of users. In our simulator, if the infected phone is not able to send an MMS message at a certain time step, the phone then becomes unavailable for a time that depends on the latency it takes to reach the MMS server level within the network hierarchy. If, however, there is remaining capacity, a message is sent and the phone to which the message is addressed is informed there is a new message.

The user, however, may not immediately choose to open the message. We simulate a configurable amount of *wait time* that the user experiences before answering a MMS message. We model this wait time using a bi-modal distribution reflecting anecdotal evidence that users either check new MMS messages relatively quickly, or they check messages a short while after receiving them. We have chosen to model this distribution as a mixture of Gaussians, where the range of response times for retrieving an MMS message is one hour. The two means are centered at 20 seconds and 45 minutes. Figure 3.9 depicts the bi-modal wait time distribution plotted as a probability density function. The first Gaussian has a standard deviation of 200 seconds. The second Gaussian has a standard deviation of 250 seconds. The Gaussians are mixed equally with weights of 0.5. The first Gaussian has a higher peak because we do not allow negative wait times. Section 4.3 shows the results of incorporating wait time on malware propagation.

After the infected phone transmits the MMS message, it is free to transmit other messages. The target phone that will receive the message remains dormant

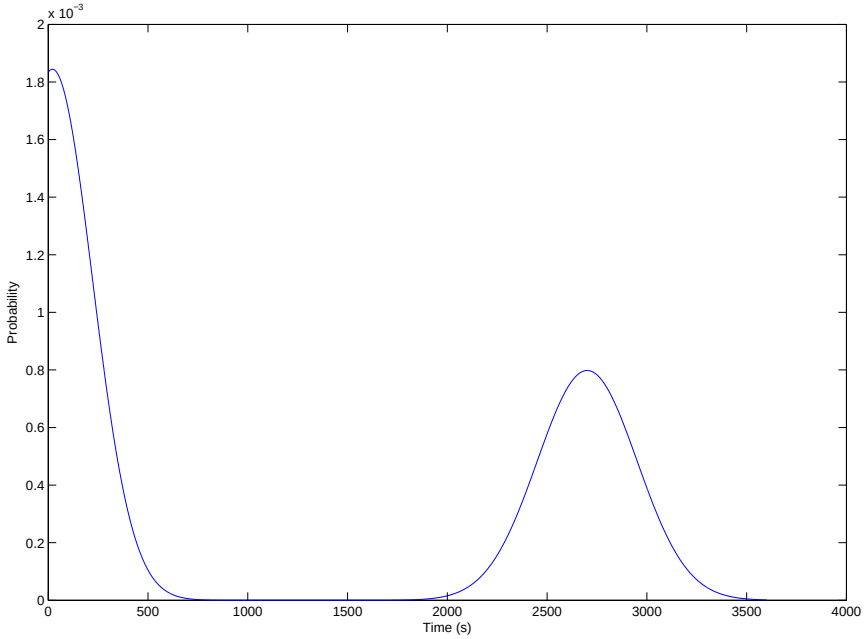

Figure 3.9: Probability density function that determines how much time will elapse before a user retrieves a MMS message.

until the wait time elapses. At this point, the simulator informs the appropriate event producer that the phone is about to start an infection. The communication event producer then determines if there is capacity and bandwidth available to retrieve the message from the MMS server. If there is not enough at that time step, the phone waits until the next time step to try again. If it is successful, the phone enters a state of being infected that lasts until the MMS server transmits the malware to the phone.

3.7 Acknowledgments

Section 3.4 was, for the most part, written by Michael Liljenstam of Ericsson research. He was also responsible for creating the network topology generator.

4

Experimental results

In this chapter we use our simulator to evaluate various aspects of malware propagation on mobile phone networks. We compiled our experimental results by running each scenario five times and averaging the result. The lengthy amount of time for each simulation run to complete, plus the relative consistency of the results, balanced accuracy and practical time considerations. Each simulation used a one-second time step and all runs start with one infected phone. Most simulations run for 43,200 seconds (12 hours), which in most cases provided a wide enough view of the infection behavior. As mentioned previously, we used a cellular network topology based on an underlying population distribution. The area modeled is the Boston metropolitan region.

We also use an Erlang social network topology to inform decisions about the address book of cell phone users. Although, we examine a few scenarios with other social network topologies, we focus primarily on the Erlang distribution because it

better matched our survey data and it provided a realistic, fully connected topology. Unless otherwise noted, malware uses this address book to contact other potential infectees until the list has been exhausted. At that point it attempts to contact phones by guessing phone numbers. However, the success of randomly guessing a phone number that results in reaching a cell phone was not immediately clear. Although it is plausible that there may be segmentation within the phone number space which can reduce uncertainty, cell phone numbers do not follow logical patterns for the most part and there are no special prefixes designated for cell phones. To calculate the success of random dialing, we divided the total number of cellular subscribers in the United States at the end of 2005, 207.9 million [40], by the space of phone numbers, 10^{10}. Thus, the probability of successfully choosing a cell phone number at random is 2.079%. The consequence is that random dialing often has little effect on the overall spread.

We begin by examining how malware spreads by first looking at unconstrained, simple cases and then subsequently add complexity and constraints and introduce exploit vectors — VoIP and MMS — that differ significantly in their behavior. We then experiment with various methodologies that exacerbate and quicken the spread of malware. We then show the effects of various counter-measures that could be deployed by network operators. Finally, we discuss the challenging topic of cleanup in the mobile phone environment. In general, we present results that show the percentage of the total population infected as a function of malware propagation time.

4.1 Unconstrained propagation

We first look at malware operating in an unconstrained environment, one in which there are no limits on bandwidth or capacity. Figure 4.1 displays the rate at which malware would spread through the network infrastructure given various address book topologies. The "complete" case emulates an address book that contains every contact in the population, and is consequently the quickest to spread, reaching the 90% of the infected population after 86 seconds. The log-normal and power law cases flat line because the topologies created a disconnected social network. The Erlang topology is expansive enough to consistently generate a fully-connected graph and reaches 90% infected after 167 seconds, about twice as long as the complete address book. The characteristic "S-curves" seen in Internet outbreaks is also present here, confirming that our simulated infection procedure operates as expected.

4.2 Voice over IP scenario

The Voice over IP scenario assumes that malware is able to use an exploit within the software modules dedicated to handling packetized voice data. To spread, it dials a phone number. When that phone answers, it sends the payload over the channel. It resembles Internet worms in that there is little a user can do to stop an infection once it has begun communication with a user. We first look at the basic infection case for different address book topologies in Figure 4.2. Propagation no

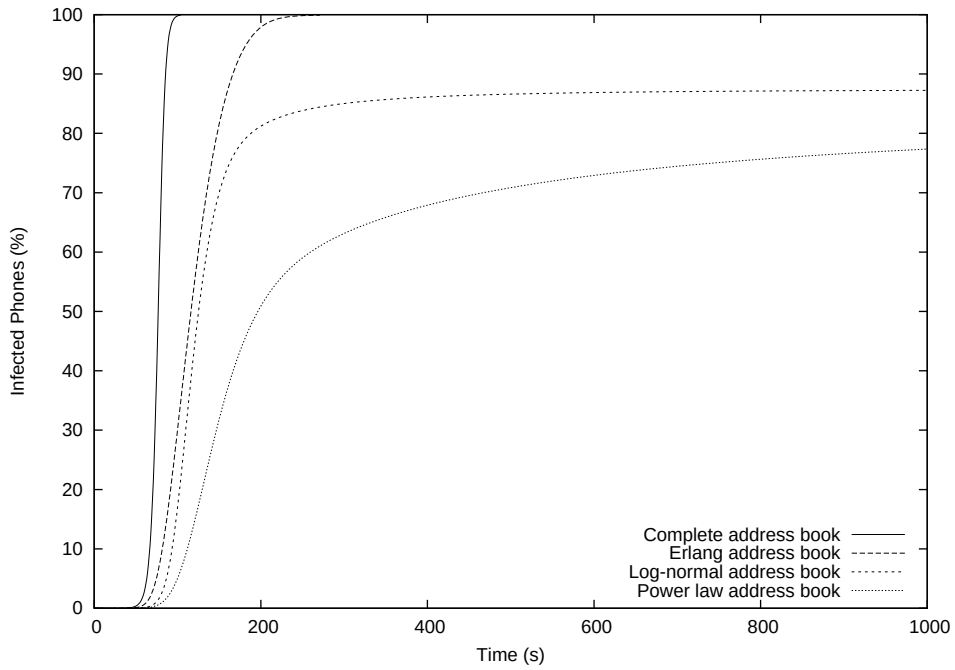

Figure 4.1: Malware infection with no bandwidth or capacity constraints.

longer has the characteristic "S"-curve, and is much slower than the unconstrained case due to bandwidth constraints of the network. Even with the complete address book (fully connected contact topology), propagation took 13,926 seconds to infect 90% of the population, 162 times as long as the unconstrained case.

The contact graphs resulting from the different models of address book connectivity also have a substantial impact on malware propagation. For example, compared to the nearly 14,000 seconds required by malware to infect 90% of the population with a complete address book, malware propagating with address books modeled using the Erlang, log-normal, and power law distributions only reached 71.1%, 47.6%, and 31.9% of the population, respectively, at the same point in time.

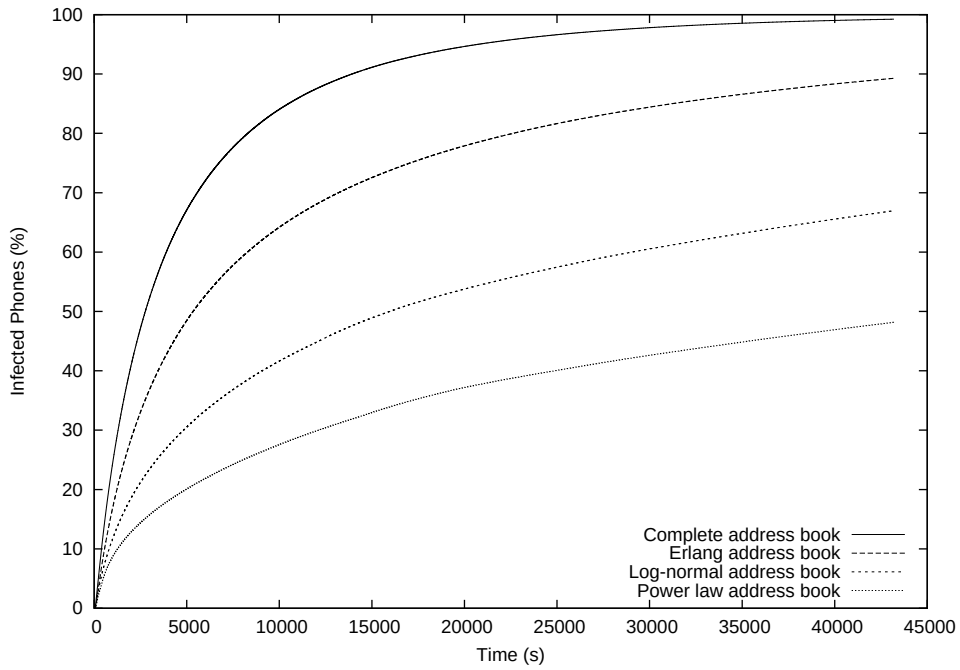

Figure 4.2: Infection rate for address book topologies using a VoIP scenario.

