# High performance, scalable, and expressive modeling environment to study mobile malware in large dynamic networks

**Karthik Channakeshava**

Dissertation submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Michael S. Hsiao, Chair
Madhav V. Marathe, Co-Chair
Y. Thomas Hou
Jung-Min Park
Jeffrey H. Reed
Anil Kumar Vullikanti

October 14, 2010
Blacksburg, Virginia

Keywords: mobile, Bluetooth, malware, wireless epidemiology, parallel discrete event simulation, interventions

# High performance, scalable, and expressive modeling environment to study mobile malware in large dynamic networks

Karthik Channakeshava

## Abstract

Advances in computing and communication technologies are blurring the distinction between today's PCs and mobile phones. With expected smart phones sales to skyrocket, lack of awareness regarding securing them, and access to personal and proprietary information, has resulted in the recent surge of mobile malware. In addition to using traditional social-engineering techniques such as email and file-sharing, malware unique to Bluetooth, Short Messaging Service (SMS) and Multimedia Messaging Service (MMS) messages are being used. Large scale simulations of malware on wireless networks have becomes important and studying them under realistic device deployments is important to obtain deep insights into their dynamics and devise ways to control them.

In this dissertation, we present *EpiNet*: an individual-based scalable high-performance oriented modeling environment for simulating the spread of mobile malware over large, dynamic networks. EpiNet can be used to undertake comprehensive studies during both planning and response phase of a malware epidemic in present and future generation wireless networks. Scalability is an important design consideration and the current EpiNet implementation can scale to 3-5 million device networks and case studies show that large factorial designs on million device networks can be executed within a day on 100 node clusters. Beyond compute time, EpiNet has been designed for analysts to easily represent a range of interventions and evaluating their efficacy.

The results indicate that Bluetooth malware with very low initial infection size will not result in a major wireless epidemic. The dynamics are dependent on the network structure and, activity-based mobility models or their variations can yield realistic spread dynamics. Early detection of the malware is extremely important in controlling the spread. Non-adaptive response strategies

using static graph measures such as degree and betweenness are not effective. Device-based detection mechanisms provide a much better means to control the spread and only effective when detection occurs early on. Automatic signature generation can help in detecting newer strains of the malware and signature distributions through a central server results in better control of the spread. Centralized dissemination of patches are required to reach a large proportion of devices to be effective in slowing the spread. Non-adaptive dynamic graph measures such as *vulnerability* are found to be more effective.

Our studies of SMS and hybrid malware show that SMS-only malware spread slightly faster than Bluetooth-only malware and do not spread to all devices. Hybrid malware spread orders of magnitude faster than either SMS-only or Bluetooth-only malware and can cause significant damage. Bluetooth-only malware spread faster than SMS-only malware in cases where density of devices in the proximity of an infected device is higher. Hybrid malware can be much more damaging than Bluetooth-only or SMS-only malware and we need mechanisms that can prevent such an outbreak. EpiNet provide a means to propose, implement and evaluate the response mechanisms in realistic and safe settings.

*To amma and anna*

*for the countless sacrifices they have made to get me where I am today*

# Acknowledgements

Throughout my graduate study many people have helped me. I would like to express my appreciation to all those who helped me.

First and foremost, I would like to thank three people who most influenced me as a researcher: Dr. Michael Hsiao, Dr. Madhav Marathe, and Dr. Anil Vullikanti. I am grateful to Dr. Hsiao for accepting me as his student and being patient with me and advising me in the initial years. I am eternally grateful to him for continuing to serve as the chair even when I changed the area of research. I would like to thank Dr. Marathe for graciously agreeing to work with me on the mobile malware study as my PhD topic. I am truly grateful to him and Dr. Vullikanti for mentoring me during these years, improving my research and writing skills. I am grateful to both of them for patiently guiding me through the research and providing feedback to me on a regular basis. I am grateful to all three of them for always being ready to help and guide me. I have been greatly influenced by their passion for research, attention to detail, and openness. I have had an enriching and enjoyable experience working with them and hope this interaction will continue for years to come.

I am grateful to Dr. Jeffrey Reed, Dr. Jung-min Park, and Dr. Thomas Hou for serving on my Ph.D. committee, and for their time and co-operation in reviewing this work. Their suggestions and comments helped immensely in improving the quality of the research and this thesis.

I owe a great deal to the faculty, staff and students of Network Dynamics and Simulation Science Lab. They have guided me in my research and made my work environment pleasurable. I am grateful to Dr. Keith Bisset for collaborating with me and discussing with me the details of

EpiSimdemics and helping me through the process of building the EpiNet simulator we present here. I would like to thank Paula Stretz for patiently answering the zillion questions I asked her about the data. I hope my queries on the Oracle database are more efficient now! I am grateful to Dr. Achla Marathe and Dr. Dick Beckman for guiding me on the statistics related questions and collaborating on the spectrum markets project. I would like to thank Sharon Smyth and Joyce Randall for the love and affection they have shown me and made me feel at home. I am grateful to all the students and postdocs at NDSSL for being friendly to me. They include Ashwin Aji, Suruchi Deodhar, Fei Huang, Chris Kuhlman, Yifei Ma, Kalyani Nagaraj, Elaine Nsoesie, Nidhi Parikh, Guanhong Pei, Katherine Wendelsdorf, and Zhao Zhao. Special thanks to Andrea Apolloni and Bryan Lewis for the endless discussions we had on a range of topics during our coffee and lunch breaks. I am grateful to Samarth Swarup and Maleq Khan for their guidance when I had a query connected with my research. Special thanks to Shrirang Yardi and Deepti Chafekar for being there to offer their support, encouragement, and motivation during my doctorate. They have always been there to discuss research ideas and shape them. Shrirang has always helped me improve my writing.

I would like to thank my very good friends Deepti Chafekar, Shrirang Yardi, Bharath Ramesh, Omprakash Seresta, Maheshwar Chandrasekar, Saurabh Bisht, Ashwin Aji, Sapna Kaul, and Suruchi Deodhar for their friendship and making life in Blacksburg interesting and enjoyable. They have been around to listen to my ramblings, to encourage me, and to provide me with the company. Having been in Blacksburg for a long time, I have had the pleasure of meeting working with many bright students and all of them have become close friends over the years. I don't think there is room to mention everyone here, but I am truly grateful to one and all.

I owe a great deal to my family – my mother, father, Bharath Channakeshava, Suma Bharath, and my extended family of friends and well wishers who have been always been there to encourage and support me. I am grateful to have the unconditional love and affection of my family and for their constant support and encouragement during the ups and downs during my doctoral studies.

The results in this dissertation were obtained in collaboration with my advisors, Dr. Madhav

Marathe, Dr. Anil Vullikanti, and Dr. Keith Bisset. My other collaborators are Dr. Achla Marathe, Dr. Samarth Swarup, Dr. Maleq Khan, Deepti Chafekar and Shrirang Yardi, Bryan Lewis, Christopher Kuhlman, Fei Huang, and Guanhong Pei.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

Advances in computing and communication technologies have made mobile communication devices extremely powerful—blurring the distinction between today's PCs and mobile phones. Worldwide sales of mobile phones to end users surpassed 1.15 billion units in 2007, a 16% increase from 2006 sales of 990.9 million [35]. In 2007, 20.9 million units were shipped in North America alone [47]. These devices have become a part of the social fabric and world economy. By the end of 2010, 1.2 billion people will carry handsets capable of rich, mobile commerce providing a rich environment for convergence of mobility and the Web [36]. Most of these devices are capable of communicating over multiple physical layer technologies, interfacing with disparate networks, such as Evolution-Data Optimized (EV-DO), wireless broadband networks at residences, coffee shops and airports, and Bluetooth or IrDA proximity networks. Higher data rates offered by new technologies such as Long Term Evolution (or LTE) and next generation 4G networks are going to further propel the acceptance and use of smart phones. We are witnessing the true beginning of nomadic and ubiquitous computing and communications.

## 1.1 The Problem

The increasing number of smart phones, lack of awareness regarding securing them, and access to personal and proprietary information, have resulted in the recent surge of mobile malware. Viruses, worms, spam and other malicious software target these devices. Traditional social-engineering techniques such as email and file-sharing as well as vectors unique to mobile devices such as Bluetooth, Short Messaging Service (SMS), and Multimedia Messaging Service (MMS) are used by these malware. By March 2008, F-Secure had counted 401 different types of mobile malware in the wild, and McAfee had counted 457 [59]. A recent survey by Credant Technologies [24] shows poor smart phone security attitude among mobile phone users. While 12% of those surveyed stored bank account details, 5% stored credit card information on mobile phones, and 1% even stored their pin and passwords. Mobile devices in the corporate environment are mostly employee owned, un-managed, and largely unprotected against mobile threats. These devices store or have access to sensitive corporate information ranging from customer records to the company's intellectual property. Over 49% of the companies surveyed in Phifer [80] reported significant use of mobile devices to transmit business-critical data. Although current generation, proof-of-concept malware such as Cabir [29] and CommWarrior [57] have not resulted in harmful outcomes, the same may not apply to next generation malware. A greater threat are "crossover" malware that spread from desktop PCs to mobiles and vice versa. Crossover [79], a recent virus, spreads from a Windows desktop to a mobile running on Windows Mobile Pocket PC® through the ActiveSync® synchronization utility. Cardtrp.A [90] is another such crossover malware that affects the device's memory card and installs Windows worms (Win32.Rays, Win32.Padobot.Z and Win32Cydog.B) when inserted into the PC. A more recent threat are large cellular botnets that can perpetrate a denial-of-service (DoS) attack against cellular network core, web-sites, or the Internet in general [37].

## 1.2 Motivation

The ability of smart phones to form proximity networks in an ad hoc manner and the exposure to diverse networks make the study of such malware challenging. Detecting and responding to such dynamic malware is extremely difficult. Existing solutions to target malware in PCs cannot be directly employed with mobile phones: (*i*) they consume a significant amount resources to function, and (*ii*) require constant updates to function well. So, we need a new mechanism to handle threats that affect mobile devices. A pre-requisite to any such mechanism is to study the problem in detail and understand the underlying characteristics of the spread. Accurate studies into these malware will yield ideas for handling attacks, provide information to detect when an infection is prevalent, predict the impact of a malware, and evaluate the effectiveness of response strategies in controlling the spread.

A number of mathematical and simulation based approaches have been proposed to study epidemiological problems in wireless networks [83]. Analytical models such as the ones presented by Yan and Eidenbenz [98] and Mickens and Noble [68] can be used to predict the worm spread. Simulations such as the ones presented by Su et al. [89], Yan and Eidenbenz [98], and Yan et al. [100] can also be used. Analytical models based on mathematical epidemiology provide a natural way to study large systems and have a number of desirable features, such as closed form expressions for important epidemic quantities, e.g., the total number of infected devices. Nevertheless, these models make a number of crucial assumptions (complete mixing, low resolution malware models) that do not hold in the real world. Detailed simulations that build on well-known network simulators such as NS-2 [76] or Qualnet [85] provide a natural alternative. These simulators allow for a quick and easy implementation and evaluation of the worm, and have been used in a number of recent studies. Researchers can create different network topologies and observe the growth of an infection with these simulators. Yan et al. [100] use random networks with random mobility models and Su et al. [89] use traces from real Bluetooth activity at locations for the study. A recent study by Wang et al. [94] considers real data from a mobile phone network to construct mobility and calling patterns to discover social networks and study the malware propagation. The granularity

of the mobility is at the level of cell towers, and within each tower's area, they consider a uniform distribution of devices and construct the network based on Bluetooth range.

Although these studies provide significant understanding of the spread, they have several important shortcomings. (*i*) Wireless networks formed by smart phones and mobile devices carried by individuals have different structural properties than the networks formed by random waypoint (RWP) movement models considered in the past. The dynamics observed in such networks are different from the networks constructed from the activity-based mobility models (see Section 5.1 for a comparison). (*ii*) The scale of the networks in terms of the number of devices represented in the evaluations (city wide, urban area wireless networks with millions of devices) cannot be handled by the state of the art network simulation tools. The network scale matters as the insights gained by using realistic networks can be easily translated into deployable security policies and an analyst can determine crucial aspects of the malware that affect its spread. Although, some existing literature [94] does consider large scale networks, the resolution of these networks and their accuracy for proximity malware are limited.

So it becomes important to consider a more detailed representation of device interactions. The activity-based mobility model provides this detail and uses real data from census, land use, and activity survey data to construct this network.

## 1.3 Our Contributions

In this dissertation, we study mobile malware and especially present the study of mobile malware that spread through proximity Bluetooth networks and cellular infrastructure – using short messaging service (SMS) or multimedia messaging service (MMS). We propose to model mobile malware using a "network-based-approach" that employs detailed representations of the underlying communication networks and parameterized models of within-host behavior and between-host interaction of the malware. We consider high-resolution proximity networks and realistic communication networks to study Bluetooth and SMS/MMS malware propagation, respectively. The primary contribution of this dissertation is the *creation of a high performance network-based modeling*

*environment to study mobile malware in large, realistic, dynamic networks.* Specific contributions are as follows:

- *A scalable, parallel, and expressive agent-based simulation environment, EpiNet, for studying the spread of mobile malware.* The environment allows us to construct realistic device networks – proximity and communication networks – from activity-based mobility models and statistics on communication patterns for a synthetic population. To the best of our knowledge, this is the only environment with the capability to study networks with over 100k devices.

- *Detailed and abstract model for the Bluetooth malware.* For the purposes of this dissertation, we propose two models: a detailed model that represents the propagation at high resolution, and a more abstract model that provides increased scalability and allows us to extend the scale of the environment to millions of devices. We show that the results from the abstract model are within a 5% error. The environment with the abstract model can now study large experiment designs in a few hours on a reasonably sized, high performance computing infrastructure.

- *Computational improvements are orders of magnitude greater.* The framework is a computationally efficient representation that scales well when larger problems need to be solved. We have achieved 300X improvements over network simulators while maintaining high fidelity network and malware models. This makes the framework practical to conduct evaluations of malware diffusion on large dynamic networks.

- *Dynamics of the Malware Spread.* We use the environment to study the dynamics of the propagation of Bluetooth and SMS/MMS malware. We perform detailed sensitivity analysis and study the interactions among the network and malware model parameters.

- *Study efficacy of response mechanisms.* We study non-adaptive and adaptive response strategies and determine the effectiveness of these strategies. We evaluate some dynamic graph metrics that have the potential to control the spread. We illustrate the use of the environ-

ment through some case studies that showcase the usefulness and the expressiveness of the environment to specify and study sophisticated individualized responses.

The size of networks we consider for the study are city wide networks with millions of devices to explore the real dynamics of the malware spread. With current implementations of EpiNet and the modeling approximations we have studied, we can study networks with 1.2 Million devices, 7.2 million activities for individuals, and 500,000 locations where the activities take place (Miami city synthetic population). For the above network, and simulated duration of a day, EpiNet completes with a runtime of 50 minutes on 20 nodes of an SGI cluster with Intel Xeon 2.83GHz processors and 8GB of RAM.

## 1.4 Summary of Results

In this dissertation, we present *EpiNet*, a new modeling environment to study wireless epidemiology. EpiNet can be used to understand mobile malware diffusion processes over a realistic dynamic network. EpiNet environment is useful to conduct studies on the impact of network structure on the dynamics of the malware spread. It serves as a tool for analysts and network planners to identify influential and/or vulnerable devices and evaluate different control strategies to build secure infrastructures. Scalability is a design goal and we improve the scalability of EpiNet by proposing model and mobility resolution changes that result in orders of magnitude speedup in the runtime. We call this faster scalable implementation EpiNet. EpiNet now scales to highly unstructured, dynamic networks with 3-5 million devices. To the best of our knowledge this is the only modeling environment that scales to device networks of this size. It also implements more sophisticated intervention schemes[1] and can be used in a large experiment design over a reasonable high-performance computing infrastructure. EpiNet represents a highly scalable version of EpiNet we had proposed earlier and we refer to EpiNet when we are discussing the simulation environment.

[1]The name EpiNet comes from these interventions schemes that have been implemented to *cure* the network of malware.

Although we study the spread of Bluetooth worms over mobile networks and use a malware model we develop, the EpiNet environment can be used to study other malware. A new malware model can be created and plugged into the environment. With the abstraction of the malware model and the change in the resolution of mobility data, we obtain significant speedup over other network simulators. EpiNet obtains 300X improvements in runtime with only 0.1% error in the final infection size in comparison to network simulators. Because the simulation environment uses an abstracted Bluetooth worm model, we are able to obtain orders of magnitude speedup in comparison with a detailed simulator like NS-2. For the same setting (as with NS-2), EpiNet achieves the results in 10 minutes without parallelization. With parallelization, EpiNet can be used on much larger networks and can help evaluate entire regions in a particular city. We are working on scaling this to entire cities. Simulations of such large scale, city wide networks help in gaining insight into the spread characteristics, and devise intelligent schemes to prevent a widespread digital epidemic.

We simulate the spread of proximity Bluetooth malware over fairly large, synthetic, yet realistic networks consisting of millions of devices. The synthetic network is derived using a number of innovative modeling techniques, using detailed location and individual data in a US city. We show that the synthetic networks generated from realistic data exhibit features absent-in, and significantly different from, networks generated using RWP movement models. We compare the spatial and temporal dynamics of Bluetooth malware propagation over realistic networks described above and random-waypoint generated models. The results, not surprisingly, show that the dynamics are qualitatively different over these networks. Importantly, those differences highlight the need for early detection for controlling a malware epidemic. Our simplified worm model and the enhancements to EpiNet suggest that detailed protocol level states are not required to answer system level questions that relate to network-level spread. Specifically, we observe that most infections occur during the first hour of contact with infected device, indicating that coming in contact with the device at inception time can have significant impact on the eventual growth of the infection. This can also potentially impact the way these infections are identified.

We also simulate SMS/MMS based malware and hybrid malware that use both proximity based Bluetooth networks and infrastructure based cellular network to replicate on other susceptible

devices. We find that even though the structure of the social contact networks are significantly different from the proximity network, SMS/MMS-based malware alone do not propagate to all susceptible devices. Hybrid malware on the other hand spread extremely fast and have the potential to infect a large number of devices fairly quickly. We find that aggressive malware can infect 50% of the susceptible population within a matter of 2-3 hours.

## 1.5 Organization

The rest of the dissertation is organized in the following manner:

Chapter 2 discusses the existing literature in the area of malware studies for wired and wireless networks. We look at mathematical models and simulation based studies.

Chapter 3 outlines the EpiNet environment we have developed for studying the problem. This provides a high level description of the modules constituting the environment. We discuss ways to create social networks – proximity and communication networks that are realistic. We propose a detailed malware model to describe the within-host behavior of an infected device. We calibrate the model and validate it using simulation studies.

Chapter 4 proposes some approximations to the mobility of devices and malware model with the aim of scaling the simulations to large networks. We call this new improved framework EpiNet as it is useful to study interventions that are compute-communication intensive. We discuss some case studies to showcase the framework and its ability to answer certain questions. We also discuss system-level implementation of EpiNet with hybrid MPI-threads using Intel's TBB. We study the scalability of the resulting tool by performing strong and weak scaling of the simulator and show that the simulator scales well when more processing elements are provided and shows that larger networks can be studied simply by introducing more processing elements.

Chapter 5 provides the results from the simulations. We study the impact of the worm parameters and the network parameters on the spread of the malware. We perform an analysis of variance calculation to determine the impact of certain malware and network parameters. We study the

change in the propagation dynamics when communication networks are used in addition to proximity networks.

Chapter 6 discusses the response mechanisms for handling malware outbreaks. We look at some non-adaptive static (degree and betweenness) and dynamic (vulnerability) graph metrics and study their effectiveness in controlling the spread of proximity malware. We look at device-based detection and response mechanisms and their ability to detect and respond to mobile malware.

# Chapter 2

# Related Work

Ever since Robert T. Morris released a 99 line program on the Internet in 1988 [49], malware have been written and released on the Internet for various reasons. Similarly, the study of how malware propagate has been undertaken to understand the spread and take responsive action to control the impact. Current and predicted explosion in the sale of smart phones and increase in the applications supported on smart devices have shifted the malware in the direction of mobile devices. The malware has indeed gone mobile [42]. Thus, understanding the dynamics of this new threat is important and learning how to avoid the outbreak on mobile devices has gained importance. The vanishing dichotomy between Internet worms and human diseases makes studying the mobile malware problem much more challenging. The use of short range communication technologies such as Bluetooth and the emergence of malware that exploit them has made mobile malware spread like an epidemic. Three important aspects have to be considered for the study of mobile malware:

- *Location of the device.* Due to the proximity nature of Bluetooth networks, tracking the location of a device is important to predict the spatial spread of the malware. In the case of SMS/MMS based malware propagation, the location of the device provides information on the cell tower serving the mobile.

- *Networks the device has access to or creates.* Because next generation malware will likely employ multiple transmission means (for example, e-mail, web pages, social networking ap-

plications, etc.), mobile devices form the interconnect between multiple, diverse, and heterogeneous networks.

- *Usage patterns of mobile phones.* Smartphone usage patterns of individuals go a long way in both identifying the existence of a malware and predicting how a certain malware may spread over the network.

In this chapter, we provide a brief review of malware studies of Internet worms and mobile malware. We provide an overview of the threats that currently exist from mobile malware affecting wireless mobile devices. We present some prior art on studies of their propagation. We then present a summary of literature that proposes mitigation techniques and response strategies to control the spread of mobile malware.

## 2.1 Computer Epidemiology

The exploration of the computer viruses finds parallels in epidemiological studies undertaken to study the spread of human epidemics. The analogy between human pathogens and computer viruses led to the coining of the term "virus" for exploits in the computer domain. Researchers have proposed the use of mathematical epidemiology to study computer system security, and in particular computer viruses. Kephart, White, and Chess of IBM were the first to put this analogy to use in [50, 52, 51]. Kephart and White [50] observe that for computer worms, the localized nature of program sharing is important to consider in the epidemiological approach and incorporate localization in their proposal. In [51] they propose an analytical model for characterizing the infection in terms of birth rate (rate of infections), death rate (rate of curing infections), and infection transmission patterns. From their models the authors concluded that virus propagation rarely reached epidemic proportions. Although their study is sound and empirically proven, the conclusions of their study—as a result of their assumptions and simplifications—are no longer applicable. These studies were conducted at a time when the Internet connected only a few universities and research institutes and malware spread mostly through storage media (for example, floppy disks). Today, the Internet

connects billions of users across the world and malware spread is faster over the Internet. Further, the IBM study concentrated on the global aspects of the infection spread and later researchers have found that though such a study provides useful insights, a set of carefully selected low-level characteristics help in devising strategies for defense mechanisms.

On July, 19, 2001, more than 350,000 computers connected to the Internet were infected by the Code-Red (CRv2) worm in less than 14 hours. The cost of this epidemic and the subsequent strains of Code-Red is estimated to be in excess of $2.6 billion. This prompted a number of studies [71, 70] to understand the causes and reasons for the spread of Code-Red [101, 69] and Internet worms in general. Internet worms have been extensively studied using accurate worm models and network architectures that they propagate on [101]. A pair approximation (or correlation) model was developed by Nikoloski et al. [75] that uses the salient network characteristics—order, size, degree distribution, and transitivity. They conclude that network structure has considerable impact when local propagation occurs. Recently, this focus has shifted towards studying the worms that spread using the wireless infrastructure, either laptops or smart devices.

## 2.2 Mobile malware threats and their study

In the past few years, owing to wide deployment of wireless networks, the interest has turned to worms and malware created to spread on wireless networks. More than 30 kinds of malware—worms, trojan horses, other viruses, and spyware—have been unleashed against the devices [42]. Such worms propagate through devices that communicate using wireless interfaces. Of late, these worms have become more sophisticated, with capability to propagate using multiple communication schemes. One of the earliest of such viruses written for the Palm PDAs, called the PalmOS/Liberty [61] virus, represented a relatively lower threat as it required user intervention to become active. The infection resulted in applications and databases being deleted from the device. More recent malware have targeted the Nokia 60 series cell phones running Symbian OS due to their popularity and advanced features like Bluetooth and SMS/MMS services. The Cabir [29], Mabir [64], Lasco [58], Drever [25], Skulls [88], and Commwarrior [57] are a few of the worms/trojans that affected

this platform. Most of these worms use Bluetooth to infect the neighboring devices and some use SMS/MMS messages to random people from the infected device's address book. Most of the above malware affect a particular device platform and cause limited damage. A particularly nasty form of Cabir spread rapidly through the audience of the 2005 world track and field championship, and stadium operators flashed warnings on the big screen [42]. Drever, a trojan, propagates by disguising as a Symbian OS update and disables SimWorks (a Symbian antivirus software). Redbrowser [81] was the first trojan to affect the J2ME enabled phones. It uses a social engineering technique to trick the user into sending SMS messages while actually sending a flood to a specific number, increasing the user's service charge. InqTana [30] Bluetooth worm targets vulnerable Mac OS X systems and propagates from one device to another. This worm could be easily modified to corrupt the data on the device. SymbOS/Beselo [13] is another worm that attempts to propagate via a media or image file over a Bluetooth network. It can send an MMS to each number in the phone book about every 2 minutes.

Current generation smart phone applications are usually downloaded from the market places that vendors have set up for them. The belief of a majority of users is that buying software from the Andriod Marketplace or the iTunes store implies that the applications are safe. Issues with iPhone applications stealing location information [12] and malware mobile banking applications stealing banking credentials of users (using a phishing technique) [34] are clear indications to the contrary. Some application vulnerabilities in iPhone [20] also reinforce the point that mobile malware are a sure thing and it is just a matter of time before they gain the same importance as PC malware.

### 2.2.1 Simulation and Emulation Infrastructures

In addition to the network simulators [76, 41, 2] that can be augmented to study malware, custom simulators and emulators to study wired and wireless malware have been proposed. The cyber-DEfense Technology Experimental Research (DETER) is one such experimental infrastructure to conduct Cyber-defense research on Internet-scale [11]. The PArallel Worm Simulator (PAWS) [95] and the WE emulator for analysis of worm spread and defense strategies in local area networks for

the DETER [11] testbed is proposed in [96]. These approaches are for wired infrastructure networks. l Mobile Agent Malware Simulator (or MAlSim) is a distributed, mobile agent framework developed in [60] to simulate attacks against information systems. The MalSim toolkit provides standard patterns that can be used to emulates a malware, study the spread, and control mechanisms. MAlSim specifically addresses the aspects of critical infrastructure security. A queue based model is proposed in [67] to emulate wireless connectivity to model mobile environments for malware.

Reproducing effects of large-scale Internet worms attacks in a laboratory setup realistically is important for detection and defense systems. Liljenstam et al. propose a simulation model in [62] to model large-scale dynamics of worms and its effects on networks. They map realistic IP address spaces and model detailed network behavior and show how the model is used to generate a working prototype worm detection and tracking system.

### 2.2.2 Studying mobile malware spread

Attempts to study and understand the spread of mobile malware have been undertaken. From a network theory perspective, the contact network or device network formed by wireless networks is different from random networks. Further, the propagation characteristics are complicated by the means by which these malware replicate, and the ad hoc nature of the network makes detecting and responding to such malware challenging. Table 2.1 outlines the broad scope of the related work in this area. We attempt to describe each of them and differentiate our proposed work from them.

Mickens and Noble [68] construct an analytical model based on analytical or empirical calculations for the average travel time between two destinations $\langle t \rangle$ and the connectivity distribution $P(k)$ for the network ($k$ represents the degree of a node). The connectivity distribution provides the percentage of nodes having $k$ neighbors. In other words, $P(k) \times N$ represents the percentage of nodes with a certain connectivity level. Under the assumption that the connectivity changes occur at larger instants than the simulation $\Delta t$, they represent the mobile network as a set of $M$ queues, where $Q_m$ contains $P(k_m) \times N$ nodes and every node spends $P(k) \times \langle t \rangle$ time units in each queue before exiting. They assume a susceptible-infected-susceptible (or SIS) model for the disease. $\beta$

Table 2.1: Overview of related work in the study of mobile malware.

| | |
|---|---|
| Mathematical Models | Mickens and Noble [68], Yan and Eidenbenz [99], Nekovee [73], Nikoloski et al. [75], [83] |
| Simulation Studies | Yan and Eidenbenz [98], Yan et al. [100], Zyba et al. [103], Su et al. [89], Fleizach et al. [31] |
| Realistic Studies | Wang et al. [94] |

represents the probability that an infected infects a susceptible and $\delta$ is the probability that an infected device is cured. A homogeneous infection dynamic is simulated in each queue such that $\frac{dI_m}{dt} = \beta k I_m(1 - I_m) - \delta I_m$. The authors make an assumption regarding the steady states in the mobility models that it is possible to derive steady state values for the average travel time $\langle t \rangle$ and $P(k)$. Further, they assume that the interaction in each queue occurs among homogeneous devices (i.e., devices having same degree) and is not true for scale-free networks and other networks that occur in real scenarios.

Yan and Eidenbenz [99] propose an analytical model for the Bluetooth protocol and the worm characteristics. The authors explore and propose an extremely detailed and accurate analytical model tracing the Bluetooth protocol's steps in establishing a connection with a susceptible device. In this model, they also consider the effect of other neighboring devices that are infected and their impact on completion of a successful inquiry process.

Rhodes and Nekovee [83] investigate the effect of human mobility patterns on the spread of wireless epidemics. The authors introduce an individual-based model for mobile devices and evaluate the effect of population characteristics and device behavior on the outbreak dynamics. Using a straight line movement pattern for devices, the authors illustrate that mass-action epidemic models remain applicable in low density regimes, and build an expression for contact rate based on straight line motion of the device. When devices come in range of each other, the infection is propagated from

an infected to a susceptible device with probability $p$. The devices' motion trajectory, and their relative velocity determines the exposure duration.

A combination of large-scale simulations and mathematical modeling to explore the spread of wireless epidemics in fixed ad hoc networks is presented in [74]. The authors conclude that the spreading of worms in these networks is greatly affected by a combination of spatial correlations arising from network topology and temporal correlations resulting from the interference limited nature of communications in these networks. Standard mean-field and network mean-field models from mathematical biology, which are widely used to model worm epidemics in computer networks, are inadequate for describing worm epidemics in wireless ad hoc networks, but spatial epidemic models provide a promising alternative.

Yan et al. [100] study the effects of the mobility patterns on the speeds of the spread. They consider four mobility models—random waypoint, random walk, random direction and random landmark for this study. The simulation studies are conducted on a network of 200 devices in an $75 \times 150\ m^2$ area and duration of 3600 seconds. For the landmarks they use scaled version of the Rice University campus landmarks. The studies underline the fact that mobility plays an important role in the spreading patterns and use random mobility to make this conclusion.

In [98] the authors conduct simulation studies for 200 Bluetooth devices for differing areas from $75\ m^2$ to $150\ m^2$ and observe the effect of speed and density on the spread of the malware. Most of the simulation studies have been conducted for small networks created in small locations. Such small scale studies do not provide actionable information regarding the spreading characteristics and the global impact the malware has on the system as a whole. Further, response mechanisms cannot be studied under such restricted setting and extrapolated to real networks. Furthermore, the nature of some of the mobility models themselves impact the interaction of devices and the networks formed, thus, coloring the resulting spread. For example, random waypoint mobility models are known to create a higher density of devices in the center of the region considered [15, 16], thus creating dense networks under stable conditions.

Fleizach et al. [31] investigate the effect of malware propagation over the wireless backbone net-

works. They characterize the speed and severity of the malware under realistic scenarios and explore network-based defenses against such malware. They develop an event-based simulator to study malware propagating using MMS and Voice-over-IP (VoIP) services. For this they construct a hierarchial Universal Mobile Telecommunications System (UMTS) network and generate the traffic from the mobile nodes. Because the malware propagated by contacting individuals in the phone books of the infected device, they generate various social networks to study the spread under unconstrained and constrained network characteristics.

One of the recent simulation studies was conducted by Zyba et al. [103]. The authors use a combination of traces and synthetic data to model human mobility. They use the Levy walk [82] synthetic mobility model as it is more realistic than other mobility models. In this paper, the authors focus on defense mechanisms deployed against proximity malware and explore three strategies for detecting and mitigating the threat. Local detection is used by the devices to detect the malware. Proximity and global signature dissemination is used to control the malware.

Recent work by Wang et al. [94] is one of the most realistic studies conducted for the propagation of Bluetooth malware both using proximity Bluetooth connections and MMS service. The susceptible-infected (or, SI) malware model is considered in this paper and the authors only study the propagation of the malware. Although they consider a rather simplistic malware model, the data used for estimating human mobility is realistic. They use data from a mobile phone carrier regarding voice calls and messages of approximately 6.2 million customers collected for a month. Using the communication patterns and the associated cell tower information, they identify the position of the individual. Using the data, they generate mobility patterns at the resolution of cell towers using the patterns observed for the individual. The same mobile network data is used to obtain social network for individuals to be used in the study of malware propagation through MMS service. To simulate the infection, they use a compartmental model within each cell tower's area, dividing the population into disjoint compartments whose size changes with time (as the infection spreads). The disease consists of two states—susceptible and infected. For the propagation based on MMS, the authors simply use the social network created from the data and infect a randomly chosen device (from the individual's social network) with some probability.

Table 2.2: Overview of existing literature in the study of mitigation schemes for mobile malware.

| | |
|---|---|
| Detection | Forrest et al. [33], Ellis et al. [27], Bose et al. [19], Kim et al. [54], Jakobsson and Johansson [45], Jakobsson and Juels [46] |
| Mitigation | Zyba et al. [103], Mickens and Noble [67] |

### 2.2.3 Defenses against mobile malware

The nature of the networks created by mobile devices raises serious concerns on the application of Internet based approaches to mitigating mobile malware. Several studies exist in securing the PCs from malware that can be extended to apply for such malware. The best approach to respond to mobile malware attacks can begin at the source—the device itself. Dynamic quarantine [102], comparing signatures, monitoring system calls [33] or power consumption [54], and system behavior [27, 19] are all effective techniques to detect the malware at the device. System level detection can also be achieved at the service providers' side by deep packet inspection and signature comparison techniques.

By monitoring the change in the patterns of system calls executed by a program, perturbations caused by attacks can be identified. A *synthetic* normal profile is created by exercising the program in all anticipated normal modes, or *real* profiles can be generated by normal online usage of the production system. Static analysis techniques (refer to Forrest et al. [33] and references therein) have also been used to generate the normal functioning of the system as the synthetic or real profiles are not complete. There are several ways these can still be circumvented, and techniques have been proposed to deter them. The actual techniques are outside the scope of this current work; we highlight that one can devise ways to identify whether a device is infected. Assuming that such a system is available on a mobile phone, one can detect the infection in the device. In this proposal, we use such methods to make the device aware of its own infection and stop the malicious behavior

by turning off the erring application or system feature.

Defending against malware in mobile phones has been extensively studied. Broadly, detection and response mechanisms can be divided into system level and device level. System level approaches are applicable to detect and respond to malware that spread through the cellular infrastructure, such as short messaging service (SMS) or the multimedia messaging service (MMS). Work in [18] explored rate-limits and SMS quarantine approaches after identifying infected devices. Similarly, the propagation of malware through MMS messages is studied in [84]. The authors model different MMS malware characteristics and study the effectiveness of different response mechanisms, such as virus scan and detection algorithms on all MMS messages, immunization patches, user education, monitoring anomalous behavior, and blacklisting of infected devices.

An alternative to the signature-based approach, behavioral detection [27], has emerged as a promising way of preventing the intrusion of spyware, viruses, and worms. In this approach, the runtime behavior of an application (e.g., file accesses, API calls) is monitored and compared against malicious and/or normal behavior profiles. The malicious behavior profiles can be specified as global rules that apply to all applications, as well as fine-grained application-specific rules. Behavioral detection is more resilient to polymorphic worms and code obfuscation because it assesses the effects of an application rather than specific payload signatures. For example, because encryption/decryption does not alter the application behavior, multiple malware variants generated via run-time packers can be detected with a single behavior specification. As a result, a typical database of behavior profiles should be much smaller than that needed for storing specific payload signatures for each variant of many different classes of malware. This makes behavioral detection particularly suitable for resource-limited handsets. Moreover, behavioral detection has potential for detecting new malware and zero-day worms, because new malware are often constructed by adding new behaviors to existing malware or replacing the obsolete modules with fresh ones, indicating that they share similar behavior patterns with existing malware. [19] proposes a behavioral framework to detect mobile worms, viruses, and Trojans. The authors propose an efficient representation of malware behaviors based on observations and application actions. The database generated is used to study several distinct families of mobile viruses and worms targeting Symbian OS and their variants.

In addition, the authors also propose a two-stage mapping technique that constructs a signature at runtime from the monitored system events and API calls in Symbian OS. Normal activity is distinguished from malicious behavior by training a classifier based on support vector machines (SVMs).

Device level approaches involve the self-detection based on identifying anomalous behavior of the device by intelligent software present in the device. Such techniques are applicable for malware that spread in the local proximity of the device and do not necessarily use the cellular infrastructure. Techniques for local detection based on behavioral patterns [19] and energy signatures [54] of infected devices have been proposed and studied.

The behavioral and signature recognition systems are excellent for PCs and desktop computers. Recent proposal to use the memory-printing of client devices is in Jakobsson *et al.* [45]. Memory-printing is a novel and light-weight cryptographic construction whose core property is that it takes notably longer to compute a function if given less RAM than that for which it was configured. This makes it impossible for a malware agent to remain active (e.g., in RAM) without being detected when the function is configured to use all space that should be free after all active applications are swapped out.

## 2.3 Drawbacks in existing approaches

Understanding the dynamics of the mobile malware and using the results obtained to gain insights into the spreading patterns and try out different interventions strategies requires a high-resolution study environment. The practical use of such studies is that one can implement, and try different deployable strategies in case a real malware spreads through the network. To obtain scalable and at the same time accurate results imposes a great challenge. It is important to trade off some detail in certain aspects that do not have a severe impact on the results and consider details in other more important places. Thus, it has become necessary to understand the dynamics of mobile malware and learn about ways to prevent a major outbreak. For conducting a qualitative study of mobile malware, three important aspects are considered:

- *Track the location of the device.* Due to the proximity nature of Bluetooth networks, tracking the location of a device is important to predict the spatial spread of the malware. In the case of SMS/MMS based malware propagation, the location of the device provides information on the cell tower serving the mobile.

- *Determine the network access available to the device.* Because several next generation malware will employ multiple transmission methods (for example, e-mail, web pages, social networking applications, etc.) the network created by these devices is diverse and has different characteristics.

- *Monitor usage patterns of mobile phone users.* Smart phone usage patterns of individuals go a long way in both identifying the existence of a malware and predicting how a certain malware may spread over the network.

Existing approaches to studying mobile malware are limited at different levels and no one proposed method addresses all the above aspects. Some of them [94] consider lower resolution information for mapping the devices (although this is good enough for SMS/MMS malware). Others [98, 100, 103, 89, 31] use small size networks, and detailed packet level simulations to study the proximity-based propagation. The study of the spread of malware through simulations can be accomplished with varying levels of accuracy, complexity, and scalability.

There are several drawbacks in the approaches outlined above. All the mathematical models outlined above make simplifying assumptions to make modeling tractable. Some of the models proposed do not consider the worm characteristics or the uncertainty of Bluetooth connections. The Bluetooth model developed in [99] considers a detailed representation of Bluetooth but makes several simplifying assumptions to make the modeling tractable. They assume a uniform mixing of devices and a Poisson arrival process for new neighbors that are not true for realistic human mobility. These modeling techniques also require steady state conditions in the mobility. With activity-based mobility models, the arrival and departure processes are not Poisson, and the density of locations vary with time as people enter and leave locations according to their activities. Thus, the device densities at these locations are not the same throughout. The mobility of individuals create time

varying interactions, creating completely different network characteristics during a normal day. Thus, the assumptions of steady state conditions do not hold.

As the authors themselves claim, the mobility model in [83] is a highly simplified representation of human mobility. Because they apply this model for low density regimes, they neglect the correlation between multiple infected devices in a susceptible devices' range. In a wireless network, this can cause congestion and interfere with the infection propagation process (for example, packets collide, and interfere with each other). In real networks, density varies with time and location, and thus a model that works for low density regimes does not work for high density locations.

Simulation studies using network simulator (such as NS-2) on the other hand represent excellent detail in malware's implementation, protocols required, and the physical layer. This level of detail to represent the per-node behavior makes them unlikely to be applied to large networks. Simulations with NS-2 for networks with 500 devices in a single location take wallclocks of over 30 hours to simulate 4 hours of device interactions. Thus, most existing studies that use simulation are limited to small scales of 100 or 200 devices at most. Other simulation studies in the network science (for example, [94]) do consider a large number of devices, but at the lowest level their simulations use compartmental models assuming uniform mixing. They also fail to consider aspects related to the nature of the communication medium. The uncertainties included in wireless communication cannot be completely neglected while studying their propagation.

Our goal here is to study large scale networks with millions of devices and study aspects to answer policy-and system-level questions. Our approach is to consider an intermediate option that represents the nature of the malware, the communication protocols, and infrastructure in an abstract manner and use realistic networks that evolve as people go about their activities. At the lowest level, the network we construct reflect realistic nature of human mobility and aid in the construction of more realistic device interactions that can change the spread characteristics.