However, if we look at the infection rate on a log-scale time axis, seen in Figure 4.3, the "S"-curve appears again, demonstrating that there is an epidemic threshold, but that it takes a significant amount of time to reach. The characteristic curve appears regardless of the social network topology, although the log normal and power law topologies would most likely flat line at approximately 90% and 80% respectively, as evidenced by the unconstrained propagation in Figure 4.1.

As a point of comparison, it is useful to know the absolute rate at which malware could spread under the VoIP scenario. We can perform a few back of the envelope calculations to produce a rough estimate. We start by identifying that the RNC to SGSN link is the main bottleneck, since that is a 100Mb/s link, while

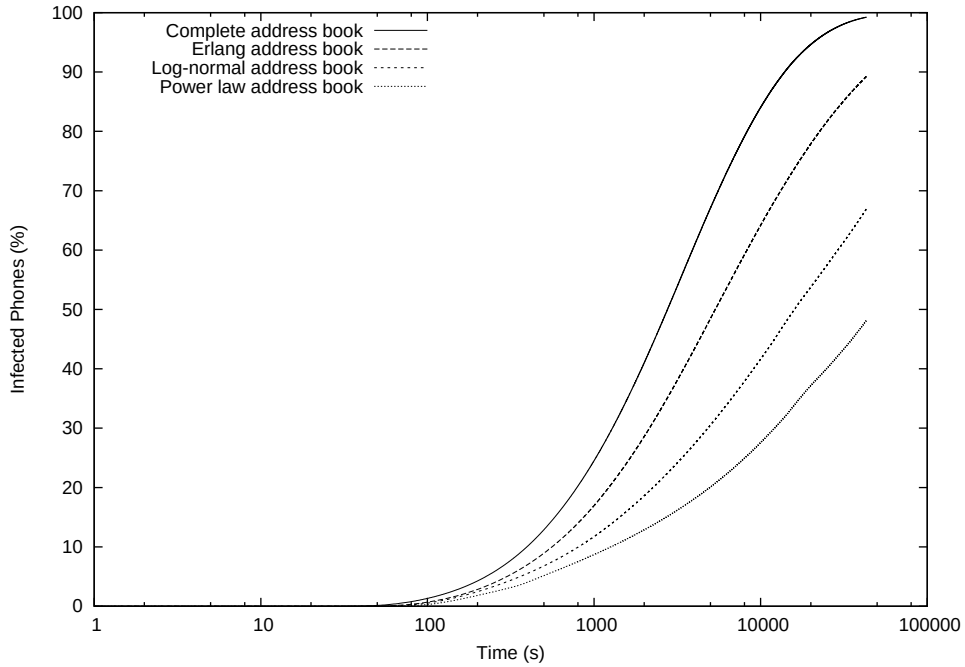

Figure 4.3: The infection rate on a log scale time axis.

there are approximately 256 Node B's connected to one RNC at 2Mb/s (meaning they could feed 512Mb/s into an RNC). The maximum user bandwidth allowed is 384Kb/s, and, to infect a phone, bandwidth needs to be used on the source path as well as the destination path. Thus, only half of the RNCs can be used at the same time to infect, assuming the perfect case. As a result, 6,250 phones can transmit malware at the same time. If we divide this number by the total population, we need 882 rounds of perfect network usage to infect all the phones. With each infection taking 2.605s to transmit the payload, the entire population can be infected after 2,297.6 seconds. For comparison, in the VoIP case described previously, the malware has only infected 31% at this same point in time.

The growth curves show that bandwidth becomes the major limiting factor very quickly in the lifetime of the malware. One might intuitively suspect that bandwidth at the radio cell level presents the most significant hurdle. However, measuring average congestion across the network topological hierarchy revealed that the RNC to SGSN link is actually the primary bottleneck. Figure 4.4 shows the average congestion when using the Erlang address book topology for the radio cell level, the Node B to RNC link and the RNC to SGSN link. We measure average congestion by summing the amount of bandwidth used across all elements at each link and dividing by the total amount of bandwidth available. The measurement provides a rough network-wide gauge of congestion. Because of the large fan-in ratio from Node Bs to RNCs, there is a large flow of traffic constantly arriving at each RNC. The RNC then has to move this traffic to an SGSN. However, the total traffic from the Node Bs overwhelms the total bandwidth available to the RNC. Interestingly, the congestion peaks very early, within approximately 5–7 minutes. Then congestion decreases slightly over time even though more phones become infected.

This result is somewhat counter-intuitive and deserves further explanation. Logically, it would seem that as more phones become infected, they will in turn try to contact even more phones, and therefore add to congestion. However, we have defined that only a successful connection attempt uses bandwidth. A success occurs when the phone is able to make an end-to-end connection with another phone. If there is too much congestion or the phone busy, then the infecting phone

does not consume bandwidth. As noted in Section 3.1, we assume that there is a separate control/signaling channel that handles call-setup before the actual payload data is sent. Thus, calls that do not connect, do not use data bandwidth.

Congestion decreases for two reasons. First, although more phones are attempting to dial, they are more likely to dial phones within heavily populated/congested areas. Consequently, they cannot reach their intended victim because of congestion and do not add to further congestion. Thus in dense regions, congestion remains high. However, in sparsely populated regions, congestion goes down, because those phones are likely attempting to contact victims in congested areas. The result is that average congestion goes down, because of the mix between heavily and lightly congested regions. The other reason for the decline in congestion is the increasing number of phones that have completed contacting the entries in their address books and have begun randomly dialing. Since the chance of actually reaching a cell phone is low when randomly dialing (2%), those attempts generally do not add more congestion, as they do not send the malware payload.

A related question is how soon due users experience congestion. Although we assume that the network topology carries no other traffic except the payload data, in reality there would be a certain operating load. This load would only serve to limit the malware's potential to spread quickly. However, even when malware is just starting to propagate, phones would begin to experience congestion and unavailable service within a short amount of time, especially in the areas where the malware appears.

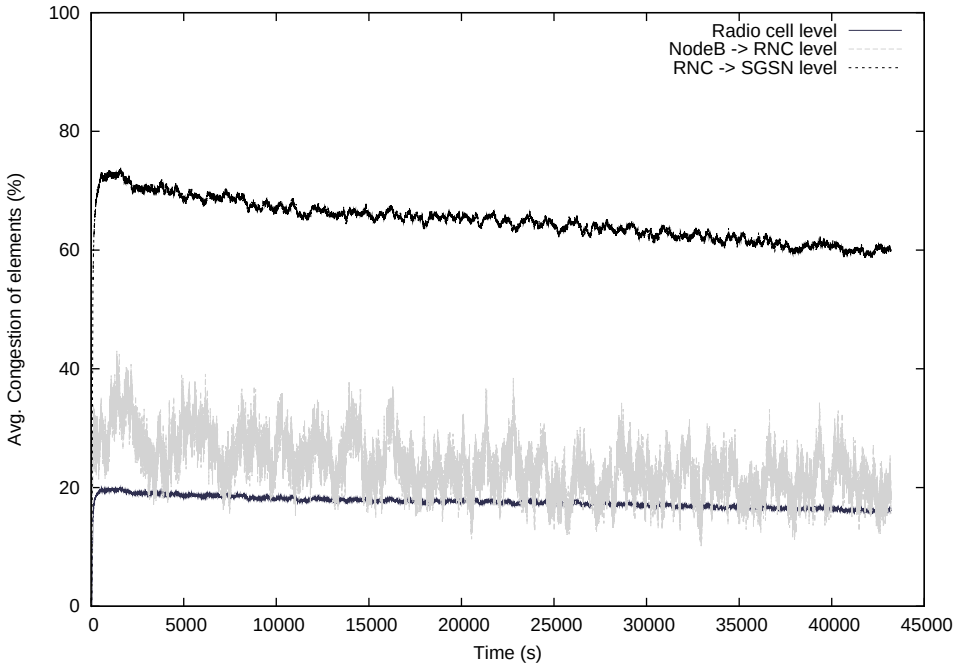

Figure 4.4: Average congestion levels over network elements in a VoIP scenario.

Figure 4.5 shows the percentage of infected cell phones who attempt, and are denied, communication due to unavailable bandwidth — any other phone trying to use VoIP service, infected or not, would experience similar congestion over time. Figure 4.6 shows a related measure, the absolute number of phones denied service over time. Both figures illustrate two sides of the same picture. Early on there are few phones trying to communicate. The ones that do get through are concentrated by geography due to our social topology generation. The remaining phones, also located in nearby regions, are denied service. Thus the percentage of phones denied service is high, despite there being relatively few infected phones. Over time, the number of infections increase and spreads out to cover the wider region. Although

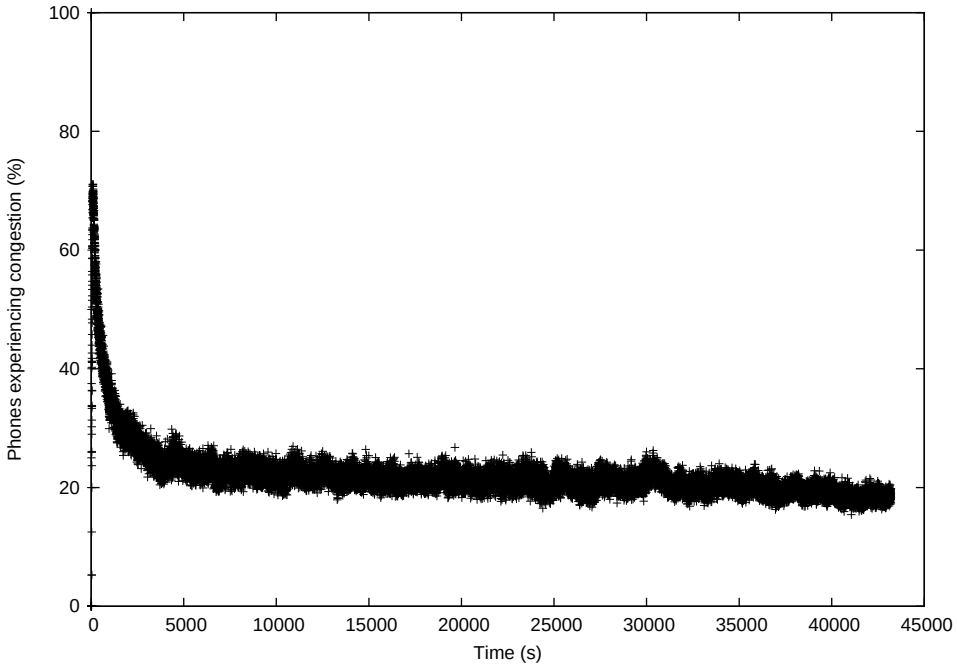

Figure 4.5: Percentage of infected phones experiencing congestion when they try to communicate.

the absolute count of those denied service blossoms, it actually represents a smaller fraction of the overall phones attempting communication, because more succeed as they are located in different areas.