Table 2.3: Comparison of EpiNet with other approaches in literature. We highlight the important aspects of each previous work and list the differences from our work.

| **Factors** | **Analytical results** [99] | **Simulation based computational models** | | |
|---|---|---|---|---|
| | | **Random mobility** [98, 100] | **Mobility model based on real data** [94] | **EpiNet** |
| Scope | Single location/city area | Single location | Large area | Large area |
| Temporal Scale | 1 second | ms. / $\mu$s. | Time unit (time to infect) | Time unit (30 s. resolution) |
| Spatial Scale | meters | meters | meters | meters |
| Mobility model | Random waypoint model | Random Waypoint, Random Walk, Random Landmark | Cell tower position from mobile call data | Activity-based mobility model – assigns locations |
| Device interaction network | Dependent on mobility model parameters | Based on mobility models | Homogeneous distribution of devices within each tower | Heterogeneous distribution of devices based on activity locations |
| Within-host Malware Model | Analytical expression | Detailed implementation | Compartmental models (SI) | Both high resolution and low resolution models |
| Detection | Can be represented in certain cases (model specific) | Can be implemented | Not studied | Network- and device-based detection (abstract implementation) |
| Control mechanisms | Can be done individually for each model | Can be implemented | Not studied | Expressive, adaptive & non-adaptive responses |

# Chapter 3

# The EpiNet Simulation Environment

Malware, also known as malicious code, malicious software, or worm, refers to a program that is covertly inserted into a system with the intent to compromise the confidentiality, integrity, or availability of the victim's data, applications, or operating system [65]. Mobile malware, although a mere nuisance so far, is starting to become a significant external threat to individuals and organizations [38]. Current and next generation smart phones come equipped with wireless technologies for short range communications, e.g., Bluetooth. While these are useful to form dynamic, ad-hoc networks (also called MANETs), they provide a new avenue for increasing the device-to-device spread of such malware.

Existing simulation platforms [76, 85, 41, 2] can provide a detailed network representation. They have one major drawback – they perform simulation at packet level and do not scale to large, dynamic, device deployments. Other methods [103, 89] use traces gathered from real sensor deployments for malware studies. The problem with trace data is its limitation to the location of the sensors. So, it is only useful for within location modeling of the malware and cannot be used for large spatial area used in realistic studies. Some studies use random mobility models (models such as random waypoint, random walk, random landmark, etc.) [98, 100].

All the above approaches have the following limitations:

1. *Lack of detailed dynamic device structure.* Previous work uses aggregate or collective device network models with unrealistic assumptions of node distributions and interactions. Where sensor data is used, the interactions cover a smaller region, a building or a large conference hall, or some buildings within a campus.

2. *Simplistic mobility models.* Mobility has a significant impact on worm dynamics. However, most prior art uses simple mobility models (e.g., random waypoint, random landmark, etc), which are known to be very different from realistic urban mobility models. Human mobility shows a high degree of temporal and spatial regularity and a significant probability of returning to highly frequented locations [40]. There is a lack of real data on urban mobility and collection of such data involves privacy and logistical issues.

3. *Scalability.* Existing approaches make several assumptions to improve scalability at the expense of accuracy. These techniques are not able to scale beyond a few hundred nodes once any of these assumptions are relaxed.

Following the analogy with human epidemics, researchers have defined Internet epidemiology (e.g., [63]) as the study of the spatio-temporal spread of malware on communication networks. In this dissertation, we are interested in *wireless epidemiology* – the study of malware spread over digital devices using primarily short range contacts. A number of mathematical and simulation based approaches have been proposed to study epidemiological problems in wireless networks [83]. A promising approach that has evolved in the last few years is best termed as a "network-based approach", in which a detailed representation of the underlying communication network is used along with parameterized models of infection progression and transmission. In this dissertation, we propose to use a synthetic population and their activities to generate mobility patterns. The arrival-departure at individual activity locations are converted to proximity network representations. The simulations are conducted by a parallel discrete-event simulator. In this chapter, we discuss the simulation environment and the modules we have created within the framework to study mobile malware.

In this chapter, we present EpiNet, a parallel, agent-based simulation environment that scales to

large, dynamic networks. We describe the construction of the proximity-based networks within locations to study the spread of proximity malware using Bluetooth interface. We discuss mthods used to construct realistic communication networks that are used to study the spread of malware over the cellular infrastructure through short messaging service (SMS) and/or multimedia messaging service (MMS). Any network-based approach consists of a parameterized model of infection progression – within-host and across-host propagation models. We describe this model next. We discuss the calibration of these models using calibration experiments. Lack of real data on actual malware spreads makes this process hard to validate. We use controlled simulation exercises to calibrate and validate the model for the malware. We then describe the EpiNetsimulator and its algorithm. We implement this algorithm as a parallel simulation tool using the message passing interface (MPI) library. We also discuss the interventions schemes that are implemented within the framework.

To the best of our knowledge, this is the only modeling environment that scales to millions of devices over large urban areas. The modeling environment provides an expressive environment to specify and study several complicated static and dynamic intervention schemes.

## 3.1 The EpiNet Infrastructure

Figure 3.1 is a pictorial representation of the modules in the framework. In general, the framework we describe in this section is applicable to study any kind of malware spreading through wireless networks. In this chapter, we describe the framework under the assumption of malware that spread in mobile handheld devices such as smart phones and consider only proximity networks. The framework consists of two phases as outlined below:

- *Construction of realistic contact networks.* We construct device contact (or wireless) networks using activity-based mobility of people. This mobility model considers mobility based on activities they perform during a *normative* day and constructs proximity networks.

- *Construction of abstract worm model.* We construct a high-level model for the malware based

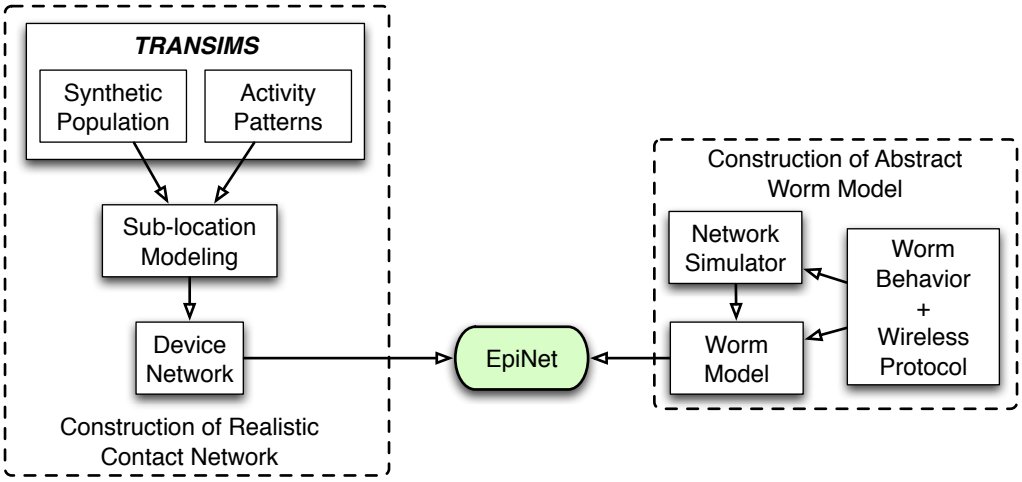



Figure 3.1: The EpiNet Simulation Infrastructure. The infrastructure contains aspects of realistic network creation and abstract simulation of malware.

on the protocol used by the malware. This model provides an abstract model of the Bluetooth worm.

For the construction of a realistic wireless network to study the spread of the malware, we consider a synthetic population as generated in the Simdemics simulation framework [8], primarily used for human epidemic studies. The Simdemics framework considers activity-based mobility models to obtain mobility patterns for this synthetic population. Because devices are carried by people, we also consider the same models to mimic device mobility. Modeling the worm helps in abstracting the details of the wireless protocol and the worm behavior and improves scalability. More detailed explanation of the modules is provided in later sections.

## 3.2 Construction of Realistic Networks

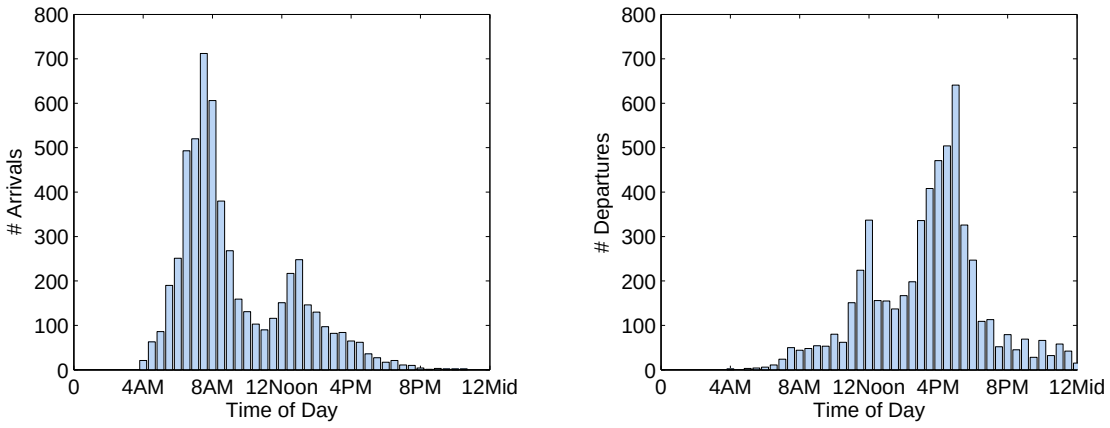
### 3.2.1 Construction of Realistic Proximity Networks

Construction of a wireless network is important for evaluating any network protocol or algorithm. Wireless networks are constructed randomly or through synthetic data. There are benefits and

drawbacks in each of these methods and the problem studied guides the method. In this proposal, we are interested in studying the problem of wireless worms that spread through personal communication devices. So, modeling human mobility accurately is important to observe a realistic nature of the malware. Further, modeling actual human mobility also helps us to create wireless contact networks that are different from random networks.

In this section, we describe the *activity-based* mobility model (ABMM) used for generating human mobility. First, we describe the synthetic population that forms the basis for the activity-based mobility. Second, we describe the generation of activities for the synthetic population. Lastly, we describe the process of converting the activities into wireless networks of a certain range.

#### 3.2.1.1 Synthetic Population

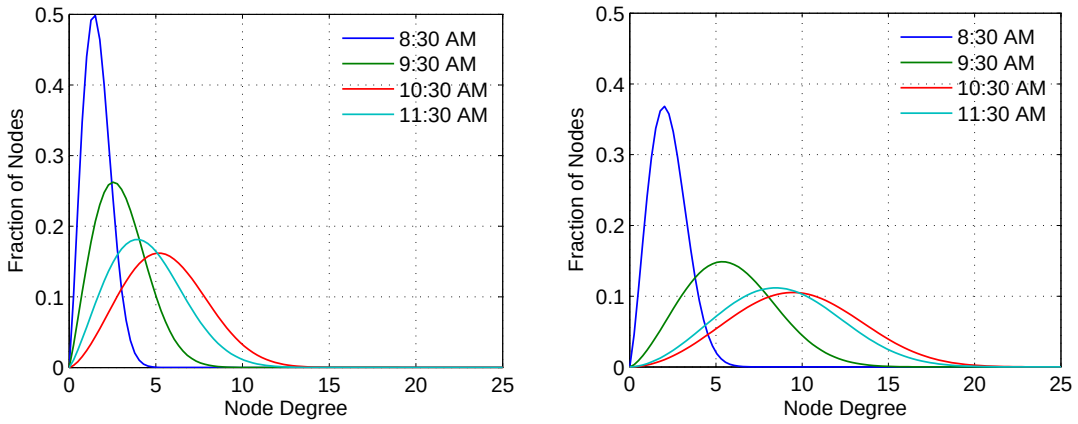



(a) (b)

Figure 3.2: Arrivals and Departures at activity locations created by the activity-based mobility models. Both arrivals and departures vary temporally and do not follow any particular pattern for the locations. They also vary across location types (not shown here). **(a):** Arrivals at the location for the day; **(b):** Departures from the location for the day.

A pre-requisite to the understanding of the spread of wireless malware is a realistic representation of the wireless network we need to study. Mobile devices and smart phones are used by people going about their daily activities and using devices when required. These activities expose the

devices to other smart devices and networks—creating dynamic and arbitrary network structure. In this proposal, we use synthetic population generated by the Transportation Analysis and Simulation System (TRANSIMS) developed at Los Alamos [5] to construct realistic wireless networks. TRANSIMS has been extensively used for a several transportation studies. In this section, we briefly discuss how TRANSIMS generates the synthetic population to provide perspective for the future sections where we construct wireless networks created by devices carried by the synthetic population.




(a) (b)

Figure 3.3: Temporal Degree Distributions at different location types. Degree distributions for a work and shopping activity location at 8:30AM, 9:30AM, 10:30AM, and 11:30AM, respectively. As the day progresses, the number of people in each location increases and the network created has slightly higher variance in the people's degree. This discrepancy is more pronounced for shopping locations than work locations. **(a):** Work location; **(b):** Shopping location.

The basic algorithm for the construction of a synthetic population is based on aggregated, block group level, census data. Data about land use and demographic information, combined with survey data from thousands of households, is employed to create a household location for the synthetic population. The output of the synthetic population generator is a *proto-population* [1]. The synthetic population was constructed for conducting micro-simulations in activity-based models using

[1] A collection of synthetic individuals, each associated with demographic variables drawn from census data.

aggregated demographic characteristics in a four step process. A sequence of daily activities and the locations where these activities are performed is determined for each person, based on activity surveys, travel time, and land use data. At the end of these four steps, TRANSIMS produces positions for the people in the synthetic population on a second-by-second basis for a large metropolitan area and has effectively been used to construct detailed mobility patterns for entire cities. We use this synthetic population and activity sequence for each individual to construct networks.

Figure 3.2 shows the arrival and departures at a location. These arrivals and departures do not follow any particular distributions. As a result of these arrivals and departures, the density of locations are time varying, creating very dynamic wireless networks. Further, the density of every location does not follow the same pattern and depends on the time of day and kind of activity location. Over a larger physical region, the activities create components with differing spatial and temporal characteristics. The temporal degree distributions at different locations are shown in Figure 3.3. Here, we assume a certain area for each location and construct a random geometric graph (RGG) and plot the degree distributions for different times of the day.

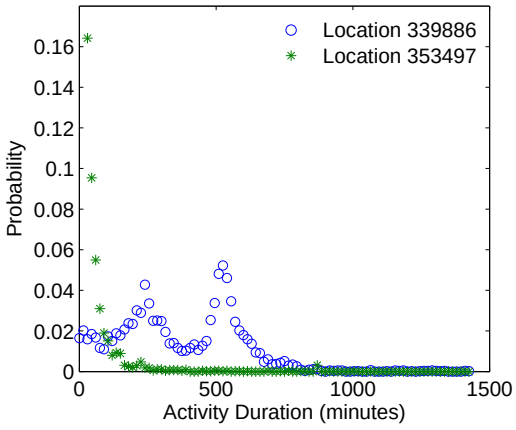



Figure 3.4: Activity duration distributions at two different locations in the NRV dataset. The activity durations are not similar across all locations.

Figure 3.4 shows the activity durations at two different locations in the New River Valley dataset. Clearly, the activity durations at each location are different and are based on the individuals' activity patterns. For location 353497, the activity duration is exponential while location 339886

has a completely arbitrary distribution. The activity patterns generated in TRANSIMS provides the basic information required to construct interaction networks. For human disease models, the arrivals and departures at locations is used to determine interaction networks at locations. Similarly, for the study of wireless networks, we construct device networks that are created at locations. This network is constructed by *modeling sub-locations*.

#### 3.2.1.2 Modeling Sub-locations

The activity-based mobility model generates information regarding the participants in the network created at locations. The actual wireless link at each individual location is required to study wireless networks. This is performed by modeling sub-locations. In this section, we describe the process of converting the activities into a wireless network which is then used to study the propagation of the worm. A *sub-location* is defined as a smaller area within a location that models the range of devices.We use sub- location modeling to build a *device contact network* within a location. Throughout this proposal, we refer to device contact network and device network interchangeably. It is an abstraction that allows us the flexibility to model range of a wireless interface. Because we are interested in studying Bluetooth devices, we model Bluetooth networks. Nevertheless, the sub-location modeling approach provides a clean abstraction to model mobility within location and wireless propagation effects. Abstractly, one can assume that devices co-located in a sub-location can communicate. Combining multiple neighboring sub-locations can provide inter-sub-location links. Simply by altering a devices' sub-location, we can model time varying links between sets of devices.

Currently, we just model wireless medium without errors and neglect indoor propagation effects. We assume the range of Bluetooth devices to be $10\,m$ as for Class II Bluetooth devices. Before going into the details, we make the following assumptions while generating the wireless networks:

- Although transit between activity locations can contribute to the spread of the malware, we neglect them. Nevertheless, transit can be modeled as an activity, with transit time as visit durations at a *transit location*, and worm propagation modeled. For the purposes of this

proposal, we neglect the transit locations.

- At present, we do not consider mobility of devices within locations. All devices are assigned random positions within the location and they remain there until the end of the activity. Essentially, we create a geometric random network within locations and use this to study the propagation.

- Because we do not have data regarding the areas of the locations where the synthetic population performs activities during the day, we make assumptions based on the occupancy of the locations. Currently, our studies can be conducted with varying densities to study the effect of density on the spread.

- We also make some assumptions that improve the scalability without significant change to the spread. We assume the arrivals and departures of people at activity locations rounded to 5 minute intervals. Although this alters the arrival and departure, it does not significantly impact the spread dynamics while improving scalability of simulations.

We divide each activity location into multiple areas called *sub-locations*. Unlike social contacts of people where physical proximity has no bearing, device contact networks are formed by devices in range. Modeling sub-locations helps to isolate events and provides an opportunity to parallelize the algorithm used in the study. Now executing events at sub-locations are independent of other sub-locations. Because wireless networks are based on distance, there is some dependency between sub-locations and is handled correctly at simulation time. Modeling sub-location in this way only creates a coarse device network; i.e., all devices belonging to a sub-location are connected to each other and form a clique, irrespective of the actual physical distance. Some of these devices may be out of range when considering a particular communication protocol. During simulation, the actual distance between devices is taken into account, and worm propagation occurs only among devices within *range*. Sub-location modeling provides flexibility to use different models for assigning devices to sub-locations, helps consider mobility of users, and allows different wireless technologies with varying ranges to be emulated. For example, to change the wireless technology to be IEEE 802.11 based ad hoc networks, we need to change the size of the sub-location and the range the simulator

uses while propagating the infection. The simulator can be extended to handle link reliability and signal propagation internally.

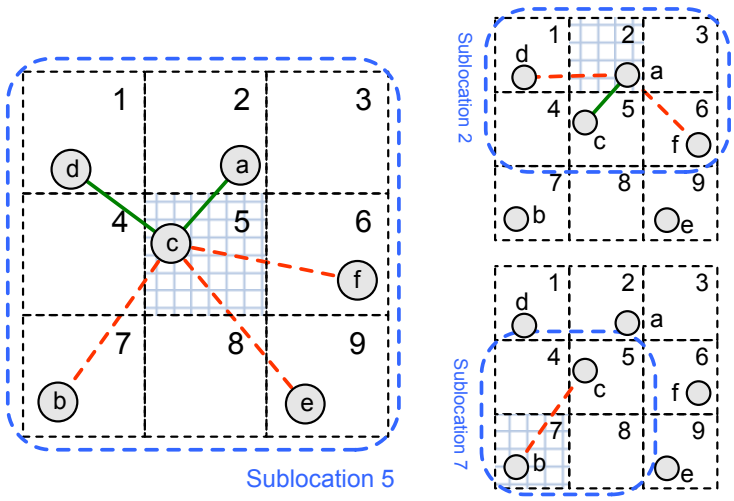



Figure 3.5: Modeling Sub-locations for creating wireless networks. Device network created through sub-location modeling. Solid (green) lines indicate devices in range and dashed (red) lines indicate out of range devices.

The sub-location modeling follows the following steps:

1. Compute occupancy of the building using the activity information.

2. Assign an area to each location based on the occupancy of the location. One can obtain this from actual building data, but for lack of this information, we are assuming the area. Sometimes, we assign the area based on the required density value for the locations.

3. Assign random positions within this area when people arrive at the locations. These positions will also be the device positions in the location. In Figure 3.5, devices $a$–$f$ are assigned random positions in the location and are shown in those positions.

4. Divide the building into grids whose size is derived based on the range of the device being simulated. In this case, we are using the range of Bluetooth devices, 10 $m$. We choose a grid with side of 10 $m$ to approximate the range of Bluetooth devices. Figure 3.5 shows the grids of side 10 $m$ marked with numbers for each grid cell.

5. Form sub-location such that devices in adjacent grid positions are added into the same sub-location. This implies that each sub-location contains multiple neighboring grid cells obtained in step 4. From Figure 3.5, 'Sub-location 5' has all the grid positions from 1–9 in the sub-location, implying that all nodes $a$–$f$ belong to the sub-location for grid position 5.

6. Output the sub-location allocation into a file for the simulator to process.

In Step 4, we consider a certain block size of 10 $m$ $\times$ 10 $m$ for Bluetooth proximity networks to divide the location into grid cells. The block size is derived from the range of the wireless technology we consider and can be appropriately varied to consider different technologies. For IEEE 802.11 based wireless network, we can consider a grid size of 250 $m$ $\times$ 250 $m$ (if we assume a range of 250 $m$ for the interfaces). The grid blocks are used in Step 5 to form sub-locations and decide the devices' sub-locations. For example, from Figure 3.5, Sub-location 5 consists of devices in all the neighboring grid cells (devices marked $a$–$f$). This is required to consider as devices can reach beyond the grid boundaries. We actually model duplicates across sub-locations to account for this. The network provided as input to the simulator consists of a clique among all devices inside the sub-location. Actual links are modeled inside the simulator. Figure 3.5 shows the edges in *range* marked as solid (green) lines and links that get dropped (in the simulator) for being *out-of-range* as dashed (red) lines.

### 3.2.2 Construction of Realistic Communication Networks

In addition to the proximity spread, current generation mobile malware also replicate through short-messaging service (SMS) and multimedia messaging service (MMS) and instant messaging (IM) clients. There are several instances of mobile malware that spread using different media in addition to Bluetooth [29, 57] and use SMS or MMS messages as a medium of replicating on susceptible devices. Realistic communication networks – e-mail, SMS/MMS and IM networks – are required to study such malware, predict their dynamics, and study aspects of their control. As instant messaging on smart phones is gaining in popularity, it can potentially serve as the next targeted application. Thus, any framework that models mobile malware has to be able to construct

and use these communication networks for estimating the dynamics and the efficacy of the countermeasures. In this section, we discuss *Session Generator* (SG), a synthetic mobile traffic generation platform, to construct SMS, MMS, and the IM networks. The session generator has been used as a means to generate synthetic voice calls based on existing statistics for mobile phone calling patterns in literature. In this section, we present the use of this module in the study of malware propagation over SMS/MMS networks. We show the architecture of the module and its ability to construct statistically accurate sessions based on the given input statistics and use it to construct the social networks that are used to propagate the mobile malware.

The SG module has been presented as a part of the Synthetic Spatio-Temporal Relational Session Modeling (SSRSM) environment in [9]. The SSRSM framework has been employed to undertake a couple of studies: primary markets for dynamic spectrum access (DSA) [10] and interaction between communication and transportation infrastructures [6].

#### 3.2.2.1 Teletraffic analysis and statistics

A significant amount of research has been conducted on network traffic analysis in wireline networks, LANs, and Internet, e.g., [78, 32, 26, 48, 17]. These studies have established that the network traffic does not follow Poisson and that the nature of the traffic is bursty. In the case of wireless networks, there have been a number of studies for very small networks, e.g., local area networks within small regions, such as a campus or a conference site [4, 55, 91]. The more recent study on a real wireless personal communications network presented in [87] provides a much better understanding of voice call patterns in real, current generation networks. Cellular call data has been used in other ways to understand aspects of human mobility [40] and spectrum usage patterns [97] in next generation dynamic spectrum management.

The problem with this data is that it contains proprietary information of service providers and their subscribers and has significant privacy concerns. Most of this data is for internal consumption to design better backhaul networks to carry service provider traffic over the wired infrastructures. Some publications like [87] hide the actual data, and the calling patterns released in the paper are

scaled. The location and the network where the data is collected is also hidden.

The goal of the session generator we implement are as follows:

- *Provides researchers access to a tool that can generate synthetic traffic.* Integrates public data and statistics collected in surveys and third party data providers and generates realistic traffic patterns (voice calls, data traffic, etc.). This data can be combined to generate individual level calling patterns for studying different problems.
- *Generate the communication networks based on the observed output.* The communication patterns seen as a result of creating sessions from these statistics provides the interaction network that can then be used to study aspects such as malware propagation, information diffusion over digital devices, and others.

#### 3.2.2.2 Architecture of the Session Generator

Figure 3.6 shows the high level architecture of the SG module. The SG is an event-based simulator that generates communication patterns for individuals in a synthetic population. The communication patterns can be voice calls or data channels between individuals (peer-to-peer communication). The patterns between two individuals are dependent on the contact durations derived as a result of the activity-based mobility of individuals. It is these patterns that we use in this dissertation to construct the communication network for simulating and studying the short messaging service (SMS) based malware propagation.

The inputs to the session generator consist of the following:

1. *Statistics for the sessions to be generated.* This includes either individual or system-level statistics for several parameters: number of arrivals, duration of the channel, and the input distributions to these parameters (standard or arbitrary distributions).
2. *Demographic based assignment of devices.* SG uses a device assignment based on systematic classification and regression tree (CART) on public data set from National Health Interview

Survey (NHIS) [21].

3. *Population demographics for which the sessions are generated.* The demographics of the people making the calls and using sessions to communicate. This includes age, activities, and activity types for consideration in the generation of sessions.

4. *Social contact network.* The social contact network is generated based on the activities performed and includes the set of people an individual can potentially come in contact with. This super-set of social contacts is then used to guide the selection of communicating individuals based on data about peer-to-peer communication patterns.

The outputs from the session generator are the sessions of the individuals in the population, the start, duration, and devices used for the session.

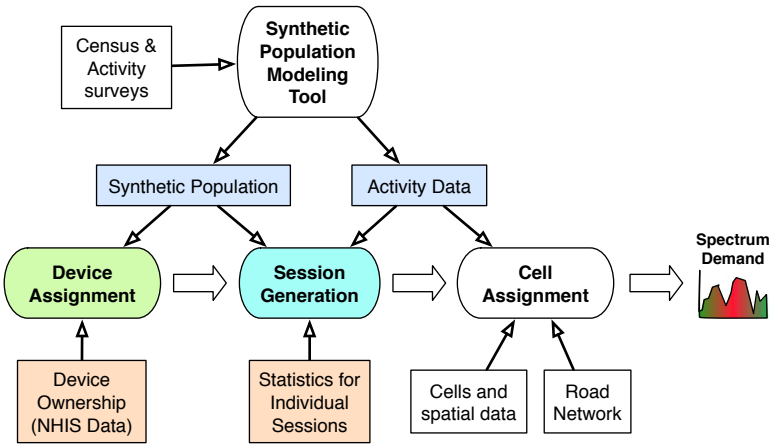
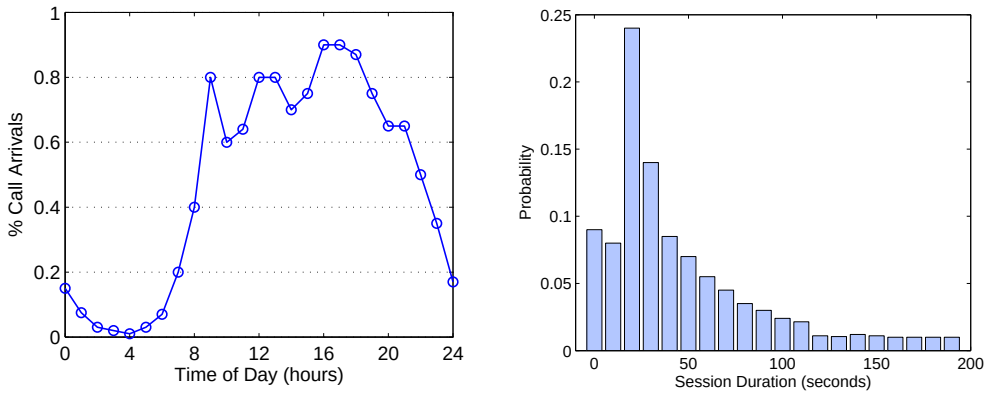



Figure 3.6: Architecture of the Session Generator implemented for constructing realistic communication networks. The boxes with color fills correspond to the actual parts of the SSRSM that are used for generating the calls to build the communication network.

#### 3.2.2.3 Implementation of SG

SG is implemented as a simple event-based simulator where the arrivals and departures of communication instances are events. Each event is processed independently to create sessions for the




(a) (b)

Figure 3.7: Example input statistics for the Session Generator. Input statistics for the number of arrivals and session duration distribution to model communication patterns of individuals. These arrivals decide the frequency with which the calls are made in the synthetic population.

individuals. The current implementation uses global hourly arrival rates as the input to generate the the `Arrive` event for the individuals. This statistic is obtained from [97] and we scale this to different configurable values to obtain realistic arrival patterns. Note that the data provided in [97] are scaled and we do not know the exact values. We conduct a parametric evaluation of the scale and use it to generate the sessions. The base values of the unscaled number of arrivals for each hour during the day is shown in Figure 3.7a. When the `Arrive` event is executed, the other end point can determined from the social contact network based on several parameters. We select random individuals from the population to satisfy this arrival rate. The other set of input statistics we use is for the duration of the communication channels between individuals. We use LogNormal distribution for modeling the session durations based on [56]. Note that the session duration is different for each application; we are not interested in mimicking the application but need a communicating link to generate the communication network.

The EpiNet simulation framework initializes each individual device with the set of edges based on this network. When a device becomes infected, it propagates the infection by selecting individuals (at random or based on frequency of contact) from this social network. We consider different studies

to determine the effect this has on the propagation of hybrid malware that spread over multiple interfaces available in a mobile device. The implementation of the SMS/MMS diffusion process is described in Section 5.6.1. The propagation dynamics of the SMS/MMS and hybrid malware are studied in Section 5.6.

## 3.3 Within-host malware model

Malware studies in literature model the nature of the malware at different levels. Mathematical models use the protocol of the malware to devise a concise way to represent the conditions that matter to the spread of the malware. In this chapter, we propose one such high level model to represent the characteristics of the malware. We approach the model as a representation of the malware in each individual and use it the agent-based framework proposed here. First discuss the aspects of the Bluetooth malware and the steps it takes to propagate itself to other susceptible devices. Our model requires calibration to provide realistic results. We describe the experiments we conduct to calibrate the model and use the calibrated model for validating the results. The validation is performed in comparison with detailed simulations conducted with NS-2, a network simulator. We then discuss the preliminary work we have conducted in reducing the model to improve scalability of the model to target large scale studies. We propose to study this in more detail in the future.

### 3.3.1 Modeling the Bluetooth Malware

In this section, we describe the design of the Bluetooth malware model. We follow a top down approach: Firstly, we discuss the malware protocol, i.e., the stages of an infected device during each infection cycle. Secondly, we describe the design of the model based on this protocol. Thirdly, we discuss the characterization experiments we conduct using NS-2 to calibrate the high level model. We discuss the parameters of interest for the characterization. Lastly, we discuss the validation of the model using the calibrated model. Table 3.1 defines some terms used in the malware protocol description.

| Terms | Definition |
|---|---|
| $N_{resp}^{to}$ | Maximum inquiry responses expected by the inquiring device during an inquiry request. When an inquiring device receives $N_{resp}^{to}$ responses from neighboring devices, the inquiry terminates and the next process is begun. |
| $N_{resp}$ | Actual number of inquiry responses received by the inquiring device. This effectively means that the inquiry process timed out. |
| $T_{inq}^{to}$ | Inquiry timeout value when the inquiring devices terminates the inquiry and proceeds to the next stage. This is the total inquiry time when $N_{resp} < N_{resp}^{to}$ |
| $T_{inq}$ | Actual time to complete the inquiry request. If $T_{inq} < T_{inq}^{to}$ then the inquiry stage completed with $N_{resp} = N_{resp}^{to}$. If $T_{inq} = T_{inq}^{to}$ then $N_{resp} < N_{resp}^{to}$ |
| $T_{inq}^{n}$ | Time taken for a particular inquiry request to return the response from a neighboring device. This time is measured from the time the $n^{th}$ inquiry request was started till the time the $n^{th}$ response was received from a neighboring device. |
| $T_{inf}$ | Time taken for a new device to be infected. This time includes the time taken to page the device and the times to communicate with the device to determine its susceptibility and finally infecting it if susceptible. |
| $T_{idle}$ | Idle time between infection cycles. Following every infectious cycle, each device reaches an *idle* state and remains un-infectious for the idle duration $T_{idle}$. |

Table 3.1: Bluetooth Protocol and malware Parameters

#### 3.3.1.1 Malware Protocol

We briefly describe the malware protocol in [98] for completeness. The Bluetooth malware follows four distinct steps during the infection cycle—inquiry phase, page phase, infection phase, and idle phase. During the inquiry phase, the infected device gathers information about the neighborhood. During this phase, the inquiring (or an infected) device sends inquiry requests and waits for responses from other devices. The device continues to perform this until it receives $N_{inq}^{to}$ responses or the request times out in $T_{inq}^{to}$. The infected device maintains a neighbor list consisting of nodes that respond to the inquiry. Only devices that are *discoverable* respond to this message. After $T_{inq}$ seconds, the infected device enters the infection phase, where it processes $N_{inq}$ neighbors, one at a time. This phase involves sending a *page* request to each neighboring device and obtaining a page response, sending a message that verifies the condition of this neighbor. The neighbor's condition is either susceptible, not susceptible, or infected. The susceptible node then receives the malware packet with payload to cause the infection. If the neighboring device is not susceptible or is already infected, then the packet is not transmitted. The infected device then repeats these steps with all devices in the neighbor list. Once the entire neighbor list is processed, the infected node becomes idle for $T_{idle}$ before restarting the infection process with a fresh inquiry process.

Our modeling approach incorporates the malware protocol as a probabilistic timed transition system (PTTS) called the *malware manifestation*. This model represents the various stages of the malware protocol executed by an infected device. The timing of the transitions and the probability of being in particular states of the manifestation is obtained from simulation studies. In these *calibration experiments*, we conduct a detailed simulation using packet accurate small scale malware simulations with tools like NS2 [76] or Qualnet [85]. In Section 3.3.1.2, we describe the PTTS model for the Bluetooth malware. Further, in Section 3.3.3, we provide information regarding the validation of the model and compare it to the results obtained from the detailed simulation results.

#### 3.3.1.2 Malware Model

The malware model is built to abstract the details of the actual detailed Bluetooth protocol. Figure 3.8 shows the malware modeled as a PTTS. Though [99] models the Bluetooth protocol analytically, the several modeling assumptions regarding homogeneous distribution of devices and steady state conditions do not occur in realistic wireless networks. Building analytical models accounting for the realistic conditions is intractable. Detailed simulation studies on smaller scales do not result in actionable information to tackle the malware spread and study its true impact. Abstract simulations by using abstract models for the malware under realistic network conditions motivated the modeling approach we take.

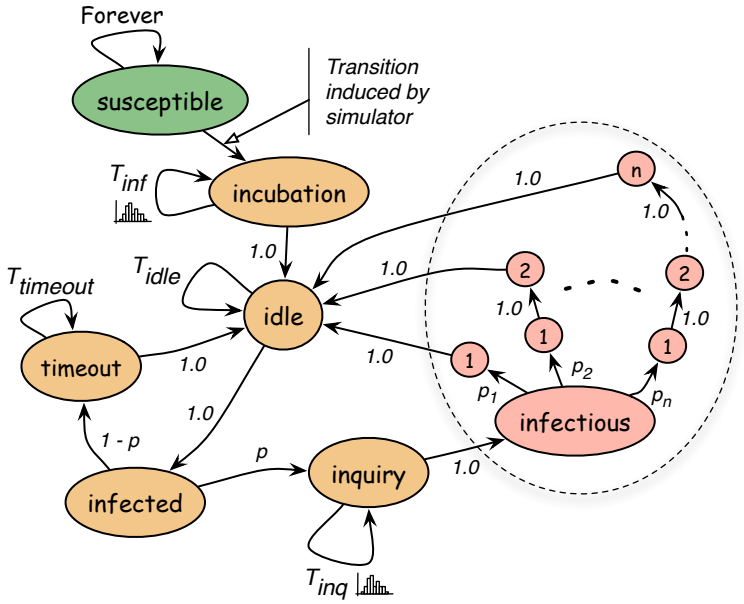



Figure 3.8: Behavioral model for the Bluetooth malware characteristics or the manifestation of the malware.

Initially, all nodes are *susceptible* and remain susceptible until they become infected through the simulation. When conducting the simulation study, we consider some nodes as infected initially and force this transition on these nodes. Once a node becomes infected, the node *incubates* for a certain time, denoted by $T_{inf}$. This is the time taken to actually pass the infection to a new node and is lumped in the *incubating* state. This time is obtained from the $T_{inf}$ histogram. When *incubating*,

the node cannot infect any other susceptible nodes–i.e., the node is not infectious. Once the device is infected, the malware begins executing its protocol. The protocol begins with an *idle* state where it waits for a certain idle time, $T_{idle}$, without spreading the infection. $T_{idle}$ is a malware specific parameter that can change based on the malware characteristics and indicates the time between two infection cycles. After this the malware enters the infection cycle by becoming *infected*. In the model, the infected device does not spend any time in the *infected* state and moves either to *timeout* with probability $p$ or to *inquiry* with probability $(1-p)$, where $p$ is the probability that the node's inquiry terminates without receiving any response. This incorporates the conditions when the inquiry terminates with no responses or obtains at least one response. The probability $p$ is obtained from the $T_{inq}$ histogram. Once in the *timeout* state the malware spends $T_{inq}^{to}$ time and returns back to the *infected* state. Because there are no neighbors discovered, the node has to perform another inquiry after $T_{idle}$ time. If the node picks the *inquiry* state, after a $T_{inq}$ time it shifts into the *infectious* state when the infectious cycle begins. The time spent in these set of states depends a lot on the Bluetooth protocol and its functioning. Therefore, the probability $p_t$ of choosing a certain cycle time $t : t \in (1, n)$ is deduced from the $T_{inf}$ histogram. This is represented in the disease model as $b_t$ branches with $t$ states in each branch. In each state $b_t^i$ where $i \in (1, t)$, the probability of infecting at time $i$, $p_{inf}^i$ is derived from the $T_{inf}$ histogram. After the last state in branch $b_t$ the node moves back into the *idle* state and the next infection cycle continues until the node remains infected or simulation is complete. Here, we make an important assumption: we assume that the probability of infecting a susceptible node at time $t_1$, $p_{inf}^{t_1}$ is independent of $p_{inf}^{t_2}$ at the next instant $t_2$. From our validation results in Section 3.3.3, we find that this seems to work very well.

### 3.3.2 Model Calibration

The model in Figure 3.8 has aspects of the Bluetooth protocol that require calibration. To perform this calibration we conduct detailed simulations in small settings to obtain certain parameters that are used to calibrate the model. Further, because we are studying the Bluetooth malware, the parameters of interest are derived from the aspects of the Bluetooth protocol that matter in the

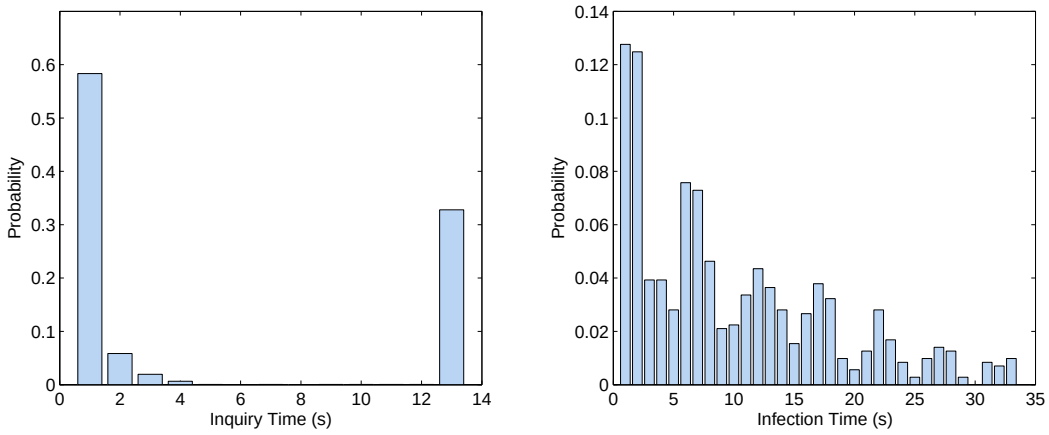
spread of the malware.

#### 3.3.2.1 Calibration Experiments

The experiments were performed using the detailed simulation implemented in NS-2 and specific parameters required were analyzed and obtained from the experiments. We have implemented a Bluetooth malware model in NS-2 for this study using UCBT model for Bluetooth protocol stack [93]. As detailed simulations are not scalable, we consider a single location with 700 people arriving and departing at various instants during the 4 hour interval of the simulation. The arrivals and departures are based on the activity-based mobility model and the arrival and departure times are rounded to 300 $s$ intervals—no new arrivals/departures during the 300 $s$ interval. We assign an area of $120 \times 120 m^2$ to the location and generate the device positions randomly in the area. We neglect mobility of devices in the location and the device remains at the same position allocated for the duration of the stay. We randomly select the individual to be infected and start the spread of the malware. We make some assumptions for the purposes of simplifying the calibration experiments. Nevertheless, they do not alter the procedure for a study dropping some of these assumptions.

- *All devices are discoverable.* This ensures that all devices respond to the inquiry request from any neighboring device.
- *All devices are connectable.* This means that connections can be established to all the devices irrespective of their status, infected or susceptible.
- *All devices are susceptible.* For the purposes of the calibration we are assuming that all devices are in either of the two states, susceptible to being infected, or infected.
- *No errors are introduced in the medium.* The wireless transmissions are error free and the malware inquiry request or the malware code itself have no errors during transmission.