An interesting corollary to the finding is that phones in heavily populated regions, such as city centers, are less likely to be infected sooner because bandwidth to their areas is quickly exhausted. Meanwhile, devices operating in sparsely populated regions have more available bandwidth and will be more likely to become infected early, consequently infecting others sooner as well. Another measure related to the user's perspective is the correlation between the number of contacts in a phone's address book and the time at which it becomes infected. Since we use

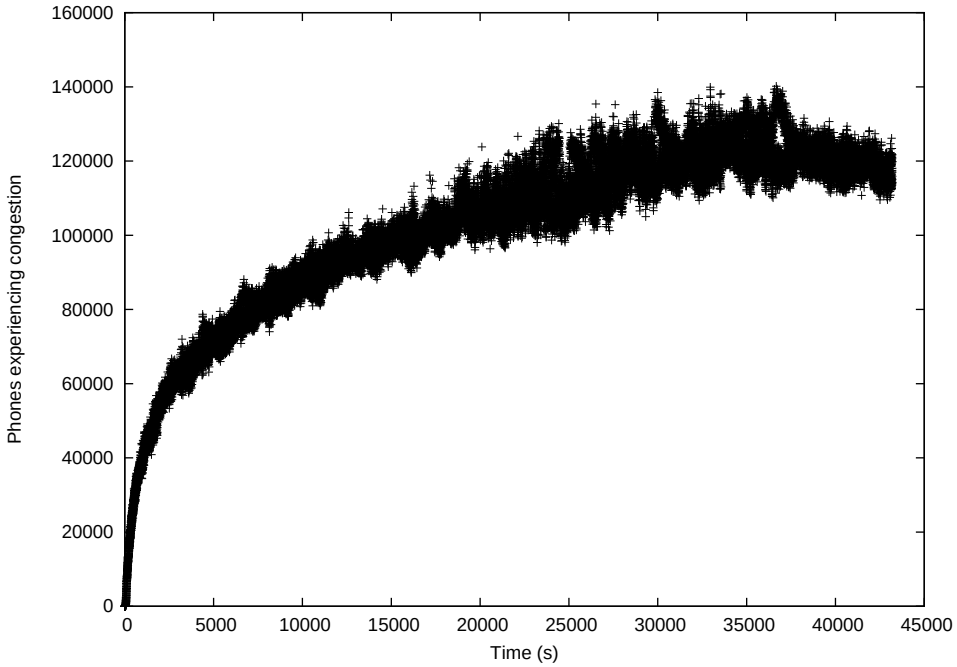

Figure 4.6: Absolute count of infected phones experiencing congestion when they try to communicate.

bi-directional connections, those with more contacts will in turn be reachable by more phones and will likely be reached sooner than a phone with relatively small address book. It is an interesting, though unexplored question, how uni-directional links, investigated by Newman et al. [39], would alter this observation. Figure 4.7 illustrates the phenomenon through a scatter plot of the number of contacts possessed by a phone and the time at which they became infected. There is a clear downward trend that suggests those with fewer contacts will remain uninfected longer.

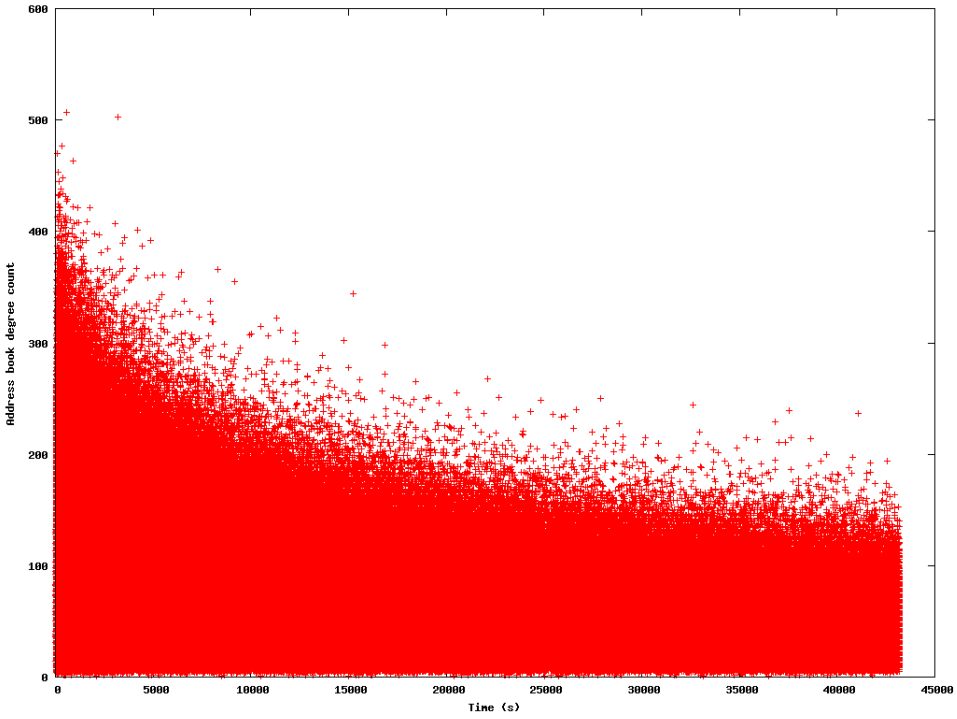

Figure 4.7: A scatter plot depicting the time at which a phone was infected and the degree of their contacts.

4.3 Multimedia Messaging Subsystem scenario

The Multimedia Messaging Subsystem (MMS) is a means for sending messages with attachments, such as photos or videos. If malware were to spread through the use of MMS messages, then a message would be generated on the infected phone and routed to a centralized MMS server. This server often has a severe capacity constraint (Section 3.4.4), which limits the total number of messages per second which can be sent or received. The phone to which the message was addressed would then download the message from the server. As mentioned in Section 3.6.3, we also employed a bi-modally distributed wait time so that each

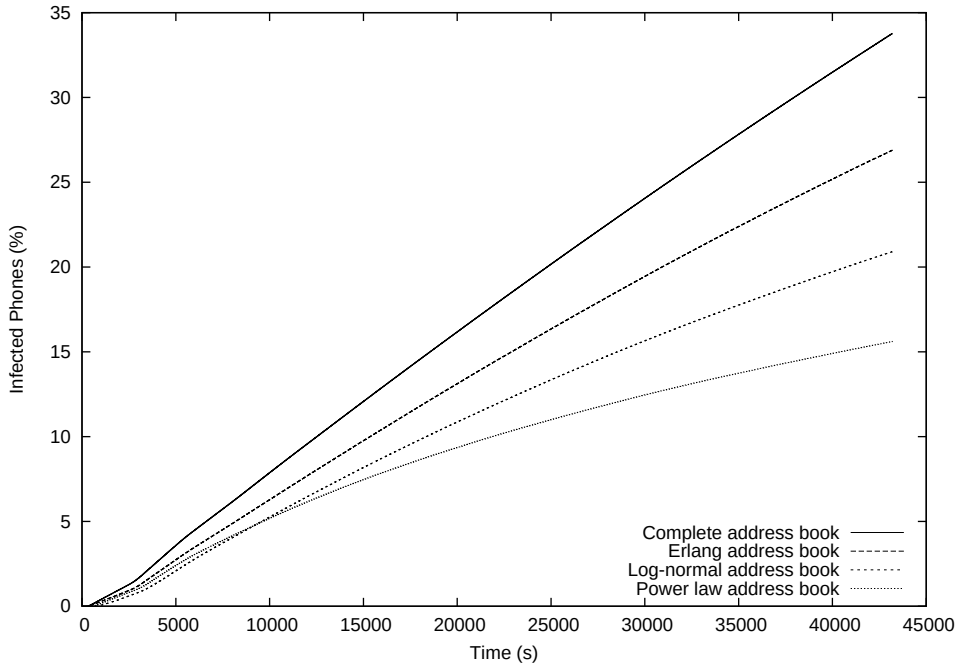

Figure 4.8: Infection rate for different address book topologies with MMS-enabled malware.

user receiving a message would delay for some period of time before attempting the download. Figure 4.8 shows the propagation characteristics constrained by the MMS configurations for different address book topologies. It shows that the social network does have an effect on the rate at which malware can spread, but the differences are not as pronounced as in the VoIP scenario. Indeed, if senders and receivers of MMS messages equally shared the capacity, the maximum rate achievable would be 50 infection/second. Compared to VoIP, MMS malware takes a dramatically longer period of time to infect the same number of devices.

Figure 4.8 shows a slight bump in the infection curve near the beginning, approximately at 45 minutes. This time corresponds to the second mean of the

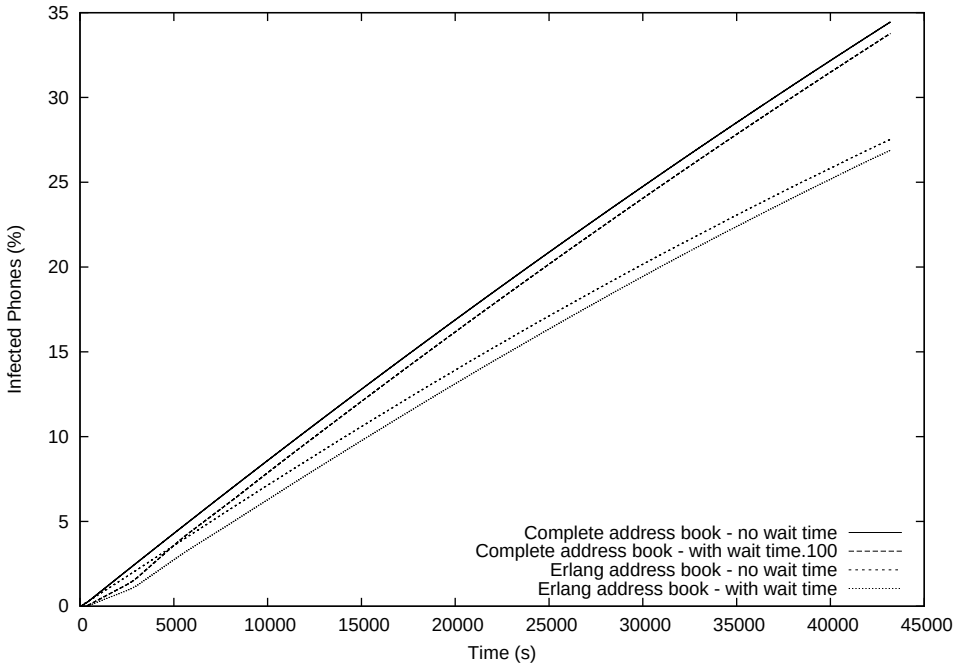

Figure 4.9: Infection rate comparing wait times vs. no wait times for an Erlang address book distribution and complete address books.

bi-modal wait time distribution, which is when the malware achieves the full rate of infection. Since MMS requires users to react to the messages to continue propagation, how much of an effect does the wait time have on the overall spread of the malware? Figure 4.9 demonstrates that the added delay does not significantly alter the overall propagation, aside from nominally slowing down the ramp up period. Indeed, once there are enough phones infected, the MMS server reaches capacity at almost every time step regardless of wait time.

Of the relatively small number of malware packages aimed at cell phones so far, they have typically required user intervention before the phone actually becomes infected. For example, the Commwarrior virus, which spread through MMS, used

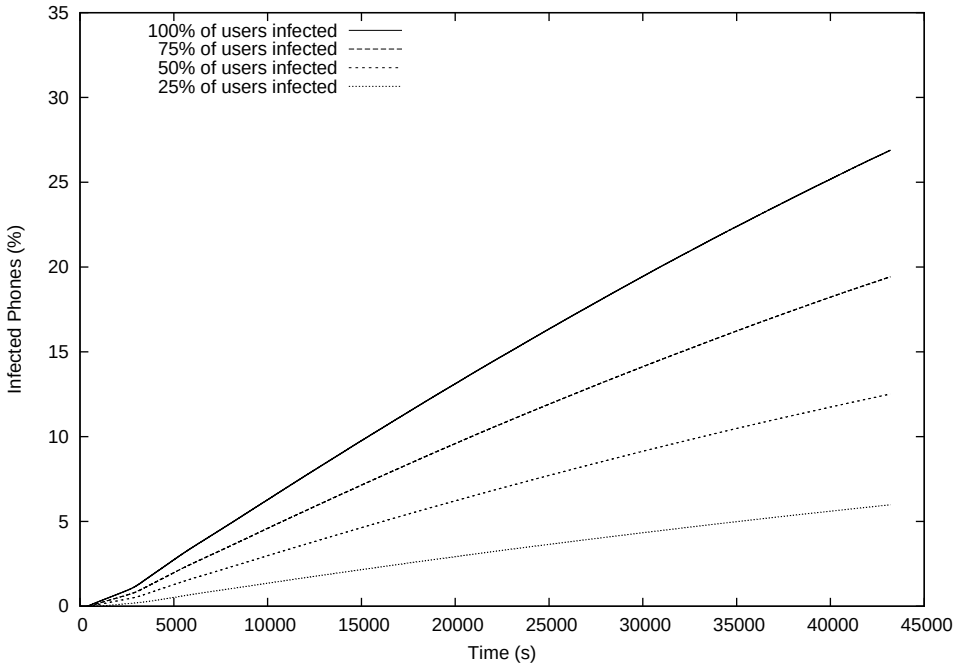

Figure 4.10: The effect of user interaction on malware spreading through MMS.

certain text messages to convince the user to open the attachment, such as "Game from me. It is FREE !" [62]. To understand what effect user interaction had on the spread of such a virus, we ran a simulation where a user successfully interacted with the virus with a probability of 100%, 75%, 50% and 25%. Figure 4.10 shows the results of malware propagation under these different scenarios. The results show that the rate of infection scales down accordingly.

In all MMS scenarios thus presented, it is immediately clear that the bottleneck resides in the capacity of the centralized MMS server. Presumably, as normal usage of the MMS service rises over time, operators will provision their networks accordingly. In Figure 4.11 we look at future configurations of the MMS scenario

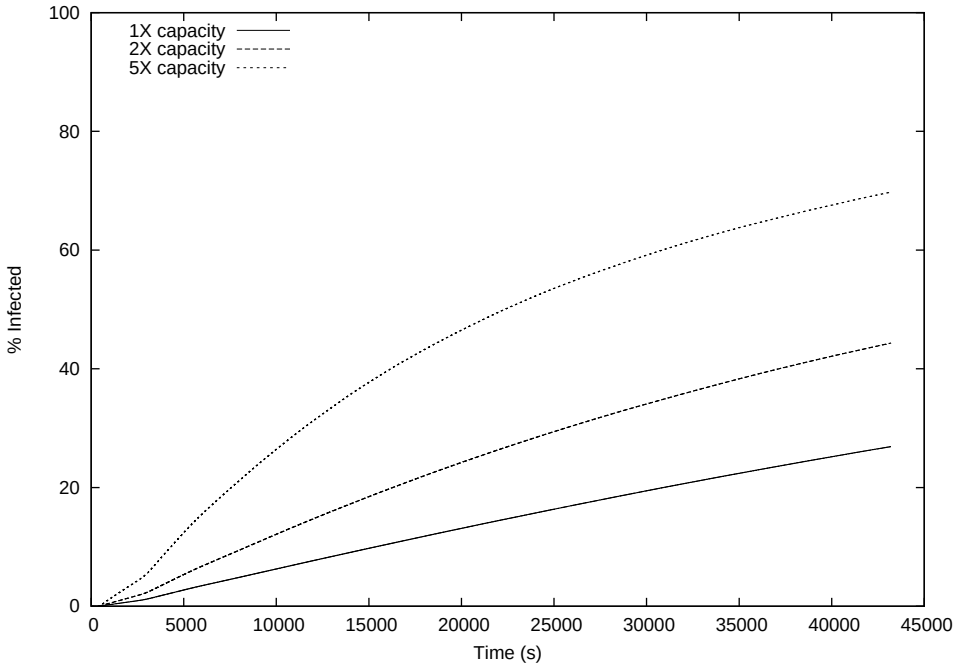

Figure 4.11: The effect of upgrading capacity on the MMS server begins to make MMS look more like a VoIP scenario.

where capacity has been increased by a factor of two and a factor of five. The rate of infection is obviously faster, but importantly we see that, at five times present capacity, the effects of bandwidth begin to become important. Even if a centralized server can handle more messages per second, it is still connected through a bandwidth-limited link, albeit a relatively fast link.

It is also useful to know the maximum rate at which malware would spread assuming perfect conditions in the MMS scenario. The nominal 100 messages/s capacity, however, makes it challenging to know the best infection technique to use. At the beginning, malware might spend all its time sending, while towards the end, the capacity would better be used for receiving infections. To simplify, we

assume an equal usage of the capacity, so that 50 infections occur per second, while the other 50 messages are used for sending malware. We also assume that every cell phone user receives the MMS instantly and always becomes infected. At 50 messages/second, the malware would need 112,426s to infect the entire population, assuming each infection took 1 second to reach the victim. Our simulations end at 43,200s, which is 38.4% of the time needed to reach the 100% level total. As comparison, the MMS scenario, using an Erlang address book topology, without any wait time, reaches a 27.5% infection level at the end of the simulation, a difference of about 11% from an optimal spread. The discrepancy occurs due to longer propagation times due to congestion in the network and duplication of effort because infected phones are contacted more than once.

4.4 An attacker's delight - Engineering malware for speed

Internet malware has the potential to use various mechanisms to speed up the rate of infection. Weaver et al. [48] describe many of these methods, such as using hit lists and effectively dividing the known address space to avoid dead ranges. In the cellular phone realm, these methods are either nearly impossible to achieve — the address space is difficult to know — or relatively meaningless — there are no "super-nodes" to use in hit lists. Instead, the attacker can try to leverage knowledge of the constraints of cell phone networks to engineer a worm that can

spread more quickly. We present a number of techniques that could be employed by malware authors and evaluate their potential to cause harm. We examine methods for both MMS and VoIP scenarios using the Erlang social network topology model.

4.4.1 Using only the address book

One approach to speed up the spread of malware might realize that congestion and capacity are significant bottlenecks, and that if the malware stopped after communicating with all the contacts in the address book, it could reduce congestion. However, Figure 4.12 shows that this policy has little effect — the two curves are nearly indistinguishable. The social network topology is so well-connected that by the time most phones reach the end of their address books, the infection phase in nearly over. Figure 4.13 illustrates another point of view of the phenomena and shows that only 5.3% of the total population has started to randomly dial by the end of the simulation. In the MMS scenario, the server capacity plays a critical limiting role that cannot be avoided by this technique either.

4.4.2 Sleeping after successful infections

Another approach to alleviate congestion so that more nodes may communicate would be to sleep for some period of time after a successful infection. This approach could potentially allow other phones to communicate with their contacts. Figure 4.14 depicts the infection level if phones sleep for ten seconds after a successful infection, showing that the technique results in a deleterious effect to the

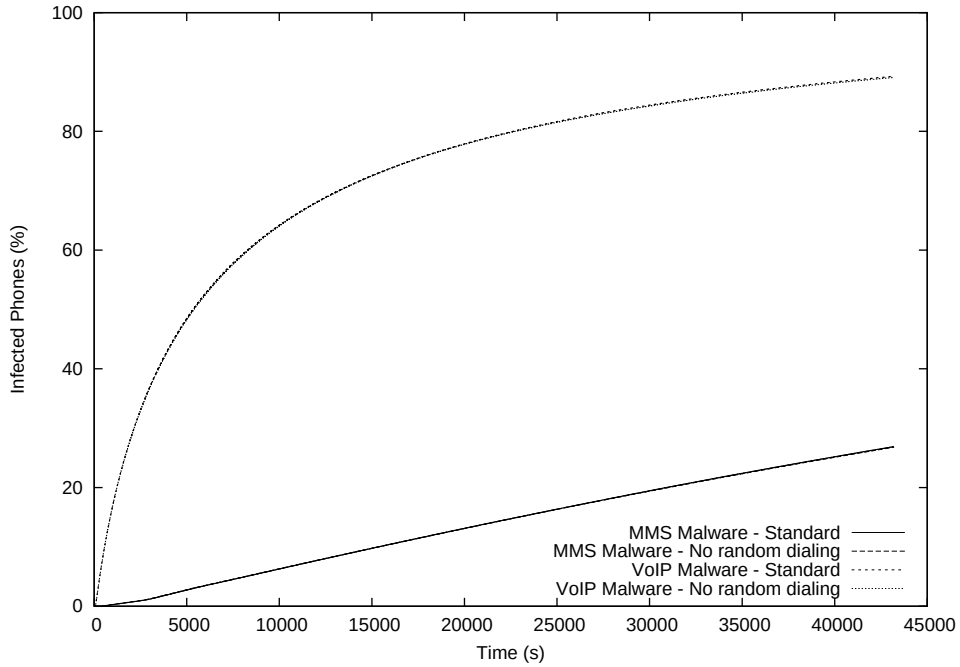

Figure 4.12: Infection rates for MMS and VoIP malware that does not attempt to randomly dial phone numbers. Note the "no random dialing" curves are occluded by the "standard" curves.

overall spread by slowing down the rate of infection. The conclusion is that if a phone has bandwidth to send in the present, it will likely be able to send in the future. By sleeping, phones in lightly congested areas do not make use of available bandwidth, while heavily congested areas still remain congested.