The parameters of interest from this study are outlined in Section 3.3.2.2. We extract these parameters and use them to calibrate the model. The calibration experiments take approximately 48




(a) (b)

Figure 3.9: Results for inquiry and infection time obtained from the detailed simulations. **(a):** Inquiry time distribution from detailed simulations conducted at a particular location; **(b):** Infection time distribution from the same simulation study.

hours for a simulation duration of 4 hours. Clearly, this cannot be used to make a detailed study and analysis of the malware at a decent scale.

#### 3.3.2.2 Calibration Parameters

From the malware protocol described in Section 3.3.1.1 it is clear that the inquiry process has to complete either with the $N_{resp}$ number of responses, or the inquiry terminates after $T_{inq}^{to}$. Because every Bluetooth interface is performing frequency hopping, the time taken to receive a response from a neighboring device is a random number. Similarly, the time taken to complete the inquiry request is also a random number but depends on the density of devices in the neighborhood and $N_{resp}^{to}$. Figure 3.9a shows the distribution for the inquiry time for the detailed simulation studies.

Once a device is infected, it periodically cycles through an idle phase during which it does not propagate the infection, and a infectious phase. The total time the infected device spends in the idle phase is constant idle time $T_{idle}$. The total time the infected device continuous to infect other devices ($T_{inf}$) is dependent on $N_{resp}$, the number of those with which a connection can be established

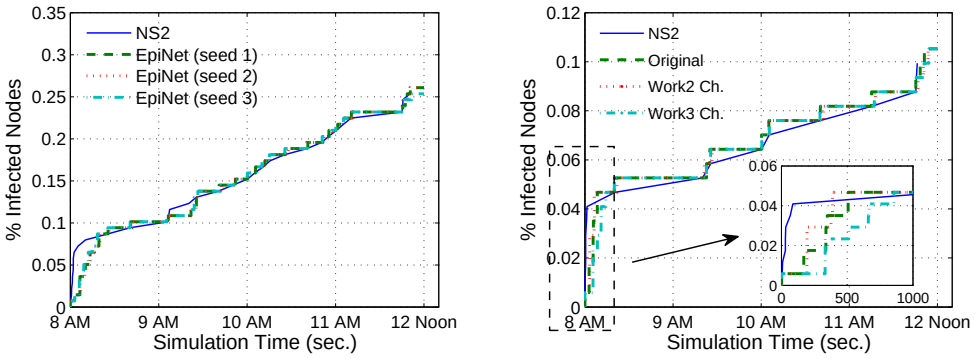
and whether the connected device is infected or susceptible. Figure 3.9b shows the distribution for infection times in the detailed simulation study. Thus, the parameters we are interested in evaluating from the detailed simulations are the distributions for $T_{inq}$ and $T_{inf}$, $T_{idle}$ value for the malware distribution, and the probability that the inquiry times out (obtained from the last bar in the $T_{inq}$ distribution).

### 3.3.3 Validation of malware Model

Before using the model to conduct the studies for the spread of the Bluetooth worms, we first need to validate the model. We validate the malware model (Section 3.3.1.2) by comparing the results with detailed packet level simulation using NS-2. For this comparison, we first calibrate the model from exact simulation with specific parameters—inquiry time and infection time distributions and inquiry timeout. We use the calibrated model for validation. We implement a Bluetooth malware model in NS-2 using the UCBT's Bluetooth [93] implementation. Because packet level protocol simulations are not scalable, we consider scenarios with 100–400 devices in a single location, with activities over a 4 hour period during the day. Note that we use activity-based mobility model in NS-2 for the calibration and validation experiments. The malware packet size of $20,000$ bytes and probe packet of 27 bytes were used for the simulation studies as in [99].

The assumptions for the calibration experiments hold here as well. For the experiments with the EpiNet simulator we use the abstract Bluetooth malware model we describe in Section 3.3.1.2 and calibrate it with the experiments in Section 3.3.2.1. We compare the results of the simulations in terms of the number of infections and the time taken. We find that the EpiNet study for the same one location scenario in the calibration experiments takes 25 minutes (the detailed simulation study takes approximately 48 hours or 2 days to complete the same).

Figure 3.10 shows the comparison between the cumulative infections in the NS-2 simulations and EpiNet. We find that the results are comparable and extremely close. Considering that the simulation granularity for EpiNet is 1 $s$, the infection growth tracks the growth observed by the detailed simulations. Figure 3.10a shows the comparison between various seeds of EpiNet's simulations with




(a) (b)

Figure 3.10: Malware Model Validation with NS-2 Simulations. **(a):** Comparison between NS-2 simulation results with the results from the malware model at work location 2; **(b):** Effect of characterization at location on simulation of another location.

NS-2. Here the model is calibrated using NS-2 simulations from the same location. For example, if work1 was the location used to calibrate the model, EpiNet is run on the same location with the calibrated model. In the large scale simulations that we need to study, there are tens of thousands of activity locations and so the model cannot be calibrated for each location individually. We also need to validate is the impact of using the model in location $L_2$ while the model is calibrated using $L_1$. Figure 3.10b shows exactly this. We calibrate the model using location work1 and use it on the different locations (work2 and work3). We find that using a different location to calibrate the model does not affect the results significantly. There are slight differences, but they are marginal when compared to the benefit we gain by using a model calibrated from a single location across all locations.

Figure 3.10a shows the comparison between the exact simulation and the Bluetooth malware model for a location. We can see that the simulation using the Bluetooth model tracks the infection CDF for the packet level simulation. The time taken for the NS-2 simulations for this setting was 48 hours (2 days); the simulation using the malware model with EpiNet took 10 minutes. There is a huge time advantage in using the high level simulation, and the accuracy is good in comparison with the detailed simulation. For this comparison, we have characterized the models using the $T_{inf}$

histogram for location $l$ ($H^l_{inf}$) with 1 $s$ as the width of each interval (for example, any $T_{inf}$ between 0 and 1.0 $s$ is counted in bin 1, and so on).

## 3.4 The Parallel Simulator

In this section, we provide the details regarding the parallel simulator in the EpiNet framework. We implement a parallel discrete event simulator (PDES) where *arrival*, *departure*, and *disease update* events are executed to perform the simulation of malware propagation at activity locations. The basic structure of the simulator is borrowed from the human epidemics simulator, EpiSimdemics [8]. EpiSimdemics is a scalable parallel algorithm to simulate the spread of contagion in large, realistic social contact networks using individual-based models. It is an interaction based simulation of a class of stochastic reaction-diffusion processes [8]. The scaling in EpiSimdemics is obtained by exploiting the semantics of disease evolution and scales to 100 million individuals.

There were several modifications made to EpiSimdemics to study mobile malware. EpiSimdemics exploits the semantics of disease evolution and performs several optimizations and thus achieves scalability. The same does not hold for mobile malware as they have completely different characteristics: 1. *Time scales of mobile malware are similar to the Internet worms while the spatial nature is similar to human epidemics.* An infected device becomes infectious immediately after the infection and does not have the same incubation period as a human epidemic (normal incubation periods for human contagions extend beyond a day, and sometimes even weeks). Mobile malware simulators do not have the luxury of such extended incubation periods. 2. *Wireless devices can communicate with multiple devices at much larger distances.* Human epidemics have a much smaller spatial effect than certain wireless technologies. This complicates the infection processing parallelization task. This required modifications to the disease update algorithm and each agent carries the disease state to the activity location. The event handling takes care of updating the state of a device immediately after infection. The range based wireless networks reduces the parallelization capability of EpiNet, and modifications to the visits of the individual ensured that each individual is present in multiple sublocations simultaneously. This requires proper handling of the infection propagation. Details

are provided in the next section.

Each agent is an individual in the synthetic population and maintains a within-host state of the infection through a *contagion manifestation* or *disease model*. EpiNet embeds a Bluetooth malware's model or manifestation within each device. The manifestation is a probabilistic timed transition system (PTTS) model of the Bluetooth and worm protocol as discussed in Section 3.3.

### 3.4.1 EpiNet Implementation

Several factors inherent in computer malware behavior require modifications to the EpiSimdemics' algorithm summarized above. Contagions affecting humans have longer incubation periods, sometimes hours, several days, or weeks. EpiSimdemics exploits this nature in implementing the parallel algorithm. Unlike human contagions, which have longer incubation periods, mobile malware or any computer worm becomes infectious immediately after the infection. This reduces the opportunities for exploiting parallelism available to EpiSimdemics. Thus, scaling EpiNet is much harder and it is identical to a parallel discrete event simulator. The only scaling that can be exploited is the parallelization of the event execution at locations. Another aspect of the Bluetooth or wireless malware is the network. Unlike human contagions which can propagate through the air, wireless malware can spread only to devices that are in range of the infected device. In case of Bluetooth, this is 10 $m$, and for IEEE 802.11/11e/11n standards the range is higher. Each sub-location in EpiSimdemics creates a mutually exclusive event boundary allowing parallelization of event execution to the sub-location level. EpiNet, on the other hand, performs sub-location modeling differently, as outlined in Section 3.2.1.2, and creates random geometric graphs (RGG) in locations. This results in overlap in the sub-locations and does not allow for sub-locations to execute events in parallel as there is some dependency between sub-locations.

EpiNet implementation accounts for the newly infected devices to become infectious immediately by designing the individual to maintain a local copy of the disease manifestation. Essentially, the individual carries the manifestation to the location. When the activities of a particular individual are processed to construct the visits, in addition to creating arrival and departure events at times

when the individual arrives at a location and departs from the location, the disease manifestation is created. The creation of the manifestation for each individual allows the simulator to track the progress in the disease states of the manifestation. Because we round the arrival and departures for 300 $s$ intervals, the arrival and departure events are executed at that granularity. The disease states, on the other hand, require much more frequent updates; in the worst cases, they are updated every second. We use the *disease update* events to perform the state transitions in the manifestation.

The second aspect is the distance requirement for wireless communication links. Because Bluetooth has a range of 10 $m$, the EpiNet simulator is required to transfer the physical location of the device at a sub-location and use this for creating communication links between devices at runtime. Only if devices are in range can Bluetooth worm propagation occur. This is a configurable parameter and can be altered for another kind of wireless network.

#### 3.4.1.1 Data Partitioning and Algorithm

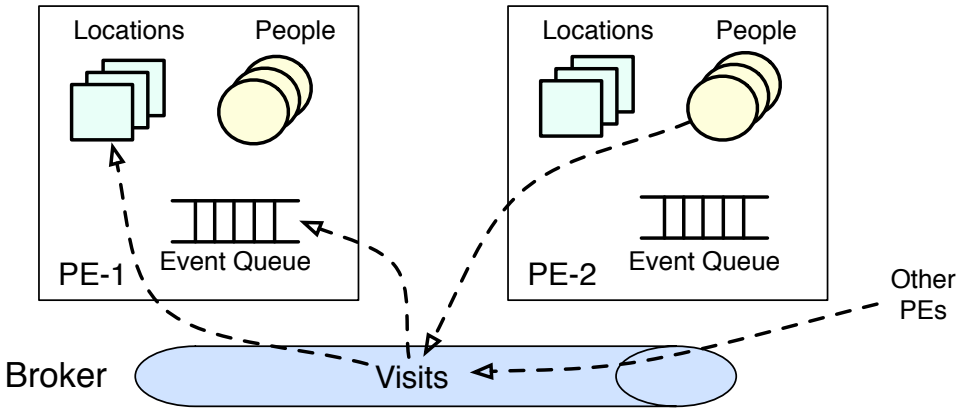



Figure 3.11: Implementation of the data partitioning and event creation in EpiNet. Each PE shares a set of locations and individuals and maintains an event queue for the events that are processed in a PE. Individuals (from other PEs) that perform activities in locations of PE-1 create events in PE-1's event queue.

Before describing the algorithm for EpiNet, we outline the scheme used for parallelizing the data. Figure 3.11 pictorially represents the data partitioning and the way the events are created for

visits to a location. The data, in this case the synthetic population and the locations, are divided into every processing element (PE). The way this division is performed is similar to EpiSimdemics. Current implementation performs a simple division based on the individual's identification number. Although this can provide better performance if a more intelligent mechanism is used, we do not implement it in this version. We define the PE assigned with a block of individuals and locations as the *home* PE. When an individual performs an activity at a location, the home PE transmits the visit information to the location's home PE. If the home PE for the individual and location are the same, no transmission takes place. Only simulation triggered updates, such as when a device becomes susceptible or crashes – i.e., the device is removed from the simulation, or the device is patched and updated by a message to the home PE of the individual. At the location's home PE, the arrival at and departure from the location are controlled through *arrival* and *departure* events executed at the location PE.

Each PE performs the steps as outlined in Algorithm 1 every $\Delta t$. In our simulation $\Delta t = 1\ s$, although we can use any time granularity required. Step 1 computes the new infections as a result of the SMS malware for an infected device. This implements a probabilistic model for infecting the device with SMS malware. Note that irrespective of how the infection occurs, the malware we consider is a hybrid that propagates infection in either medium. Step 2 and 3 are computed for each person in the PE. `computeVisits` looks at the schedule of the individual and based on the activity location and start time of the visit, sends a message to the PE containing the location where the activity takes place. This happens only if the activity starts at the current value of $t$. `sendVisits` sends the messages to the locations (on other PEs) where the activity occurs. Note that each PE is performing these steps in parallel and continues to send messages and receive messages from the broker. When the receiver retrieves the messages through `readMsgs`, it constructs events for the arrival and departure of the individual and adds them to the event queue. At the end all PEs synchronize before processing the events. Once every PE has the required individuals in the locations, we can process events and propagate the infection (`computeInfections`). Any call to `computeInfections` results in new infections and each new infection is propagated to the home PE of the individual for use in the future. Locally, an infected device updates the disease mani-

**Algorithm 1:** The EpiNet Algorithm.

```
for t ← 0; t < T; t ← (t + Δt) do
    foreach P_i ∈ P do
        //compute new infections from SMS malware
1       computeSMSInfections(P_i);
        markInfected(P_i);
        //send visits to location PEs
2       computeVisits(P_i, t);
3       sendVisits(PE);
        //process visit messages
4       Visits ← readMessages();
        //create events from the visits
5       makeEvents(Visits);
    //wait till all PEs obtain the data for instant t
    synchronize();
    //execute the events for each location
    foreach location l_j ∈ L do
        //compute the proximity malware infections
6       computeInfection();
7       sendOutcomes(PE);
    readMessages();
    synchronize();
    //update health state
    foreach i ∈ P_i do
8       updateState();
```

festation and creates disease update events for performing transitions through the manifestation. The `computeInfections` executes the events serially for each location in a PE. After completion of processing the events for all locations, we need to synchronize and wait for other PEs to complete this step before continuing with the next instant $(t + \Delta t)$. The infection messages conveyed to the individual's home PE update the state of the individual as *infected* by `updateState`.

### 3.4.2 Interventions in EpiNet

The responses implemented in EpiNet can be classified broadly as (i) adaptive/non-adaptive and (ii) local (individualized) and policy oriented.

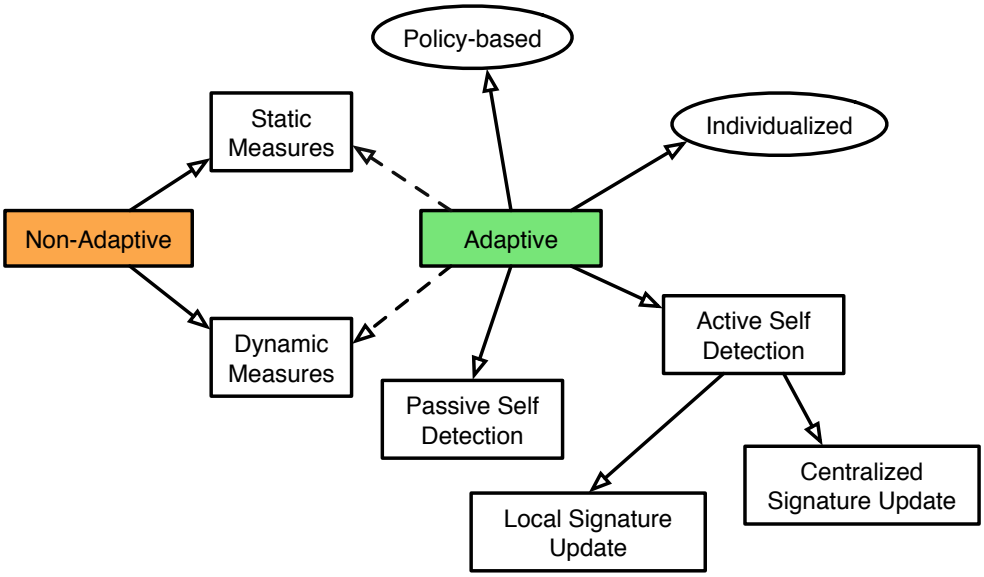



Figure 3.12: Response mechanisms implemented in the EpiNet framework. We have non-adaptive and adaptive responses that can use a combination of static and dynamics metrics. Dynamic metrics are determined through simulations. Policy-based and individualized responses are inherently adaptive and are implemented by the service provider and the individual, respectively.

#### 3.4.2.1 Adaptive and non-adaptive interventions

***Non-adaptive interventions*** occur before the start of the wireless epidemic and are a result of certain devices being patched for a vulnerability exploit that is released. These would be for non-zero-day malware where the fixes are available before the exploit starts to spread. If the device is not patched, then it becomes susceptible to the malware. The unrealistic assumption in such interventions is that people do not adapt to the malware and that devices that are not patched initially remain unpatched throughout the epidemic. Further, they are limited to studying interventions that are permanent, i.e., the patch completely eliminated the risk of getting infected once the device becomes exposed to the malware.

***Adaptive intervention*** strategies, on the other hand, incorporate changes in the movement of the people and include treatments that have both temporary effects (shutting the devices off for a brief period, disallowing the use of devices in certain locations, etc.) and permanent effects (patching a vulnerability). They are strategies that primarily change over the duration of the epidemic and undergo changes in the degree at which interventions are applied. For example, while an epidemic is in progress, the level at which the interventions are applied may depend on the efficacy of the previously employed strategy. If an earlier (early during the epidemic) strategy is found to be effective, it can be repeated, targeting other devices at varying stages of the epidemic. If a particular strategy is found ineffective, a different strategy can be tried. In the interventions we have implemented in EpiNet, we can differentiate various strategies by how frequently interventions are applied and triggering conditions are checked, i.e., the degree of adaptation.

At an implementation level, a non-adaptive intervention is specified as either changing the vulnerability of a device to contract the malware, or changing the behaviors of the intervened devices. In the former case, the intervention does not change the device-device interactions, so the disease may transmit through the contact network via the same edges. But the transmissions are hindered at the intervened nodes. This class of interventions include software patching, updating signatures for malware detection, or disabling communication sub-systems if anomalous activity is detected. A device that has been patched still goes to the same places and interacts with the same devices

as before. But now the malware cannot propagate via this device.

#### 3.4.2.2 Policy-oriented and individualized interventions

***Policy oriented interventions*** are patches or fixes that are propagated by the service provider or the application developer (through the software vendor or service provider) [53]. They include over-the-air (OTA) delivery of important updates from time to time without user monitoring. Individualized interventions, on the other hand, are the responsibility of the user. Depending on the user's awareness about the latest vulnerabilities, the user patches the device regularly or uses a malware detection software (either signature-based or signature-less) to guide patching. Some individualized interventions may be simple, but temporary–for example, switching off devices in case of malicious behavior, or disabling or uninstalling the vulnerable software.

The static interventions entail device selection based on graph metrics, and the selection of devices occur outside the simulation and can be configured offline. On the other hand, dynamic responses are implemented during the simulation. We currently implement three dynamic interventions: (a) Passive self-detection, (b) Self-detection with local update, and (c) self-detection with global update. All dynamic responses are dependent on self-detection. We define self-detection as the ability of the device to recognize self-infection through signature analysis or behavior recognition. Any device that performs this detection is *informed*. The self-detection with local update implements a mechanism in which an informed device can trigger any infected neighboring device into recognizing the infection and becoming *informed*. This is only local in the sense it is possible only in the neighborhood of an *informed* device. In the global update mechanism, a central server is informed of the malware through an *infection update* message, and the central server patches devices randomly when the number of infection update messages reaches a threshold.

## 3.5 Summary

In this chapter, we have outlined the parallel EpiNet simulation environment with the various modules that go into it. We discuss the construction of realistic networks – proximity networks for the study of Bluetooth and other proximity based technologies, and communication networks for the study of SMS/MMS based malware. We describe the modeling procedure for the probabilistic timed transition system to model the within-device malware protocol. We also provide information on the calibration experiments we conduct to calibrate the model. We use smaller networks in controlled settings to obtain parameters, such as probability of infection and duration a device remains infectious before entering the idle state. We then validate the model with small networks of 500 devices with network simulator ns-2 [76] and find that the model represents the malware propagation very accurately. We present the parallel simulator implemented to conduct the simulation studies using the network and the malware model.

The EpiNet framework is used to study malware propagation on a 30,000 device network in the Chicago area and we find that for a simulated time of 24 hours the runtime is around 45 hours on 5 processing elements. Although EpiNet was orders of magnitude faster than existing packet level simulation, it still lacks the power and the expressibility to be used for large designs on much larger networks consisting of millions of devices. Specifically, we need to address the following issues related to developing HPC-models for wireless epidemiology: (a) Optimizing inter-process communication and load-balancing for heterogeneous time varying short range communication networks is non-trivial. (b) Dynamic, independent actions of individuals results in changes in the network and intervention strategies which implies that non-adaptive schemes for efficient computation are not applicable. (c) There is a need for a large number of replicates to use simulation results in practical settings (searching for extremely large parameter space)

In the next chapter, we discuss details of EpiNet, a highly scalable implementation of the same environment with several approximations. We show that this new modeling environment can be used on large dynamic networks to study the impact of network structure and identify devices that are vulnerable or influential in the infection propagation. EpiNet can be used as a tool by analysts

and network planners to explore a larger parameter space to obtain insight into aspects of malware and their control for better network design and maintenance.

# Chapter 4

# Computational Aspects of EpiNet

The design goal of the EpiNet simulation environment is scalability to large dynamic mobile networks with the ability to specify and study sophisticated response mechanisms. This requires that the simulation environment scale to networks of 3-5 million devices. There are several challenges that the wireless epidemiology problem poses when implemented on HPC platforms and these need to be tackled in order to obtain scalability.

- Scale, heterogeneity, and time varying nature of short-range communication networks make optimizing inter-process communications and load balancing non-trivial.
- A highly dynamic nature of these networks as a result of changes resulting from individual's actions and interventions implies that non-adaptive schemes for efficient computations are usually not applicable. Moreover, it is important from the standpoint of human productivity to be able to represent these interventions easily.
- The need for a large number of replicates to use simulations in practical settings. The replicates are required to search an extremely large parameter space to derive results that are statistically sound.

In this chapter, we build on the discussion of EpiNet from Chapter 3. EpiNet can now be used to study and understand malware diffusion processes in realistic networks, to study the impact of the

network structure on the dynamics of malware spread, to identify individuals who are influential and vulnerable, and as a tool for analysts and network planners for building a more resilient infrastructure. The version of EpiNet presented in this chapter, is a computationally more efficient simulation engine to study malware propagation. It allows implementing and studying more advanced intervention strategies using dynamic graph metrics such as vulnerability and adaptive policy-based and individualized interventions. In the next few sections, we explain the enhancements made to EpiNet. We then describe some case studies that require and use these improvements. Later, we perform some experiments to analyze the scalability of this environment to represent and study much larger networks for more advanced studies and evaluations.

## 4.1 Enhancements to EpiNet

In this section we discuss in detail the issues that need to be addressed to make EpiNet scale to large networks and computationally intensive adaptive and non-adaptive response mechanisms.

### 4.1.1 Scalability of EpiNet

First and foremost, we improve the scalability of EpiNet; EpiNet can now study large dynamic networks with 3-5 million nodes; to the best of our knowledge, this is the *only modeling environment* for malware spread that scales to device networks with well over 100,000 nodes. These networks capture the human mobility in an urban region and the resulting time varying interactions of a digital device. As discussed earlier, the time varying nature and the lack of symmetry in such networks makes mapping on parallel machines non-trivial; in general, determining basic dynamical properties in such stochastic systems, such as whether the system will reach configurations with many infections, can be PSPACE-complete[1] [7]. As discussed in [92], network structure has a significant impact on the dynamics and the conclusions that can be drawn from such simulations – this makes scaling to large unstructured networks very important for practical use of these techniques. Additionally, we find that there are a number of parameters with high variance that

[1]Informally, this corresponds to the class of problems that can be solved in polynomial space.

require a large number of simulation runs for a particular study. In these cases, it is necessary to improve the overall time to complete the execution of such studies. From a HPC stand-point, this means that we need techniques that can allow scaling and effective use of a large number of processing elements (PEs) to conduct such extensive, high resolution simulations.

EpiNet achieves this scaling using a slightly lower resolution mobility and within-device malware model that enable both sequential and parallel performance improvements. In addition, EpiNet also employs a hybrid MPI-threads implementation for better utilization of multicore processing nodes. Table 4.1 illustrates the scaling results and compares existing simulation-based platforms. These ideas led to a 300 fold improvement in the overall performance for some networks, compared to other simulations. EpiNet also does well for studies with multiple simulation runs (see Table 4.2 for time taken to complete a study on a 100 node cluster).

### 4.1.2 Interventions in EpiNet

The second enhancement provides an expressive framework to specify and study sophisticated intervention strategies. An integrated modeling environment to support wireless epidemiology should allow policy makers and analysts an environment in which they can undertake various *what-if* studies; most modeling environments simply do not have this capability and users implement this in an ad-hoc manner. A novel aspect of EpiNet is that it has been designed specifically for analysts to be able to represent and study various dynamic counter-measures to control the spread of the malware. This requires two new capabilities, which pose further challenges for high performance computing: (i) One needs to measure the state of the simulation at regular intervals and perform situational assessment. (ii) The network and individual states have to be changed dynamically, as a result of behavioral changes, which have important consequences for the dynamics. All these changes cause significant dynamic message exchanges in the parallel framework and can potentially result in some slowdown depending on the actual intervention studied.

This chapter presents the changes we discuss above. First, we discuss the changes in the mobility and model resolution, the error introduced in the final infection size as a result. We incrementally

Table 4.1: Scalability comparison between different simulation environments. Comparison between ns-2 [76], and the two versions of EpiNet (Chapter 3 and Chapter 4, respectively) for different networks in terms of simulation time and error in results. ns-2 is compared with sequential executions of EpiNet on a single processing element (PE). Comparisons in column 4 are between the two versions of EpiNet on 30,000 devices. Note: Simulated time is 24 hrs. (column 5) and 4 hrs. (columns 3 and, 4).

| | | Devices–500<br>Simulated Time–4 hours<br>(Sequential) | Devices–30,000<br>Simulated Time–4 hours<br>(Parallel – 1 PE) | Devices–1.6 M<br>Simulated Time–**24 hours**<br>(Parallel – 20 PEs) |
|---|---|---|---|---|
| ns-2 | Runtime | **45-50 hours** | Not compared with parallel ns-2 | Not compared with parallel ns-2 |
| | Error | Reference (gold standard) | – | – |
| EpiNet | Runtime | **15 minutes** (**200X** improvement over ns-2) | **45 hours** | Not compared with EpiNet |
| | Error | **0% error** (Results matches with ns-2 on these networks) | Reference (ns-2 was not run for this comparison) | – |
| EpiNet | Runtime | **10 minutes** (300X improvement over ns-2) | **20 minutes** (135X improvement over EpiNet) | **1 hour** (executed on 20 PEs) |
| | Error | $<$ **0.1% error** (compared to ns-2) | $<$ **5% error** (compared to EpiNet, ns-2 not used for comparison) | No reference to compare with, ns-2 and EpiNet do not scale |

implement these approximations and then use a hybrid MPI-Threads implementation using Intel's Thread Building Block (TBB) library to use multi-core HPC architectures. We present the scaling results for EpiNet, strong scaling and weak scaling. We show that errors introduced by the approximations we choose are not significant enough to alter the conclusions and the error in final infection sizes are well below $< 5\%$. For example, a 4.5% error in the NRV1 network corresponds to a difference of 1,389 (1.7% of the total devices) infections. Note that this error is different for each network. For the Miami2 network (not reported here) the % error between 20 minutes and 1 sec mobility resolution is 1.7 (11,793 infections; 0.07% of the total devices). This improved scalability allows for timely execution of the simulation with the set of interventions we discussed above. EpiNet implements several of these interventions and one can sweep through the parameters and determine the most effective strategy to control the spread.

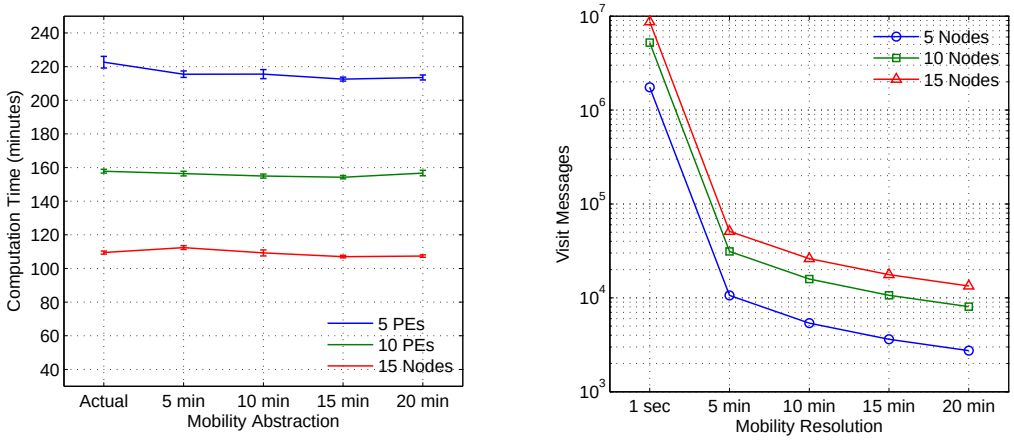
## 4.2 Scaling improvements for EpiNet

Computing dynamic measures on large, realistic urban networks motivates the scaling of EpiNet. EpiNet as discussed in Chapter 3 does not scale to large dynamic networks, and estimating dynamic metrics (as shown in Section 4.3.1) or executing large experiment design (as shown in Section 4.3.3) are not possible. We propose three approaches to obtain significant improvements in runtime for EpiNet: device-device interaction network approximation, approximations to the within-host malware representation, and system level improvements to make use of multi-core cluster environments. In this section, we look at each technique in isolation and measure the gain in implementing them and the resulting increase in error. We then combine all of them in the scaling studies we present in Section 4.4.

### 4.2.1 Approximating the host-to-host interaction network

The main goal of performing an approximation of host-to-host interaction network is to extract the maximum scaling without adversely impacting the results. The host-to-host interaction in EpiNet arises from human mobility patterns and the intra-location model of the wireless network.

Because work-load of the simulator is divided among the PEs, a set of locations and devices are allocated to each PE. When devices (or people) move from one location to another, the information of the device (i.e., current state of infection, the susceptibility of the device, etc.) is transmitted to the new location. The key idea here is to reduce the volume of messages by reducing the frequency of these updates by altering the devices' mobility. The obvious side-effect is an alteration of the device-device interaction network leading to error in propagation estimates. In this section, we evaluate the effect of the change in update interval of the mobility patterns of individuals. Specifically, we measure the change in the computation and communication overheads resulting from this approximation. We also measure the error in the instantaneous and final infection sizes and the growth of the infection.




(a) (b)

Figure 4.1: Advantages of abstracting the mobility of individuals. Scalability obtained by changing the mobility update interval from 1 second to 20 minutes. **(a):** Computation time changes as a result of changes in the mobility resolution. No improvement is observed in the total compute time; **(b):** Shows the message volume resulting due to change in the update interval. The activity start and durations are rounded to the interval boundaries so that update occurs in that frequency. We obtain a 100X reduction in message volume when mobility interval is increased to 5 minutes.

Figure 4.1b and Figure 4.4 show the results obtained by changing the mobility update interval from 1 second to 20 minutes. We measure the change in computation, communication overhead in addition to the instantaneous % error in the propagation. Computation time is not affected by changing the resolution of the mobility (as shown in Figure 4.1a). Nevertheless, the communication overhead is significantly lower. Figure 4.1b shows the reduction in the total number of mobility messages exchanged between the PEs as the resolution is changed from 1 sec. to 5 minutes, and so on. In Figure 4.1b, the update intervals are indicated along the x-axis – '1 sec' indicates that update occurs every second as provided by the activity-based mobility data. '5 min,' '10 min,' etc. indicate the rounded time intervals for the activity start and end times. We observe a dramatic reduction in message volume when mobility update interval is increased to 5 minutes and the total number of messages reduces by 100X. The change is not dramatic for higher update intervals. The reduction in message volume will be higher when more PEs or nodes are used for the simulation.

It is important to note that we plot the actual number of messages that are being exchanged and not the message sizes. In order to identify that a certain node has no more messages to send, we use a 0 byte 'end message.' The message volume we plot includes these end messages as they incur the MPI communication overhead irrespective of the actual message sizes. For the case when update interval is according to the actual activity, we have to exchange messages every second, and if there are no changes in the mobility, for $n$ PEs, $(n-1)$ end messages are sent by each PE. This is the reason for the message complexity in using an 1 sec. update interval. Thus, increasing the update interval clearly leads to better communication performance although the scale of the improvements does not extend to higher update intervals.

**Key Findings:** Mobility resolution leads to a 100X reduction in the number of messages exchanged. The update interval of 5 minutes contributes less than 1% error in both instantaneous and final infection sizes. We use this mobility resolution in all the simulations in later sections, unless stated otherwise.

### 4.2.2 Approximating the Within-host model

The Bluetooth malware model we proposed in Section 3.3.1 is shown in Figure 3.8. The model is based on the $1^{st}$ generation Bluetooth malware Cabir [29] and CommWarrior [57]. The *infectious* state consists of several sub-states arranged in branches with probability of transition into a particular branch. Each branch depicts a possible path an infected node takes and determines the infection probability and the duration of the infectiousness. Every state in these branches has a corresponding probability of infection $p_i$. The parameters in the model are calibrated from simulation studies (on a network with 500 devices) conducted using Bluetooth protocol stack [93] and our malware protocol implementation in ns-2. Due to lack of real data on malware outbreaks and spreading patterns of mobile worms, we use ns-2 studies as the gold standard. The detailed model has been validated with ns-2 in the same settings for smaller network sizes and found to be very accurate. More details of the calibration and validation studies is presented in Section 3.3.

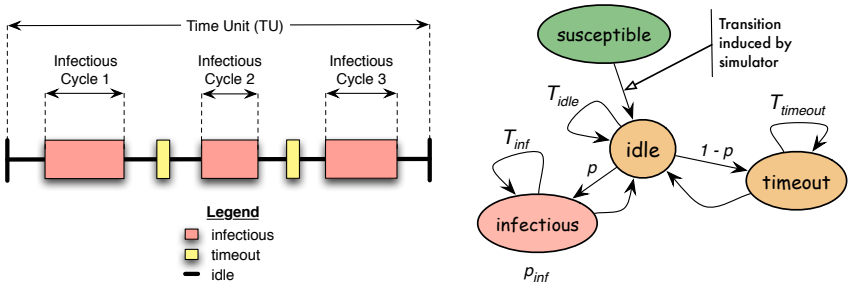
In this section, we are interested in abstracting the model further so that large networks can be studied. In order to reduce the complexity of the model, we perform two kinds of approximations:

1. *State-space compaction by collapsing states (or $M_{SC}$).* Here, we perform an abstraction in which multiple sub-states in *infectious* state of Figure 3.8 are combined into one. Because each sub-state represents a probability of infection ($p_i$) at every second, this probability is re-calculated after compacting the states.

2. *State elimination using Gillespie's algorithm (or $M_{SE}$).* We use Gillespie's algorithm to perform this abstraction, resulting in the *simplified model* shown in Figure 4.2b. As a result of this abstraction, the *infectious* state is represented as a single state. We need to derive both $p_{inf}$ (note that $p_{inf}$ in the reduced model is different from the $p_i$ of each sub-state) and time the device spends in the *infectious* state, $T_{inf}$.

As we will show, $M_{SE}$ results in better gains both in terms of computation and communication time and helps us achieve the level of scalability required to study a network with a million devices. We also observe the % error contributions of these model approximation approaches. Note that the

model reduction in these approaches are being performed in addition to the mobility resolution of 5 minutes.




(a) (b)

Figure 4.2: State elimination using Gillespie's Algorithm. Applying $M_{SE}$ on the original model through Gillespie's algorithm. **(a):** Discrete-time simulation of the detailed malware model. Shows the stages an infected Bluetooth devices goes through to infect other susceptible devices unrolled for a time unit (TU); **(b):** Simplified model after applying Gillespie's algorithm.

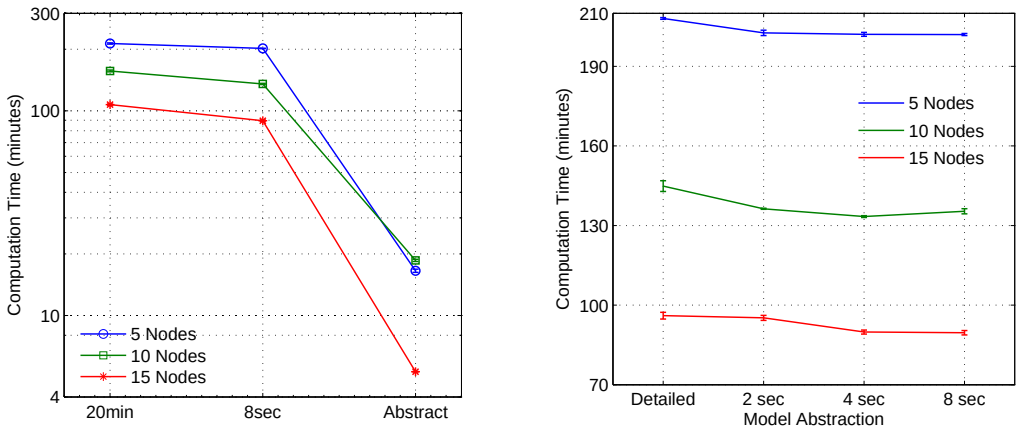
#### 4.2.2.1 State space compaction by collapsing states

The sub-states in the *infectious* state is derived from a histogram of the probability of infection in the calibration experiments (see Section 3.3.2 for details on calibration). The state-space compaction is performed by pre-processing this histogram and increasing the bin width. We use bin width of 2 $s$, 4 $s$, and 8 $s$ to reduce the number of sub-states in the *infectious* state. The results obtained from $M_{SC}$ on the *infectious* state are shown in Figure 4.3.

We do not show the computation time improvements as $M_{SC}$ achieves only modest reductions. This indicates that simple state compaction does not provide scaling.

#### 4.2.2.2 State elimination using Gillespie's algorithm

The model in Figure 3.8 requires the simulation to proceed in time steps of 1 $s$. Such a situation arises in stochastic simulations, where repeated monte-carlo sampling slows the simulation down.




(a) (b)

Figure 4.3: Computational improvements and error by state elimination using $M_{SE}$. **(a):** Computation scaling obtained in $M_{SE}$ compared with $M_{SC}$ and mobility resolution. Note that $M_{SE}$ and $M_{SC}$ include the mobility resolution at 5 minute intervals already enabled. The comparison is between $M_{SE}$, $M_{SC}$ with state compaction for 8 seconds, and mobility resolution of 20 minutes; **(b):** Scaling in the computation time by changing model resolution through $M_{SC}$. No significant gains are observed in compute time.

The approach by Gillespie [39] uses a technique for determining the time interval before the "next event," thereby skipping redundant sampling. We use a similar approach and reduce the granularity of the simulation to achieve speedup and call it $M_{SE}$. This could introduce errors, and we need to construct a simplified model to minimize their impact.

The Bluetooth malware approximation is achieved by $M_{SE}$with a certain *time unit* (TU). For example, consider 1 $TU = 10\ s$. This means that the simulation progresses in 10 second increments and no events occur within this interval. An offline simulation of the state space determines the state traversals for this TU; instead of considering single state transitions in Figure 3.8, we consider sequences of transitions that correspond to a duration of a TU. As shown in Figure 4.2a, a large number of such sequences (we use 10,000 trials) can be used to determine the average probability of infection, $p_{inf}$, and infectious duration, $T_{inf}$. In addition, we also determine the probability of traversing to the *timeout* state, $p$. A low resolution model obtained as a result of this procedure is shown in Figure 4.2b. Performing simulation in terms of TUs provides an improvement in the computation times and makes the study of large, million device networks. As Figure 4.3a shows we obtain a 10X improvement in computation time over the detailed model. We now evaluate the accuracy of such an approximation in terms of instantaneous and final infection sizes.

In the above sections, we have verified the cumulative and time series growth of the infection and computed the error introduced by $M_{SE}$. What about the effect of the $M_{SE}$ at individual locations? Next, we determine its effect of $M_{SE}$ on the spatio-temporal impact on the infections, i.e., how the model behaves spatially. Here, we look at the individual locations and determine the number of infected devices at different times – 8AM, 12 Noon, 4PM, and end of simulation – and compare the detailed model to model after $M_{SE}$. Figure 4.5b shows the difference between the models. The x-axis represents the number of infections in the detailed model and the y-axis represents infections after $M_{SE}$. Other than a few locations where the infection numbers do not match, this number is very small in comparison to the number of locations. Further, the infections at these locations also match temporally, showing that $M_{SE}$ maintains the infection count. Note that the approximate model in some cases predicts higher infections in some locations indicated by points lying on the y-axis in Figure 4.5b.