4.4.3 Transferring contacts

More advanced malware might distribute contacts between phones that have just exchanged code. This technique would have the benefit of creating larger address books (motivated by the complete address book in Figure 4.2), while having the potential to reduce duplicated effort in contacting already infected phones. We

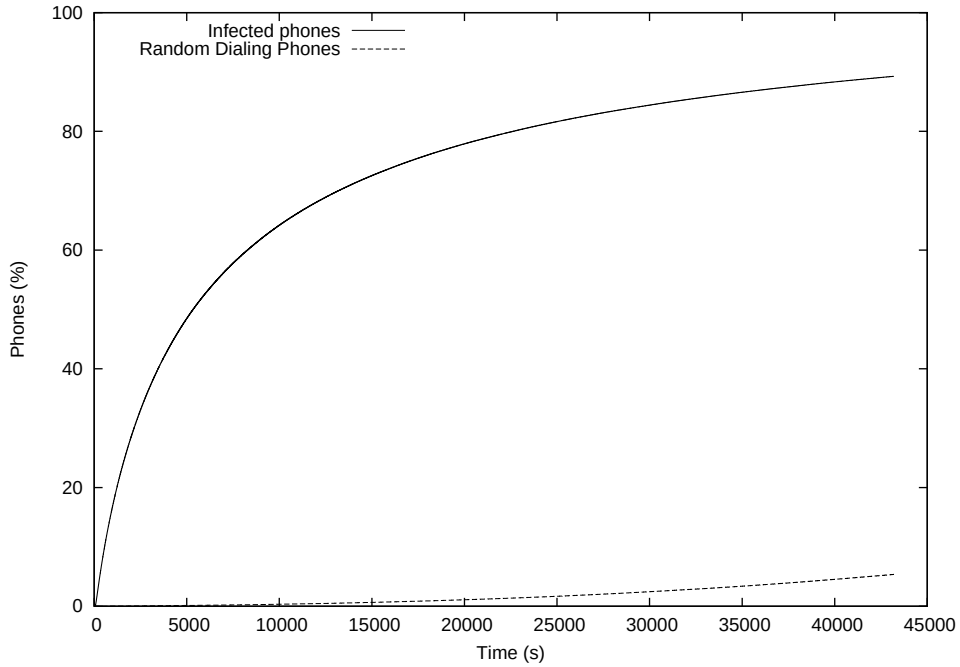

Figure 4.13: The percentage of phones that have completed scanning through their address books

implemented this feature so that, upon infection, the two phones evenly divide the contacts who have not yet been contacted between each other. Figure 4.15 demonstrates the effectiveness of the technique. The VoIP scenario is able to achieve a 90% infection level at nearly half the time it takes the baseline malware to do so. The MMS scenario does better than it could without optimization, but is still limited by the central server.

4.4.4 Wait when encountering congestion

An approach that takes advantage of the knowledge of bandwidth constraints can do significantly better if judiciously applied. Figure 4.16 illustrates malware

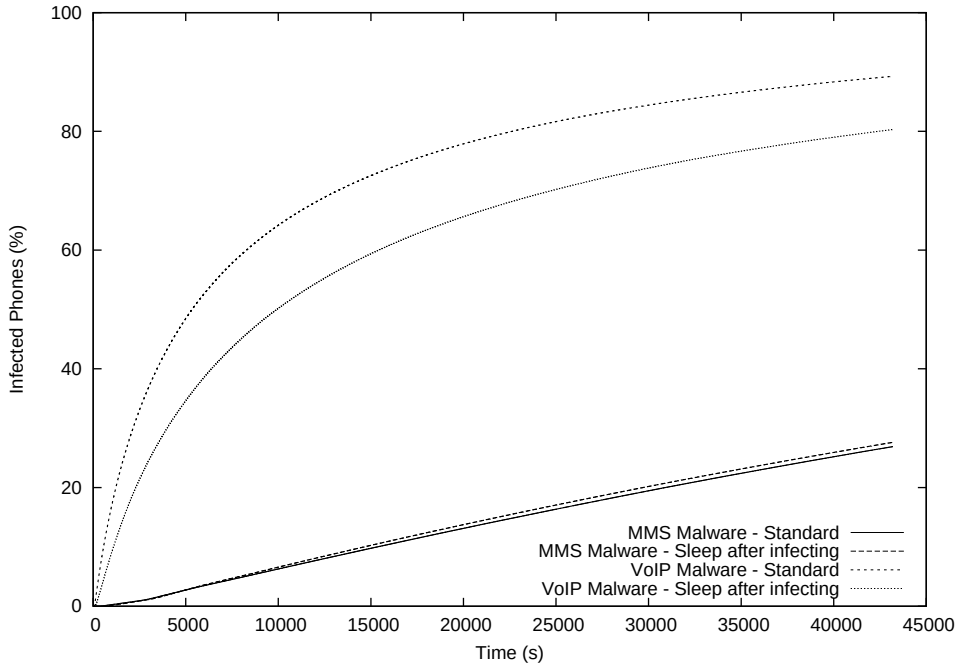

Figure 4.14: Infection rates for MMS and VoIP malware that sleeps for a period of time after a successful infection.

which sleeps for 10 seconds if it encounters any congestion — either a busy phone (one that is currently in contact with another phone) or a network unable to offer service. The result is a dramatic increase in infection potential in the VoIP case. A 90% infection level is achieved nearly four times as quickly as the standard malware. It shows that malware can affect congestion within the network to its own advantage. The MMS scenario however is aided very little, mostly because at every time step, the server will always reach maximum messages per second capacity long before bandwidth becomes a bottleneck.

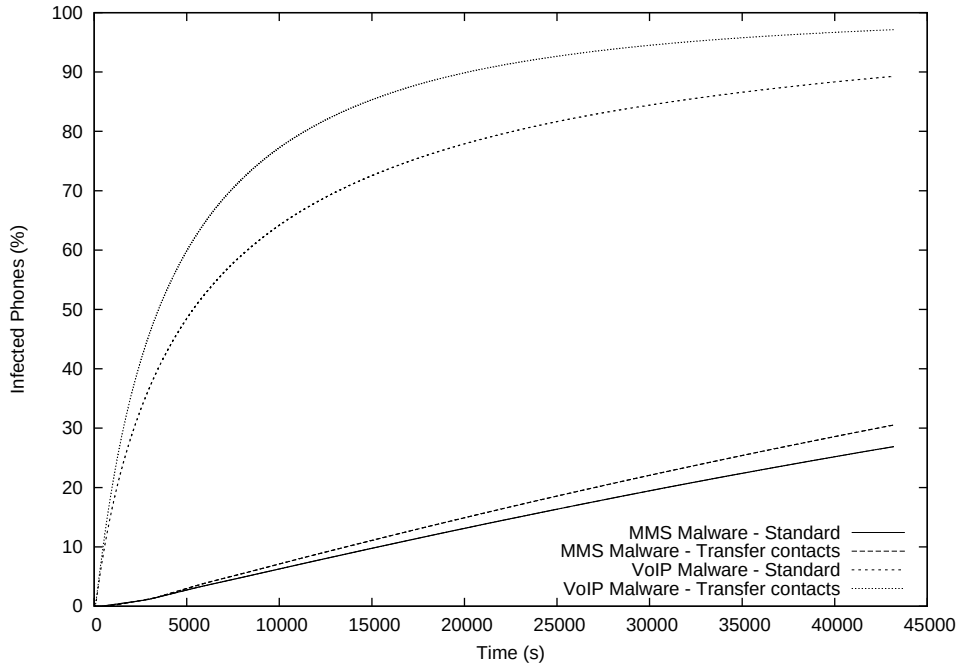

Figure 4.15: Infection rates for MMS and VoIP malware that transfers contacts between phones.

4.4.5 Avoiding congestion and transfer contacts

A clever malware creator could combine techniques that show promise. Avoiding congestion through waiting alleviates congestion, while transferring contacts more evenly divides the workload of contacting new phones. Figure 4.17 demonstrates the result from VoIP-enabled malware which uses both techniques. The individually applied techniques are also shown for comparison. The results show that avoiding congestion contributes the largest increase in propagation speed, and that dividing contacts further improves propagation although not as substantially. The MMS scenario benefits very little, however, and is not shown.

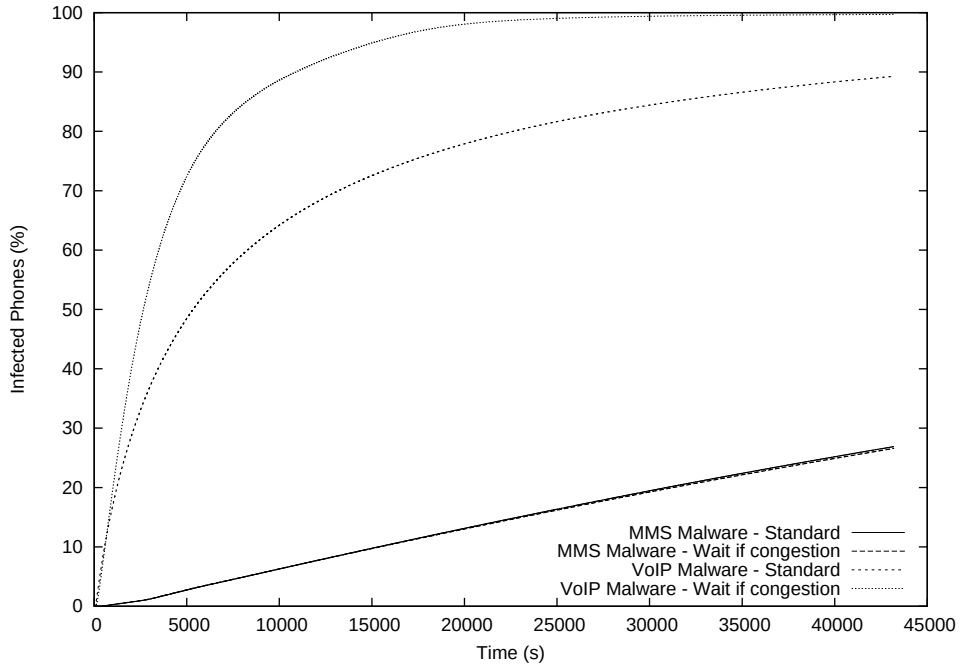

Figure 4.16: Infection rates for MMS and VoIP malware that avoids congestion through waiting. Note that the MMS curves are on top of each other.