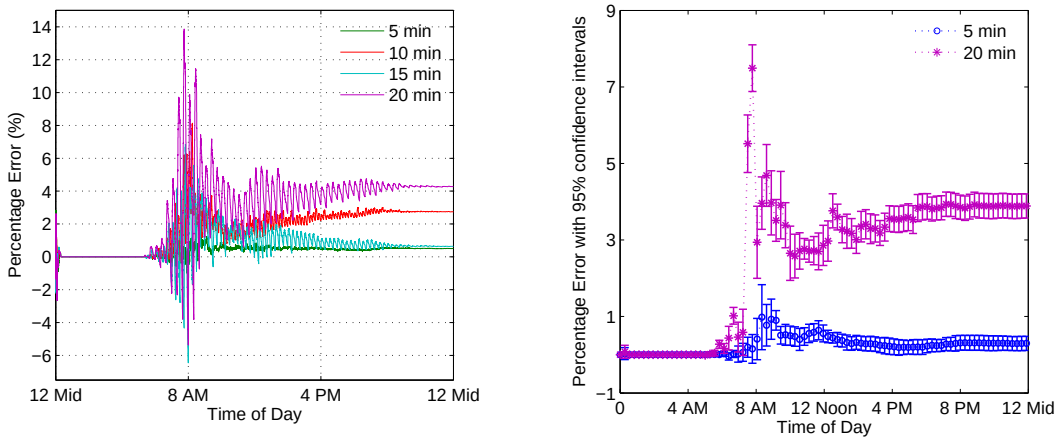
### 4.2.3 Error Measurements for the approximations

We will now evaluate the above approaches of approximating the malware model by computing the error in infection propagation. Error can be measured using various outcomes of the spread. In this chapter, we use different methods to compute the errors introduced by the approximations to the malware modeling approach. Each of these metrics displays different aspects and provides better insight into the outcome predicted by the EpiNet simulation environment. In this section, we first look at the errors introduced by decreasing the mobility resolution. Next, we look at the error introduced due to the model approximation. The following are the metrics we compute to determine the error introduced in the modeling approximations we discuss in this chapter:

1. final infections at the end of the simulated time,
2. instantaneous infections at each time step,
3. time to infection for each infected device, and
4. vulnerability of the infected devices

Increasing the update interval for device mobility alters the network created at activity locations. For example, by changing update interval to 5 minutes, the devices are moved once in 5 minutes. This alters the network by extending certain links at the source location while reducing the link durations at the destination location. As Figure 4.4 shows the error introduced measured in terms of the final infection sizes as a result of the lower mobility resolution. The percentage error increases as the update interval is increased except for the mobility resolution of 15 minutes. The error for 15 minute mobility resolution is lower than 10 and 20 minute resolutions because of the network created. It shows that the instantaneous and final infection is closer to the resolution of 1 second than 10 or 20 minute mobility resolution.

Figure 4.4 shows that the error is higher at instances where the propagation is faster and mostly during the initial phase of the spread. As simulation progresses, the error – measured with respect to the number of infections – introduced by the mobility resolution reduces. From Figure 4.1b, we

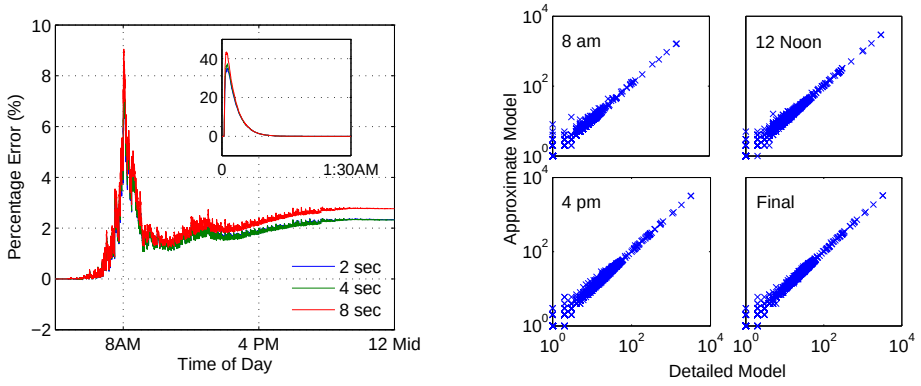



(a) (b)

Figure 4.4: Error in instantaneous infections due to lower mobility resolution. We compare the error in instantaneous infections when mobility resolution is reduced. **(a):** % error in the instantaneous infections (measured every second) for the duration of a day when mobility resolution is reduced. The error is lowest for 5 minute resolution and increases as this is reduced. The variation for instantaneous infections is higher during the initial phase of the spread and finally reduces to less than 5% in comparison to 1 second mobility resolution; **(b):** Percentage error in instantaneous infections with 5 and 20 minute mobility resolution with 95% confidence intervals. We show that the variation of this error is not more than 0.5% of the average error even when the initially infected devices are changed.

see that the advantage gained in communication times by using increased mobility resolution does not extend beyond the 5 minute interval, and the gains are marginal. Further, the approximations employed do depend on the importance of accuracy. For example, if we require a very accurate simulation result, we have to employ the highest resolution of updates for mobility. When the studies are not that dependent on the actual number of infected devices but the trend in the spread or an approximate percentage of devices infected, then lower mobility resolution can be employed to obtain faster results with low errors. In either case, it appears that the resolution of 5 minutes performs extremely well both in terms of reduced communication and low error. As a rule of thumb, if the error can significantly change the conclusions made with a study, then we use the update interval that produces the least error. For example, the error in final infection for the 20 minute mobility updates is 4.5%. The error in terms of actual difference in the infection count for NRV1 network is 1,389 infections higher than with 1 second mobility resolution. This corresponds to 1.7% of the total nodes in NRV1 network. Note that this error is different for different networks. For the Miami2 network (not reported here) the % error between 20 minute and 1 sec mobility resolution is 1.7 (11,793 infections and 0.07% of the total devices).

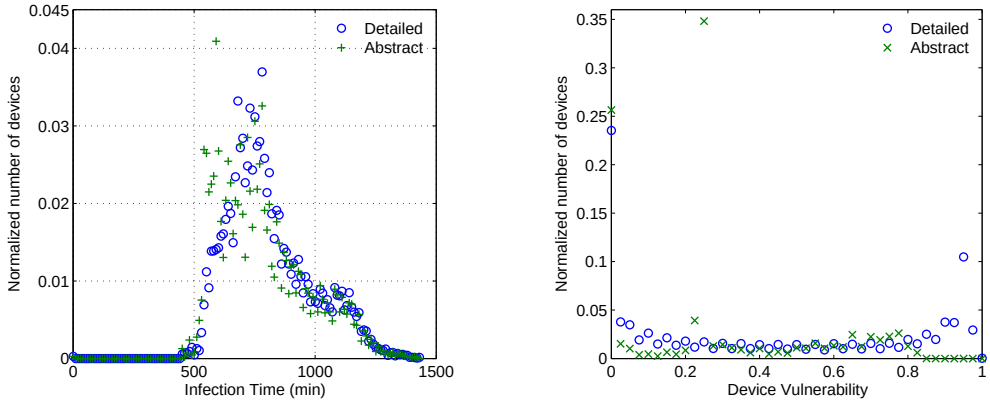
Figure 4.5 shows the error obtained by enabling the modeling approximations in addition to the mobility approximations while studying the malware spread. Here we look at two different aspects of the spread patterns and their effect on the instantaneous infection size and the spatial spread patterns. The instantaneous infections are measured as a global measure of total infections across all locations measured every second. This is shown in Figure 4.5a. The % error in the instantaneous infections is shown in Figure 4.5a and find that is somewhat higher than for the original model. However, the % error in the final infection sizes of the two is comparable. This shows that the error introduced by employing an approximate model is not very significant and can yield significant scaling in the ability of the simulator to study much larger networks significantly faster. For the spatial spread patterns we look at all the locations in the simulation scenario and consider each location while comparing the temporal infection counts at each location. We plot the actual infection counts we observe in Figure 4.5b. All points that lie along the 45° diagonal show that the number of infections at each location is the same whether the approximate or the detailed model




(a) (b)

Figure 4.5: Error due to state elimination using $M_{SE}$. Error in the final infection sizes as a result of state elimination using $M_{SE}$ measured as a percentage of the devices for the NRV1 network. **(a):** Sensitivity to model parameters increases % error in instantaneous infections when model resolution is decreased. The inset shows the % error during the first few minutes of the epidemic. Error in final infection numbers are low enough not to change the conclusions; **(b):** Comparison of infections at a spatial level within locations at different instances during the spread of the malware. Here we compare the approximate model with the detail model using the cumulative infections at different instants (8 AM, 12 Noon, 4 PM, and 12 Midnight).

is used to study the malware. In this case, although we do not observe such a dramatic result, we observe that the approximate model is accurate in predicting the number of infections in most locations. This holds well when we look at the temporal infection spread across the locations.




(a) (b)

Figure 4.6: Time to infect and device vulnerability due to model and mobility approximations. Comparison of the time to infection and the device vulnerability for the NRV1 network between the detailed approach and mobility and model approximations. **(a):** Distribution of infection time to compare the abstract and detailed modeling approaches. In the abstract model infection times of devices have moved to the left, indicating earlier infections than with the detailed model. Nevertheless, the overall nature of infection times remain unaltered; **(b):** Device vulnerability distribution comparison between detailed and abstract models with 100 trials. There are some outliers when the abstract model is used as shown where a significant number of devices have infection likelihood of 0.2. Most of the devices with high vulnerability for the detailed model are not shown to be highly vulnerable.

In addition to measuring the difference in the final infection sizes between the detailed and the approximate models, we measure two quantities: (1) the infection time distribution, and (2) the device vulnerability distribution. The infection time distribution shows the distribution of the time when the devices become infected starting with the same set of initially infected devices. This provides another dimension to the analysis of the effect of modeling and network approximations we perform to improve the scalability. Device vulnerability indicates the probability that a device

becomes infected irrespective of the initially infected devices and indicates how the modeling and network approximations affect this metric. In this section, we compare the detailed and the approximate methods by looking at these metrics. The detailed approach is with the mobility resolution of 1 second and the original malware model (presented in Figure 3.8, Chapter 3). The approximate approach is the one with both the reduced mobility resolution and state elimination using $M_{SE}$.

Figure 4.6 shows the results for the infection time and vulnerability distributions measured for the NRV1 network. Figure 4.6a shows the infection time distribution and compares the two approaches. It shows that the two approaches are very close and the peaks have almost the same values, but the infection time for the abstract model are slightly moved to the left, indicating that the maximum number of devices were infected earlier than the detailed model. It shows that when the approximations are enabled, the devices get infected earlier and more infections occur earlier. There are fewer samples for the approximate model as the simulation interval is increased to 30 seconds but the results from the approximations do not significantly alter the infection times. Figure 4.6b shows the distribution of the vulnerability obtained as a result of the approximations and compares it to the detailed approach for the NRV1 network. The comparison indicates that there is a significant change in the vulnerability distribution between the two models. Vulnerability is a metric that has significant variability, especially in a scenario where there are a large number of devices and the initial conditions (as to the initially infected devices) are changed. Determining this metric with sufficient accuracy for a network with millions of devices requires a large number of trials. This is the reason for the large variation in the two approaches, and more trials would provide a better picture on the actual difference.

**Key Findings:** Of the two model approximation techniques, $M_{SC}$ and $M_{SE}$, $M_{SE}$ provides 10X improvement in computation time and contributes 2-3% error in the final infection size. Out of the metrics we used to compare the two approaches, the vulnerability distributions show significant differences. The infection time distributions compares very well with either approach. In this section we have used the model approximation techniques in addition to reducing the mobility resolution. In later sections, we use EpiNet with both these approximations in addition to the system-level implementation optimized further for multi-core clusters.

**Algorithm 2:** EpiNet Algorithm with Intel TBB

**Input/Output:** (*Same as sequential algorithm*);

```
1  task(l);
   foreach Infected device u ∈ D_l do
       for n ∈ N_u do
2          if n ∈ S then
               i --infects, p_i--> n;
3              Store new infections;
   if reader then
4      Read data from disk;
       Send to home PEs of devices and locations;
   else
       Recv allocated devices and locations;
   for t = 0 to T do
5      Set initial states of devices;
       foreach d ∈ D_PE do
           if activity exists then
6              Send device to location;
7      Receive devices from other PEs;
8      Synchronize PEs;
9      parallel_for(task(l), L_PE);
       for l ∈ L_PE do
           Send infection notification messages;
           Receive notifications;
10     Synchronize PEs;
```

### 4.2.4 System-level optimizations using hybrid MPI-Threads

In today's high performance computing environments, hierarchical hardware designs are employed – shared memory nodes with several multi-core CPUs. So, it is important to be able to take advantage of this and explore options to implement a hybrid approach with a single MPI process per node and multiple threads performing specific tasks. We employ the Intel®Threading Building Blocks (TBB) for creating the multi-threaded implementation of the MPI-based parallel implementation of EpiNet.

We use a simple form of loop parallelism through the `parallel_for` construct provided in TBB. Algorithm 2 shows the pseudo-code for EpiNet using TBB libraries. The task for each thread is infection computation (Line 1). We alter the implementation to optimize the infection computation part of the algorithm, which accounts for 50% of the total simulation runtime. Further, because the infection computations at locations are independent (devices do not simultaneously exist in multiple locations), we can process them in parallel. We implement the infection computation in the original parallel implementation of EpiNet in parallel. The *task* is defined as shown here. TBB uses the grainsize parameter to determine the number of tasks allocated to each thread. We design each task to consist of processing a set of locations in $L_{PE}$ grainsize at a time. Aspects of notifying the new infections and receiving infection notifications are moved outside the loop.

In the rest of this chapter, we refer to this implementation as EpiNet-TBB. Section 4.4.3 describes the experiments we perform and analyzes the results.

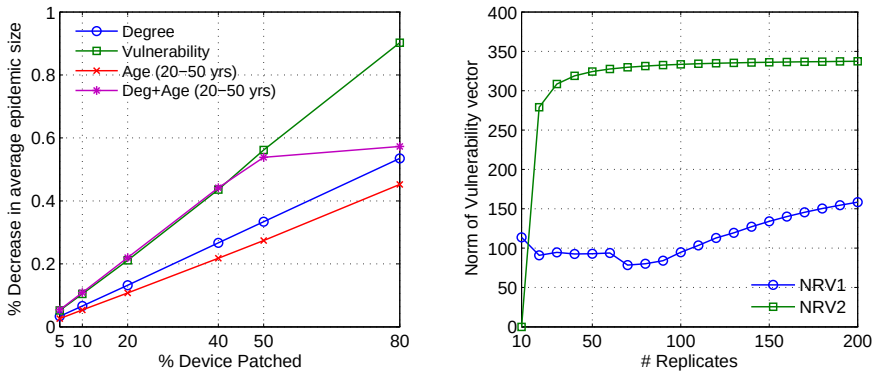
## 4.3 Illustrative Case Studies using EpiNet

In this section, we use the EpiNet simulator to study the propagation of mobile malware on large networks. We study the NRV and Miami networks with the people in two separate demographics – one consists of people in the age-group of 20-50 years (NRV1 and Miami1) and another with people in the age-group of 13-80 (NRV2 and Miami2). Figure 4.7 shows the configuration of these studies and we use the EpiNet simulator instead of EpiNet.

1. **Networks:** (# Devices; # Locations)
   (a) NRV1: 77,659; 30,896
   (b) NRV2: 126,800; 39,766
   (c) Miami1: 1,269,650; 448,453
   (d) Miami2: 1,642,565; 465,267
2. **Initial infections:** 1%
3. $T_{idle}$: 20 $s$; $T_{timeout}$: 12.80 $s$, $N_{resp}$: 4
4. **Infection seed time:** 12:00 Midnight
5. **Simulated time:** 24 Hours from 12 Midnight to 11:59 PM next day
6. **Simulators used:** EpiNet
7. **Seeds:** 5 (for each combination of input parameters). All plots show 95% confidence intervals.
8. **Computing resources:** SGI Cluster with 96 nodes, 3GHz Xeon processor with 8 cores/node and 16GB RAM
9. **# PEs:** NRV1/NRV2: 5-30; Miami1/Miami2: 15-40 nodes

Figure 4.7: Experimental configurations and parameters.

We illustrate EpiNet via three illustrative studies. The first addresses the problem of finding the best set of devices to patch in order to control the malware spread. We study an important dynamical measure, *vulnerability* (defined below), and show that interventions based on this measure are much more effective than interventions based on simple structural measures. Vulnerability is a dynamical measure and require a large number of monte-carlo samples to get reliable estimates, and we use EpiNet to achieve this. In the second study, we set up a factorial design to study the efficacy of interventions as a function of spatial and demographic heterogeneity of the mobile network. Our results show interesting differences between a rural setting and two urban regions that have different demographics. The result highlights how such environments can be used in practice and the importance of using realistic mobile networks.




(a) (b)

Figure 4.8: Vulnerability estimation on large dynamic networks. **Case Study 1:** Vulnerability as an effective metric to select devices to patch. **(a):** Plots decrease in average epidemic size when the % devices patched in increased. The devices are selected based on degree, vulnerability, owner belonging to the 20–50 age group, and age and degree combined. Vulnerability provides the upper bound for efficacy; **(b):** The number of replicates required to stabilize vulnerability estimation on NRV1 and NRV2 networks; **(c):** Cumulative fraction of devices' vulnerability as the number of replicates is increased for NRV1. There is a high variability in vulnerability and a large number of replicates is necessary to correctly estimate it.

### 4.3.1 Case Study 1: Computing dynamic measures in networks

The case study illustrates how simulations such as EpiNet can be used to solve the *optimal control problem (OCP)*: Given an instance of the wireless epidemic problem and a budget $B$ on the total number of devices that can be patched (e.g., due to time, bandwidth, or airtime restrictions), find the optimal set of devices to apply patch so as to reduce the final attack size. The general problem is computationally hard [28], and this motivates the use of simulation based methods.

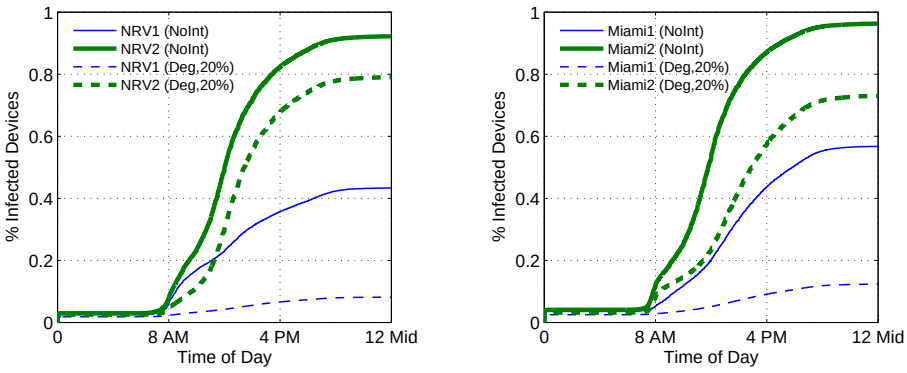
Here we concentrate only on non-adaptive methods. We compare the selection of nodes based on static structural measures (e.g., degree, betweenness, age) with a new dynamical measure we call *vulnerability*; the vulnerability of node $v$ (denoted by $V(v)$) is defined as the probability that it gets infected (under specific initial conditions) – this is a *dynamical measure*, which depends on both the temporal network and the dynamics. As shown in Figure 4.8, patching nodes in order of their vulnerability outperforms other strategies for the OCP problem.

### 4.3.2 Case Study 2: Effect of device penetration

Our second study is motivated by recent work by Wang et al. [94]. We consider how adoption of digital devices by progressively younger children is likely to affect the spread of malware and interventions to control it. We consider proximity Bluetooth networks for the New River Valley region in Virginia with two widely different populations – **NRV1** (with 77,600 devices carried by people in the age group 20–50 years) and **NRV2** (with 126,800 devices carried by people in the age group 13–80).

We study the propagation of mobile malware on these networks and the impact of one counter-measure based on degree. Figure 4.9a shows the spread of the infection in the two networks in terms of percentage of devices infected (y-axis) as the day progresses (x-axis). The solid lines indicate the spread of the malware without counter-measures. We can make two observations from this: (i) We see significant differences in the propagation dynamics between these two networks, which can be explained by the network characteristics – NRV1 is not completely connected and has

138 components while NRV2 is much better connected with only 13 components (because of the greater device penetration). (ii) The degree-based counter-measure shows different efficacy in these two networks (as shown in Figure 4.9a) – with 20% of devices patched, there is a 75% reduction in the number of infections in NRV1 but only 12% in NRV2. Thus, the nature of spread and the effectiveness of counter-measures are a function of the network and result in one network not always being translated to other networks.




(a) (b)

Figure 4.9: Targeted patch application results in different outcomes. **Case Study 2:** Targeted patch application on different networks. **(a):** % of devices infected with and without targeted device patching for the NRV networks with different device ownership models. Patching 20% achieves 80% (NRV1) and 10% (NRV2) reduction in final infection sizes; **(b):** Interventions for the Miami1 network based on degree of the devices.