4.4.6 Tackling MMS constraints

A MMS-enabled virus is an attractive target for malware creators. It has advantages over a VoIP worm in that it is nominally easier to create — it can rely on social engineering rather than a software vulnerability — and, moreover, both phones do not need to be powered on at the same time (a consideration that we have ignored in our simulations). However, it is clear that the capacity of the MMS server makes it difficult to spread rapidly. In the most virulent case, the malware could only hope to achieve an infection rate that was 50% of the server capacity at every time step, since phones must send and receive using the same constraints. Due to limited address books and duplicated infection attempts, it is difficult to

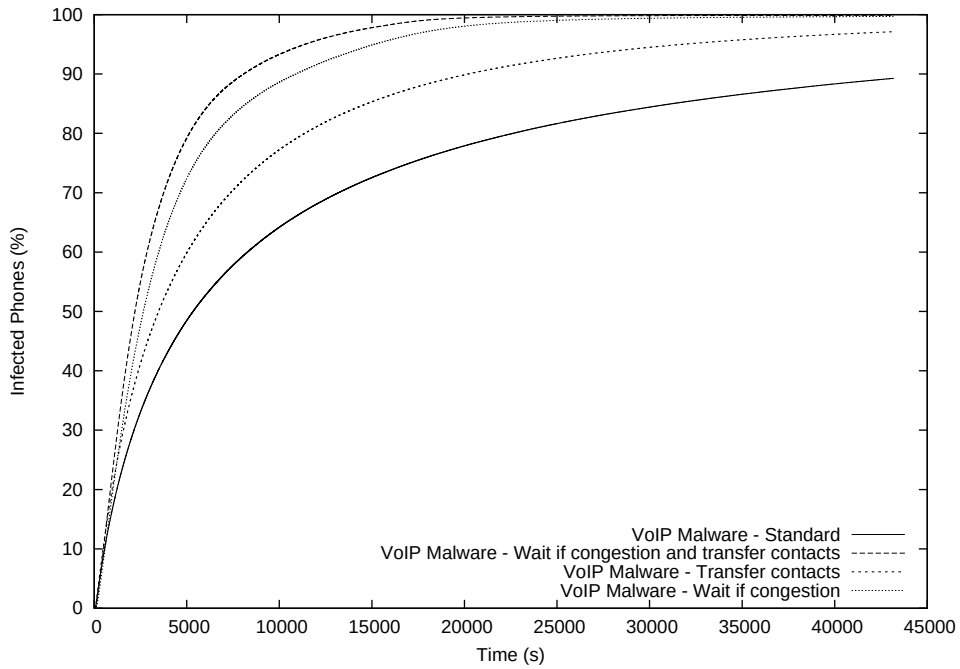

Figure 4.17: Infection rates for VoIP malware that avoids congestion and transfers contacts.

achieve this bound.

However, if the malware creator uses an out-of-band channel to communicate information, it would be possible to approach the maximum rate. One such channel that will likely exist in the future is access to the Internet. Since the number of messages is constrained, the overall network bandwidth utilization is low. Thus, if phones uploaded their address books to a centralized server on the Internet, as well as informing the server of its newly infected state, a nearly complete global address book could be created. Then, each time a phone was capable of sending a message, it could query a server for a non-infected phone and be assured that its effort would not be wasted. Figure 4.18 shows the potential that such a worm

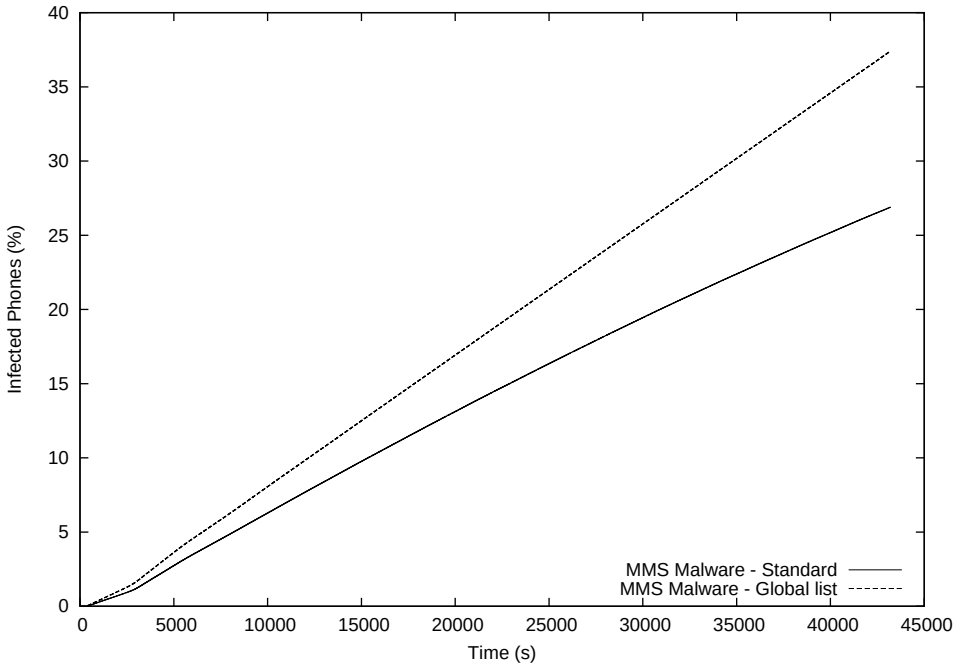

Figure 4.18: Infection rates for MMS malware that uses a centralized list to coordinate infections.

could have on the network compared to the standard MMS malware. Although it does not approach the speeds at which VoIP malware spreads, it reaches almost half the population after only twelve hours. It reaches the 25% infection level 1.3 times faster than the standard MMS malware. The average rate of infection when using the global list is 48.53 new infections per second (compared to 35.07 in the standard MMS scenario). Given the 100 messages per second capacity of the server in our simulation, and the fact that senders and receivers share this limit, it is nearly optimal. Indeed, in our calculations in Section 4.3, we showed that at 43,200s the highest infection level possible was 38.4% of the population. In this scenario, the malware infects 37.4% of the population at the same point in time.

4.5 The operator's dilemma - Deploying defenses

Once an infection begins spreading, network operators need to act with diligence to contain the worm from reaching more devices. At the same time, they would still like to be able to offer service to phones that are not infected since revenue directly depends on service. As we have demonstrated, though, there is significant congestion present once malware begins to spread. Effective quarantining not only needs to stop the ability to spread, but also reduce network usage so that service is available to customers. We examine a few possibilities that network operators could implement in an effort to contain malware.

4.5.1 Filters

Malware which spreads through MMS messages must transit through the centralized MMS server, which acts not only as a bottleneck, but also a convenient location for stopping the spread. Each message is also self-contained, meaning either the entire payload of the malware, or some identifying characteristic, must be present. This situation suggests the ability to filter out offending messages before they reach cell phone users. In Figure 4.19 we look at the response of a network operator that identifies the malware 20 minutes after infection starts and installs a filter on all incoming messages. Predictably, the spread of the virus is cut off quickly, and the rest of the population remains uninfected.

The centralized location of the MMS server, which allows for a single place to stop an attack, does not currently have an equivalent in the VoIP scenario.

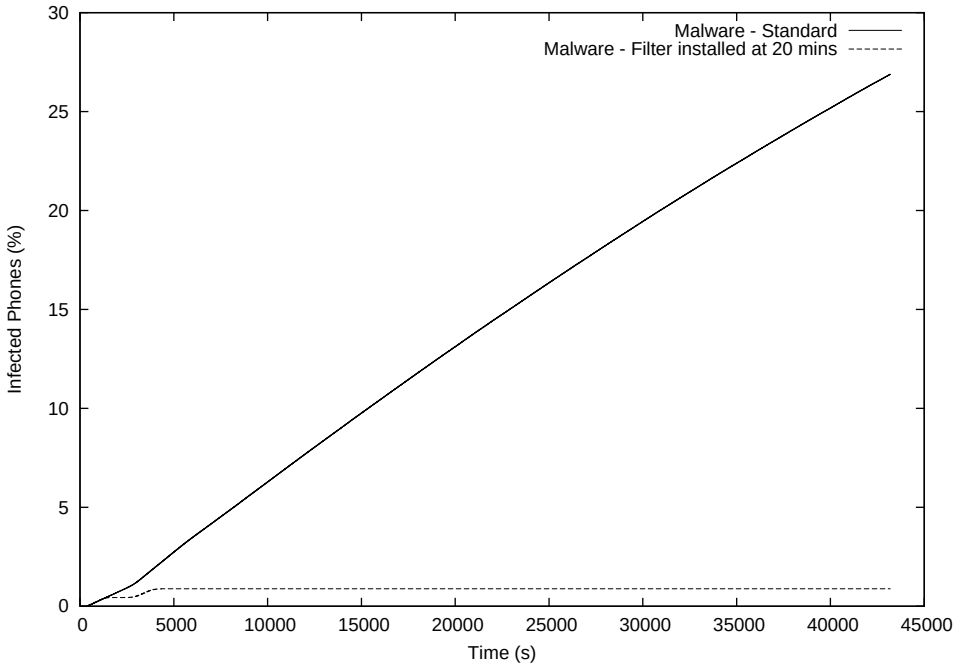

Figure 4.19: Infection rate when MMS messages are filtered after starting at 20 minutes after infection starts.

SGSN and GGSN nodes are capable of performing deep packet inspection for the purposing of billing and basic filtering. However, VoIP data may be encrypted depending on the implementation, and the streaming nature of VoIP traffic makes it difficult to apply content filters rapidly. Clearly, adding these capabilities to the infrastructure would be beneficial to network operators in the case of an outbreak that spread through an unforeseen vector, such as VoIP. If the operator had such an ability, the infection could similarly be stopped in its tracks as demonstrated with the MMS case. However, assuming that this ability may not be feasible, we examine other methods that operators could deploy.

4.5.2 Rate limiting

As proposed for Internet malware [64] and even implemented in operating systems [31], mobile phone network operators could rate limit connection attempts from phones. With rate limiting, we assume that the network operator becomes aware of the outbreak at some time and then institutes a conservative policy where a phone may only make one call every 10 or every 30 minutes. This policy would hopefully have the effect of slowing the malware down while perhaps giving more time to determine a way to disinfect phones. Figure 4.20 shows, however, that even these drastic rate limiting schemes still do not stop the malware. Rather, they can have the alternative effect of reducing congestion, allowing malware to spread with greater ease. For example, if the network operator rate limits phones to one call every ten minutes, starting ten minutes after the start of the infection, it is too late to effectively block the spread — the infection curve follows nearly the same curve as the standard infection. However, if the operator starts rate limiting at five minutes, it appears to be just early enough to catch the spread before it reaches too many people, and does slow it down. However, none of these policies stop the spread completely. They only move the early stages of infection to a later point.

4.5.3 Blacklisting users

Rather than rate limiting all users, operators may wish to disconnect and deny service to users they suspect have been infected. This technique could be accom-

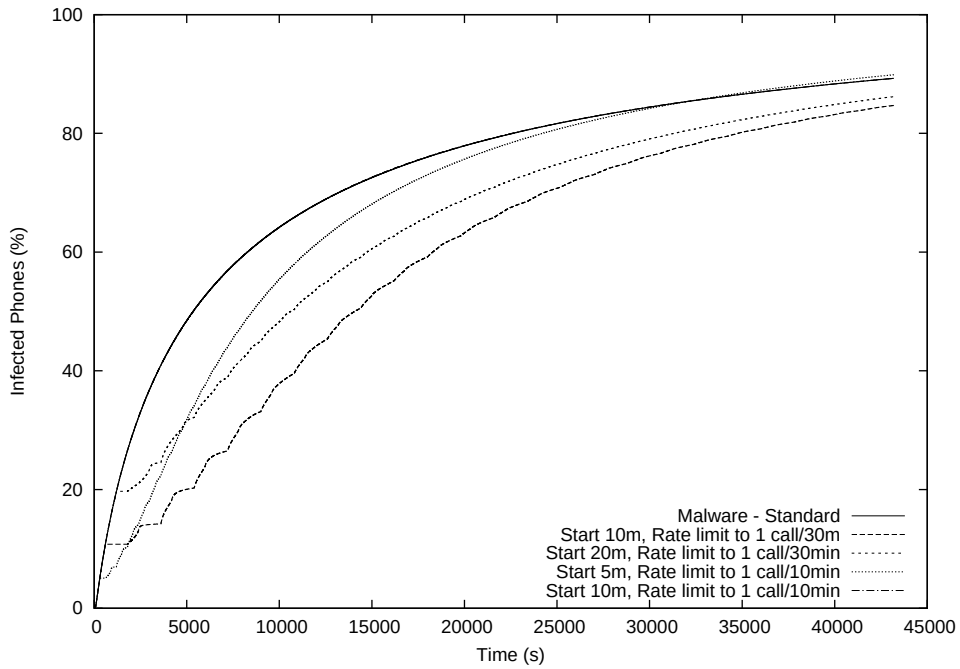

Figure 4.20: Rate limiting users to slow down the malware spread. Note that the "start 10m, rate limit to 1 call/10min" curve is occluded by the "standard" malware

plished through standard methods which turn off service for non-paying customers. The practice would stop phones from consuming bandwidth and infecting other devices. However, they would still most likely contend for radio resources at the radio cell level. We examine a few simple heuristics for determining when a phone has become infected from the network operator's viewpoint. Figure 4.21 shows the effect of blacklisting cell phone users who try to make three calls in one minute and users who try to make two calls in one minute. This policy is implemented ten minutes after the start of the outbreak. The policy is, of course, only one simple heuristic that could be deployed. Network operators likely have more advanced profiles of what constitutes suspicious activity. However, even with an aggressive

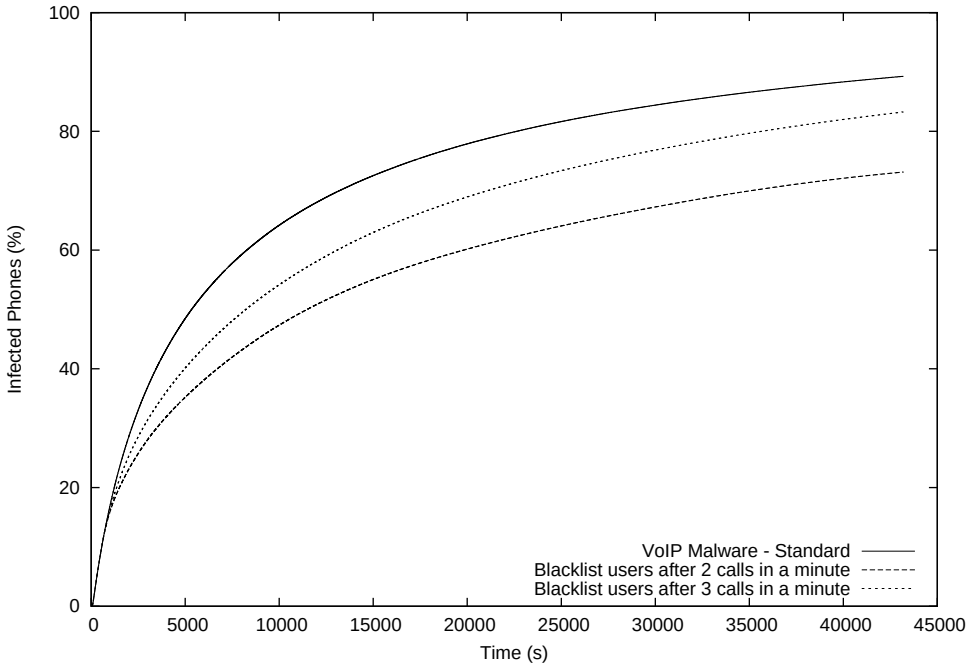

Figure 4.21: Blacklisting users exhibiting unusual behavior.

policy of blacklisting users, a large number of devices become infected, albeit more slowly. Blacklisting users, even infected ones, still ends up being a game of catch up. Moreover, removing users from the network reduces congestion and allows non-blacklisted phones to continue to spread. Ultimately, it would seem that any effort that cannot stop nearly all users at an early stage is doomed to fail, unless even more drastic measures are taken [10, 37].

4.6 Calling all janitors - Malware cleanup

After an outbreak, a large number of cell phones could be infected and unable to connect to the network, perhaps because they have been blacklisted. An im-

portant topic for the future of the network operator's business is how to disinfect phones. One option could be to have mechanisms to deploy firmware updates. This approach, however, may require that phones authenticate and connect to the network. If the phone is still infected, it may consume valuable radio resources at the least. Instead, it may be more beneficial to have users visit a local outlet to update software, which of course presents another logistic challenge. A network dealing with an outbreak of a worm would also face a public relations disaster, especially if it was the first instance of such an event. The resulting loss of customers to other networks seen as providing more secure technology could deal a serious blow to a business model, which could of course have been the goal of the malware in the first place.

5

Conclusions

This dissertation investigates malware propagation in a mobile phone network using communication services. We developed a discrete-event simulator capable of modeling the characteristics of a worm as it spreads across the topology of a single network provider. To ensure the accuracy and realism of our experiments, we created a Radio Access and Core Operator Network (RACoON) topology generator that generates a course-grained hierarchy of a single network provider's system. To model communication patterns, we studied a number of social network topologies and provided initial evidence of realistic contact degree distributions for actual cell phone users. Our simulations focused on two possible attack vectors — MMS messages and a VoIP software exploit — and found that simple malware propagation is severely limited in both instances by bandwidth and capacity constraints. However, more sophisticated malware can spread more quickly, for example, by alleviating congestion or distributing address books between phones. We also showed

that stopping the spread of virulent malware is difficult, even if acted upon quickly. Even aggressive rate limiting and blacklisting techniques cannot contain malware spreading through the system within just a few minutes of propagation.

It was our goal not only to study and understand the characteristics of worm propagation in the relatively proprietary cell phone domain, but also to raise awareness. We hope that the tools we have created will spur others to also examine the effects of malware and the defenses necessary to contain an attack. As mobile phones become more powerful — and more attractive to attackers — the topic of mobile malware will eventually begin to grab headlines. We believe it is important to jump-start the discussion before the cell phone domain undergoes what the Internet has experienced for the past decade.

Appendix A

Population density file format

The XML file that represents population count and distribution has the following format:

```
<grid name=#STRING# sidesize=#INT# sideunits=#STRING#>
    <grid_cell x=#INT# y=#INT# longitude=#FLOAT latitude=#FLOAT#
        pop=#INT# />
</grid>
```

The generated file contains a *grid* which contains many *grid_cell*s. Each *grid_cell* is a square region that has a length of *sidesize*, measured in the units specified by *sideunits*. For our study, each *grid_cell* was 1x1 square miles. The *grid_cell* element defines more data for a region, including the location in the rectangular grid as x and y values. It also includes the population for that *grid_cell* and actual longitude and latitude coordinates for the region.

Appendix B

Radio Access and Core Operator Network (RACoON) topology generator file format

The XML file generated by RACoON has the following format:

```
<scenario>
     <umts_topology>
          <nodeb id=#INT# latency=#FLOAT# pop=#INT#>
               <radio_cell user_bw=#FLOAT# total_bw=#FLOAT#
                   pop=#INT#>
                    <grid_cell x=#INT# y=#INT# />
               </radio_cell>
          </nodeb>

          <rnc id=#INT# pop=#INT#/>

          <sgsn id=#INT# pop=#INT#/>

          <mmc id=#INT# max_mps=#INT#/>

          <link bw=#FLOAT# latency=#FLOAT#>
               <attach id=#INT#/>
               <attach id=#INT#/>
          </link>
     </umts_topology>
```

```
    <routes>
        <sgsn id=#INT#>
            <dest id=#INT#>
                <hop id=#INT#/>
            </dest>
        </sgsn>
    </routes>

    <grid size=#INT# cell_side=#INT#>
        <grid_cell x=#INT# y=#INT# pop=#INT#/>
    </grid>

</scenario>
```

A scenario may consist of a *umts_topology* or/and a *gsm_topology*. If both are included they would typically be connected. However, in our study we only generate a *umts_topology*. The Node-Bs (*nodeb*) have an ID attribute, attributes for location (in grid coordinates), and air-interface latency. The Node-B has one or more radio cells that each cover a set of grid cells. The radio cell has attributes that define the maximum packet data rate per user (*user_bw*) and for all users in the cell (*total_bw*). The *umts_topology* also includes RNC (*rnc*) and SGSN (*sgsn*) nodes. Nodes are connected by links that have bandwidth (*bw*) and latency attributes. Bandwidths are given in bps and latencies in seconds.

In addition to the topology information, the scenario includes information regarding shortest path routes between SGSNs. It also specifies the grid, which is divided into grid cells. The *routes* element defines how to route traffic from a source SGSN to a destination SGSN using the intermediate *hop* elements.

The entire grid, and each cell, is quadratic, where the length of a cell side is defined by the *cell_side* attribute (in meters). Each *grid_cell* specifies an average

population. Grid cells can be omitted from the grid block, meaning that the average population in those grid cells is negligible.

Bibliography

[1] R. Albert and A.-L. Barabási. Topology of evolving networks: Local events and universality. *Physical Review Letters*, 85(24), Dec 2000.

[2] D. Alderson, L. Li, W. Willinger, and J. C. Doyle. Understanding Internet topology: principles, models, and validation. *IEEE/ACM Transactions on Networking*, 13(6), Dec. 2005.

[3] P. Barford and V. Yegneswaran. An inside look at botnets. In *Special Workshop on Malware Detection, Advances in Information Security*, June 2006.

[4] M. Bialoglowly. Bluetooth Security Review, Part 1, Apr. 2005. `http://www.securityfocus.com/infocus/1830`.

[5] L. Billings, W. Spears, and I. Schwartz. A unified prediction of computer virus spread in connected networks. *Physics Review Letters*, May 2002.

[6] G. L. Bodic. *Mobile Messaging: Technologies and Services, SMS, EMS and MMS*. Wiley and Sons, 2nd edition, 2005.

[7] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, June 2002.

[8] K. Calvert, M. Doar, and E. Zegura. Modeling Internet topology. *IEEE Transactions on Communications*, Dec. 1997.

[9] T. Chen. Trends in viruses and worms. *Internet Protocol Journal*, Sept. 2003.

[10] T. Chen and N. Jamil. Effectiveness of quarantine in worm epidemics. In *IEEE International Conference on Communications (ICC'06)*, June 2006.

[11] Z. Chen and C. Ji. A self-learning worm using importance scanning. In *Proceedings of the 2005 ACM Workshop on Rapid Malcode (WORM'05)*, Nov. 2005.

[12] D. Dagon, T. Martin, and T. Starner. Mobile phones as computing devices: The viruses are coming! *IEEE Pervasive Computing*, 03(4), Oct. 2004.

[13] R. Dotson. Speech to investors. *Financial Times*, Oct. 2006. `http://www.ft.com/cms/s/8fb970f6-556d-11db-acba-0000779e2340.html`.

[14] N. Eagle and A. Pentland. Sensing complex social systems. In *Journal of Personal and Ubiquitous Computing*, June 2005.

[15] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the AMC SIGMOD International Conference on Management of Data*, June 1984.

[16] C. Hager and S. Midkiff. An analysis of Bluetooth security vulnerabilities. In *Wireless Communications and Networking, 2003 (WCNC'03)*, Mar. 2003.

[17] C.-Y. Huang, C.-T. Sun, and H.-C. Lin. Influence of local information on social simulations in small-world network models. *Journal of Artificial Societies and Social Simulation*, 8(4), 2005.

[18] M. Hypponen. Malware goes mobile. *Scientific American*, 295(5), Nov. 2006.

[19] G. Keizer. Spam volume jumps 35% in November, Dec. 2006. `http://informationweek.com/news/showArticle.jhtml;?articleID=196701527`.

[20] J. O. Kephart and S. R. White. Directed-graph epidemiological models of computer viruses. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991.

[21] V. Kostakos. Experiences with urban deployment of Bluetooth (given at UCSD), Mar. 2007. `http://www.cs.bath.ac.uk/~vk/files/pres_ucsd.pdf`.

[22] M. Landler and J. Markoff. Digital fears emerge after data siege in Estonia. *The New York Times*, May 2007.

[23] N. Leavitt. Mobile phones: The next frontier for hackers? *Computer*, 38(4), Apr. 2005.

[24] D. Liben-Nowell. *An Algorithmic Approach to Social Networks*. PhD thesis, Massachusetts Institute of Technology, 2005.

[25] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, 102(33), Aug. 2005.

[26] P. Mahadevan, D. Krioukov, M. Fomenkov, B. Huffaker, X. Dimitripoulos, kc claffy, and A. Vahdat. The Internet AS-level topology: Three data sources and one definitive metric. In *ACM SIGCOMM Computer Communications Review (CCR)*, Jan. 2006.

[27] J. Markoff. A disruptive virus invades computers around the world. *The New York Times*, May 2000.

[28] J. Markoff. Attack of the zombie computers is a growing thread, experts say. *The New York Times*, Jan. 2007.

[29] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: An approach to universal topology generation. In *Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01)*, Aug. 2001.

[30] J. W. Mickens and B. D. Noble. Modeling epidemic spreading in mobile environments. In *Proceedings of the 4th ACM Workshop on Wireless security (WiSe'05)*, Nov. 2005.

[31] Microsoft. Changes to functionality in Microsoft Windows XP Service Pack 2. `http://technet.microsoft.com/en-us/library/ba55af8c-de59-2b4f-8128-c6a7%444490d7.aspx`.

[32] M. Mitzenmacher. Dynamic models for file sizes and double pareto distributions, 2002. `http://www.eecs.harvard.edu/~michaelm/postscripts/tempim2.ps`.

[33] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 2004.

[34] Mobile World. The mobile world briefing, Oct. 2006. `http://www.themobileword.com/tmwdev.objects/documents/pdf/TMWBriefingIs%sue40.pdf`.

[35] Mobiledia. LG enV (VX9900) Specifications. `http://www.mobiledia.com/phones/lg/env.html`.

[36] D. Moore, C. Shannon, and J. Brown. Code-Red: A case study on the spread and victims of an Internet worm. In *Proceedings of the Internet Measurement Workshop (IMW'02)*, Nov. 2002.

[37] D. Moore, C. Shannon, G. M. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of the 2003 IEEE Infocom Conference*, Apr. 2003.

[38] M. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences of the United States of America (PNAS'01)*, 98(2), Jan. 2001.

[39] M. Newman, S. Forrest, and J. Balthrop. Email networks and the spread of computer viruses. *Physical Review E*, 66(3), Sept. 2002.

[40] D. Nystedt. U.S. marks new cell phone record in 2005, Apr. 2006. http://www.infoworld.com/article/06/04/07/77227_HNcellphonerecord_1.htm%l.

[41] H. Orman. The Morris worm: A fifteen-year perspective. *Security and Privacy Magazine, IEEE*, Sept. 2003.

[42] J. Paquette. A history of viruses. http://www.securityfocus.com/infocus/1286.

[43] R. Racic, D. Ma, and H. Chen. Exploiting MMS vulnerabilities to stealthily exhaust mobile phone's battery. In *Proceedings of the Second IEEE Communications Society / CreateNet International Conference on Security and Privacy in Communication Network (SecureComm)*, Aug. 2006.

[44] Richard K. Miller and Associates. *The 2007 E-Commerce Market Research Handbook*. Richard K. Miller and Associates, 2006.

[45] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4), Nov. 1984.

[46] G. Serazzi and S. Zanero. Computer virus propagation models. In *Tutorials of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'03)*, Oct. 2003.

[47] S. Staniford, D. Moore, V. Paxson, and N. Weaver. The top speed of flash worms. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM'04)*, Oct. 2004.

[48] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium (Security '02)*, Aug. 2002.

[49] Steve Rosenbush. Smart phones: Intelligence spreads. *BusinessWeek*, May 2005. http://www.businessweek.com/technology/content/may2005/tc2005053_1044.h%tm.

[50] J. Su, K. K. W. Chan, A. G. Miklas, K. Po, A. Akhavan, S. Saroiu, E. de Lara, and A. Goel. A preliminary investigation of worm infections in a Bluetooth environment. In *Proceedings of the 4th ACM Workshop on Recurring Malcode (WORM'06)*, Nov. 2006.

[51] Symbian. Symbian OS V9.2. http://www.symbian.com/developer/techlib/v9.2docs/doc_source/reference/%index.html.

[52] Symbian. Symbian fast facts, Mar. 2007. http://www.symbian.com/about/fastfacts/fastfacts.html.

[53] Telephia. Americans lag behind europeans in smartphone adoption, Dec. 2006. http://telephia.com/html/Smartphonepress_release_template.html.

[54] S. Toyssy and M. Helenius. About malicious software in smartphones. *Journal of Computer Virology*, Aug. 2006.

[55] P. Traynor, W. Enck, P. McDaniel, and T. L. Porta. Mitigating attacks on open functionality in SMS-capable cellular networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom'06)*, Sept. 2006.

[56] U.S. Census Bureau. Census 2000 U.S. Gazetteer Files, County Subdivisions. http://www.census.gov/geo/www/gazetteer/places2k.html.

[57] T. Vogt. Simulating and optimising worm propagation algorithms, Sept. 2003. http://www.securityfocus.com/data/library/WormPropagation.pdf.

[58] A. Wagner, T. Ubendorfer, B. Plattner, and R. Hiestand. Experiences with worm propagation simulations. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM'03)*, Oct. 2003.

[59] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, Dec. 1988.

[60] N. Weaver, I. Hamadeh, G. Kesidis, and V. Paxson. Preliminary results using scale-down to explore worm dynamics. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM'04)*, Oct. 2004.

[61] Wikipedia. Caribe (computer worm) — Wikipedia, The Free Encyclopedia, 2007. [Online; accessed 13-April-2007].

[62] Wikipedia. Commwarrior-A — Wikipedia, The Free Encyclopedia, 2007. [Online; accessed 13-April-2007].

[63] Wikipedia. Erlang distribution — Wikipedia, The Free Encyclopedia, 2007. [Online; accessed 7-April-2007].

[64] C. Wong, S. Bielski, A. Studer, and C. Wang. Empirical analysis of rate limiting mechanisms. In *8th International Symposium on Recent Advances in Intrusion Detection (RAID'05)*, Sept. 2005.

[65] J. Xiong. Act: Attachment chain tracing scheme for email virus detection and control. In *Proceedings of the ACM Workshop on Rapid Malcode (WORM'03)*, Oct. 2004.

[66] T. Zeller. Black market in stolen credit card data thrives on Internet. *The New York Times*, June 2005.

[67] H. Zheng, D. Li, and Z. Gao. An epidemic model of mobile phone virus. In *1st International Symposium on Pervasive Computing and Applications Proceedings (SPCA'06)*, Jan. 2006.

[68] C. Zou, D. Towsley, and W. Gong. Email worm modeling and defense. In *Proceedings of the 13th International Conference on Computer Communications and Networks*, Oct. 2004.

[69] C. Zou, D. Towsley, W. Gong, and S. Cai. Routing worm: A fast, selective attack worm based on ip address information. In *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation (PADS'05)*, June 2005.