### 4.3.3 Case Study 3: Effect of Spatial and Demographic Heterogeneity

This study illustrates the importance of fast simulations and the need for realistic mobile networks for formulating policies as they pertain to controlling malware spread; note that the focus is only on illustrating the use of environments like EpiNet, rather than the specific conclusions. We consider a spatial individualized dynamic intervention scheme where individuals proactively obtain updates when malware prevalence in their neighborhood reaches a certain threshold. They may be alerted by a device-based detection scheme such as [103]. The individuals that actually obtain updates

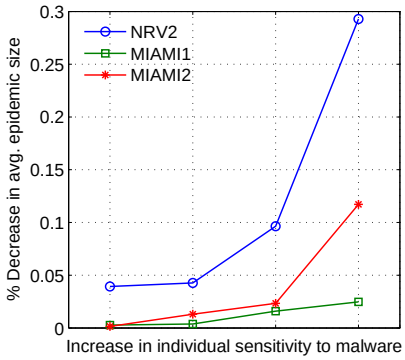



Figure 4.10: Individualized dynamic intervention based on malware prevalence. **Case Study 3:** Effect of individualized dynamic interventions with heightened individual's sensitivity to malware. High sensitivity decreases infection size by only 30%, 10%, and 5% in NRV2, Miami1, and Miami2 networks, respectively, when only the 20-50 age group applies patches.

belong to the 20-50 year old demographic. We use a factorial design with 2 factors – sensitivity of individuals ($s$) and the likely-hood ($l$) that they apply the patch with 4 levels for each factor ($s = 10\%, 30\%, 50\%, 70\%$ and $l = 0.1, 0.3, 0.5, 0.7$). Figure 4.10 shows the % decrease in infection size as the sensitivity of the individual increases. Even with a very high level of sensitivity, a 10% infection rate in the neighborhood triggers an individual to obtain a patch.

Table 4.2: Runtime for a 2-factor, 4 level, 5 replicate individualized intervention study. Study and individual runtime in minutes for the networks used in Case Study 3. The study is a factorial design with 2 factors, 4 levels, and 5 replicates conducted on a 100 node cluster. Runtime shows execution time without and with interventions (in parentheses) in minutes.

| **Networks** | **Nodes** | **Runtime (min.)** | **Study Runtime (hrs)** |
|---|---|---|---|
| NRV2 | 5 | 22 (22) | 1.5 |
| Miami1 | 15 | 86 (85) | 5.5 |
| Miami2 | 15 | 135 (131) | 33.75 |

These dynamic interventions require computations to keep track of individual neighborhoods and messaging to update global variables. Studying such interventions requires the framework scale to

efficiently manage these schemes for large networks. We determine the runtime for the complete study for each network and the individual runs and are reported in Table 4.2. The case study for the Miami2 network takes 35 hours of wall clock time on a cluster of size 100 nodes.

## 4.4 Performance Evaluation

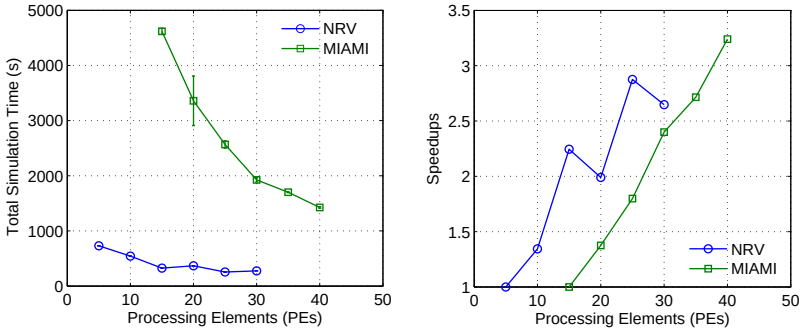
In this section, we evaluate the scaling improvements described previously. The cluster we use is a SGI ICE with 96 nodes, each consisting of two quad-core 3 GHz Xeon processors with 16 GB of RAM. The system uses InfiniBand interconnections. The networks we consider for this study include the NRV1 and Miami1 networks. Figure 4.7 shows the details of these networks and the other parameters used for the study. We model a basic infection spread scenario using the approximate model similar to that described in first case study (Section 4.3.1).

We evaluate the following areas: strong scaling (constant problem size with an increasing number of PEs), weak scaling (scaling problem size and number of PEs proportionally), and effects on performance by varying the ratio of number of MPI processes to the number of cores on multicore PEs and the scaling improvements of the hybrid MPI-threads implementation. Finally, we evaluate EpiNet's basic partitioning scheme against a static, graph-partitioning based approach.

With our proposed enhancements, we are successfully able to simulate a network with 1.2 million devices for a 24 hour simulation time with a runtime of 30 min with 40 PEs. This is a significant improvement over a previous implementation which could simulate ≈60000 device network in the same time - a 20X improvement. The following sub-sections elaborate on each of our results.

### 4.4.1 Scaling Behavior of EpiNet

Figure 4.11 illustrates the **strong scaling** using the approximate model for a fixed network size and structure. We observe that the total simulation time scales well as more PEs are added. The reasons for this include (1) reduced memory footprint within each PE because it needs to maintain only a subset of devices and locations where interactions take place, and (2) independent, parallel




(a) (b)

Figure 4.11: Strong Scaling of EpiNet for NRV and Miami networks. **(a):** Total simulation time as the number of PEs are increased; **(b):** Relative speedup with respect to the lowest PE number, 5 for NRV1 and 15 for Miami1.

processing of the locations at each PE. We use the runtime with 5 nodes as a reference for the NRV model and with 15 nodes for the Miami model to compute the scaling as number of PEs are increased.

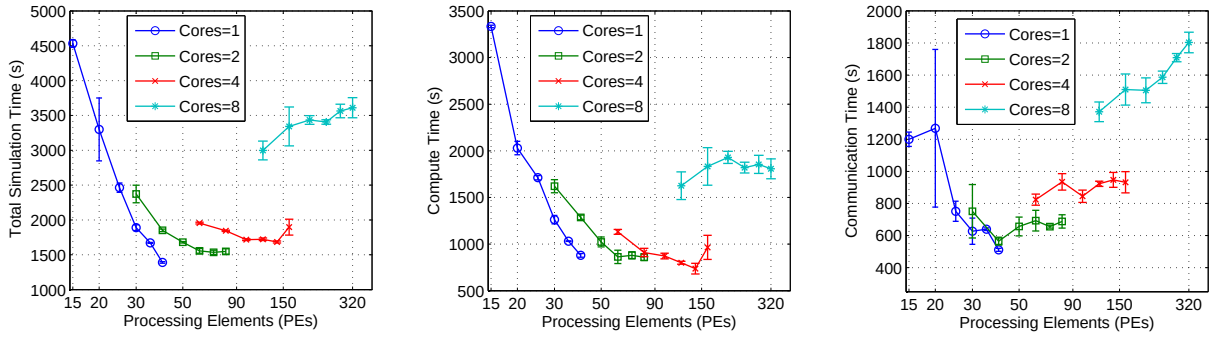
Table 4.3 shows the **weak scaling** results, which indicate the scalability of the approximate model with problem size. The Miami1 network is roughly 16 times larger than NRV1. The runtime for perfect weak scaling is shown in row 6. The number within parentheses shows the scaling factor to the observed runtime (row 4). As we increase the problem size and the number of PEs, we see that simulation time scales accordingly. This confirms that EpiNet has good weak scaling properties and can be used to study larger networks simply by adding more PEs.

### 4.4.2 Scaling MPI processes on multiple cores of same node

In this sub-section, we evaluate the impact of multicore architecture on the scaling behavior of EpiNet. Figure 4.12 shows the effects on runtime when using various numbers of MPI processes spawned on each node using the Miami1 network (note that the x-axis in these plots is the number of PEs). We use the default OS scheduling algorithm and do not modify the affinity of any MPI process.

Table 4.3: Runtime under weak scaling for EpiNet. Row 2 and 3 indicates the approximate size of the network and the number of PEs used in simulations. Rows 4 and 5 displays simulation runtime (in seconds) and the average devices per PE. Row 6 shows the normalized runtime and the factor by which the weak scaling is off from perfect weak scaling. The columns shows the values for the NRV1 and Miami1 networks used in the weak scaling study.

| **Network** | **NRV1** | **Miami1** | |
|---|---|---|---|
| **Size** | 1 | 16× | |
| **PEs** | 5 | 80 | 96 |
| **Runtime(s)** | 730.35 ±0.65 | 944.95 ±5.16 | 820.36 ±4.23 |
| **Average devices/node** | 15532 | 15871 | 13226 |
| **Norm. (Factor)** | – | 746.29 (1.26×) | 621.91 (1.32×) |




(a) (b) (c)

Figure 4.12: Scaling studies for MPI processes on multiple cores for the Miami1 network. **(a):** Total simulation time when using 1 core and multiple cores on each node (2, 4, and 8 cores), respectively. Using more PEs on the same network does not result in runtime improvements when more than 2 cores/nodes are used; **(b):** Compute times comparison when the number of PEs and the cores/PEs are increased for the Miami1 network. Compute time does not improve when more than 80 PEs are used; **(c):** The communication time when multiple cores are considered. Communication overhead starts to increase when more than 80 PEs are used.

We observe that adding more MPI processes per node and utilizing the available cores provides diminishing returns as more and more PEs are used. Specifically, runtime improved proportionally to the number of nodes for the 2 ('Cores=2') and 4 ('Cores=4') MPI processes per node cases, but only by about 33% for 8 ('Cores=8') MPI processes per node case. On a single node, the possible reasons for the slowdown include cache poisoning, network contention, memory contention, and lack of core affinity. More analysis of the execution characteristics of EpiNet is required to determine the contribution of each of these candidates. Across multiple nodes, as more PEs and MPI processes per PE are added, communication costs eventually start dominating. Runtime scaling stops or reverses beyond about 35-40 nodes due to communication costs of this particular network model. One way of amortizing the performance degradation due to multiple MPI processes on a single node is to employ a hybrid MPI-threads implementation, which can potentially improve per-node utilization. These results are presented next.

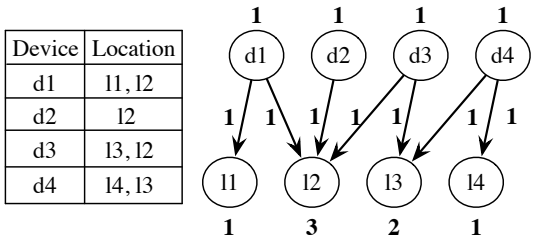
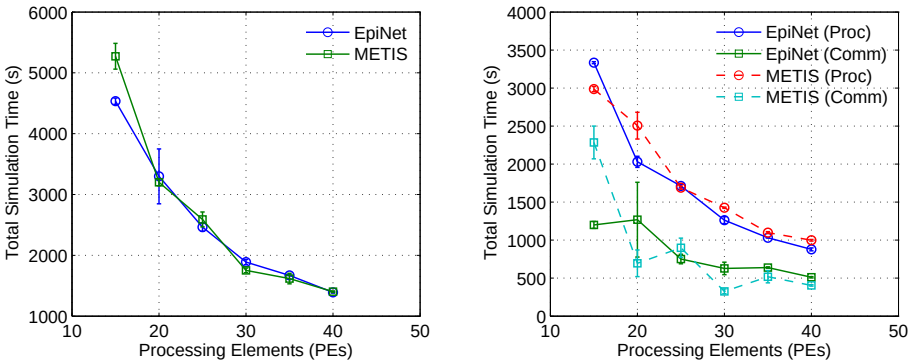



Figure 4.13: Activity information converted to device–location network for METIS. Converting the activity information into device – location graph for METIS. The numbers in the graph indicate the weights assigned to the vertices and edges.

### 4.4.3 Scaling with a hybrid MPI-Threads Implementation

This section presents the scaling results for a EpiNet-MPI implementation of EpiNet using Intel's TBB framework. To recap, the TBB implementation parallelizes the infection computation using TBB tasks (or threads) and groups all communication of new infections at the end of the computation step. In our implementation, a single MPI process is used on each node and the computation is distributed across 8 TBB tasks. We use the default TBB task scheduler. We have also experimented

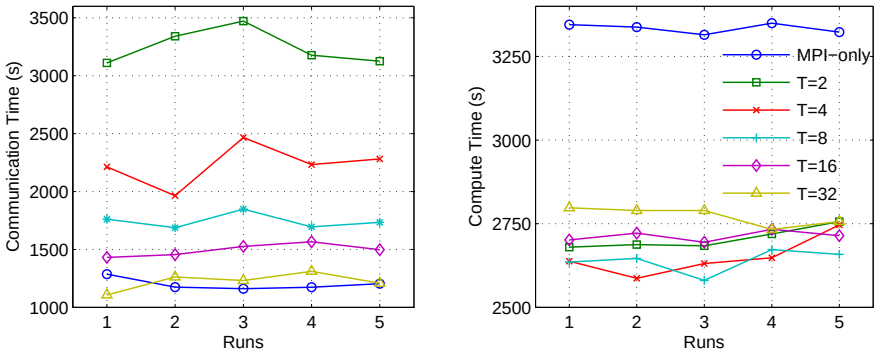



(a) (b)

Figure 4.14: Study and comparison of load balancing strategies. **(a):** Total simulation time comparison between load balancing based on identifiers (implemented in EpiNet) and METIS partitioning of the device–location graph. Not much improvement is obtained when METIS is used for partitioning the load; **(b):** Processing and communication times for two load balancing schemes. Improvements obtained in processing (communication) are offset by communication (processing) overheads, thus reducing the advantage gained.

with a different number of TBB tasks but present representative results with 8 tasks.

EpiNet-TBB still maintains the strong scaling behavior of EpiNet-MPI. However, we do not see a marked improvement in the overall performance (Figure 4.17). Digging deeper, we found that while the infection computation time in EpiNet-TBB decreased noticeably (Figure 4.15b) as compared to EpiNet-MPI, this gain was offset by an increase in the communication time (Figure 4.15a). This is an artifact of our implementation choice of separating out infection computation and communication in EpiNet-TBB. We are currently investigating a technique to better overlap computation and communication to improve the runtime for the hybrid implementation.




(a) (b)

Figure 4.15: Comparing compute and communication time of EpiNet-MPI and EpiNet-TBB. Computation and communication timing comparison between EpiNet-MPI and EpiNet-TBB with different thread counts in the Miami1 network. **(a) & (b):** Total communication and computation time for 5 simulation runs compared with EpiNet-MPI and EpiNet-TBB implementation (Note both use the same legend).

### 4.4.4 Evaluation of Load Balancing Approaches

The current implementation of EpiNet distributes the data in a round robin manner using an identifier assigned to each device and location. This strategy results in a relatively even distribution of load across compute nodes and has a simple implementation. However, we were interested in evaluating this approach against a partitioning method using static knowledge about the communication

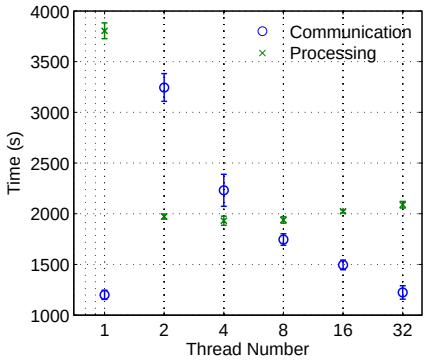



Figure 4.16: Tradeoff in processing and communication costs in EpiNet-TBB. Tradeoff between processing and communication tasks is shown as the number of threads exploited for infection computation. Note that 'Thread = 1' is the case for EpiNet-MPI implementation. Best performance is obtained when 4 or 8 threads are used.

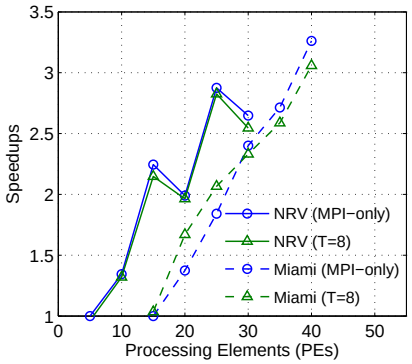



Figure 4.17: Scaling of the EpiNet-TBB implementation. Comparing the speedups of the EpiNet-MPI and EpiNet-TBB (with 8 threads per node) implementations. Infection computation with multiple cores does not improve when multiple cores are used.

patterns.

To this end, we employ METIS [86], a widely used graph partitioning program to achieve these partitions. A graph was created with nodes that correspond to locations and edges that correspond to the communication induced between these locations. It is assumed that the computational load is proportional to the number of devices at a location. METIS attempts to partition the nodes in such a way that each partition does the same amount of processing and the total edge weight between the partitions is minimized. Each partition is then assigned to a particular node.

We observe in Figure 4.14a that partitioning using METIS does not yield significant improvements over our round-robin strategy. The reason lies in the nature of the problem we are simulating. The contact graph of device interactions exhibits a relatively low amount of clustering. Intuitively, it is hard to partition people in groups, where the location that every group visits is disjoint from the location that other groups visit. Additionally, contact and device networks are essentially dy-

namic and depend on the individual and service provider responses as the simulation proceeds. There may be cases where static partitioning may help, but it is certainly not a general solution as our results indicate. We plan to explore more adaptive load balancing schemes in our future work.

## 4.5 Summary

In this chapter, we have described the improvements in the computational aspects of EpiNet. EpiNet is now a highly expressive and scalable HPC-based wireless epidemiology framework. To our knowledge, this is the only framework for malware spread that can scale to device networks of over 100K nodes and has the capability to use realistic device contact and mobility models, detailed diffusion processes, and intervention strategies. We have proposed three approaches to improve the scalability of EpiNet: (1) a lower mobility resolution to reduce communication overhead; (2) a model reduction technique to simplify the malware model; (3) a hybrid MPI-TBB implementation, EpiNet-TBB. The reduced mobility resolution provides a 100X improvement in communication (with respect to 1 sec. resolution) and contributes to $< 1\%$ error. The model reduction approach provides a speedup of 20X (with respect to the detailed model) and suffers a loss in accuracy of $\approx 5\%$. We observed that EpiNet algorithm scales well both with respect to problem size and number of PEs. We evaluated a static load balancing technique and found that it does not help in improving the scalability. Our current EpiNet-TBB implementation does not provide a noticeable speed-up due to dominating communication costs, but we have identified potential solutions.

Our performance results indicate that in addition to its expressiveness, EpiNet is also a highly scalable framework to study mobile malware diffusion over large, dynamic networks. By using a relatively simple partitioning technique, EpiNet is able to partition and distribute the work evenly across available PEs and scale up to a large number of PEs. Our results show that the current implementation has two main performance impediments: communication costs and optimization related to multicore architectures. We are addressing the former by looking at better partitioning strategies and the latter through improved EpiNet-TBB implementations. EpiNet can be extended

to study diffusion problems arising in the study of mobile social networks (e.g., opportunistic off loading of messages for viral information propagation) as well as malware propagation in hybrid networks with infrastructure support.

# Chapter 5

# Spread of Mobile Malware

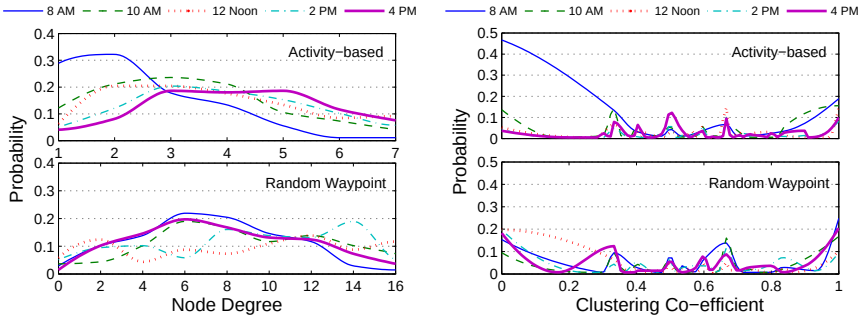
The primary application of the EpiNet simulation environment is its ability to study the spread of mobile malware. In this chapter, we conduct simulation studies to understand the propagation of mobile malware over realistic, dynamic networks. First, we present the comparison of the results with the activity-based mobility model with RWP. Because the simulation platform used for the RWP mobility model is ns-2, we can only conduct small scale studies with the simulator. So, our comparisons are performed on a single location. Next, we outline the configuration for the simulation studies we have conducted using the environment. We discuss the networks and the experiment design for the studies presented here. The initial set of experiments were conducted using EpiNet, the initial version of the simulation framework. Due to scalability limitations, we conduct the initial set of experiments on a smaller network of the Chicago downtown area. We also present the spread patterns obtained on larger networks using a more scalable version of the EpiNet simulator. Third, we present and analyze the nature of the spread we obtain on the Chicago network with no strategies implemented to control the spread. We then evaluate the effect of the within-host malware model parameters (such as idle time and probability of infection) and network parameters (such as market share). We perform a 3 factor analysis of variance (ANOVA) on the simulation response to determine the interactions between these parameters.

The nature of the malware spread is largely dependent on the underlying network. For accurate

estimates of the growth of the malware, an accurate network representation is extremely important. The spreading patterns with RWP and the activity-based mobility models are very different. RWP shows a dramatic spread of the malware and almost all the nodes become infected in a few seconds. This can be attributed to the property of RWP that the nodes tend to concentrate at the center of the location and thus exhibit different network characteristics that are a side effect of the mobility generated. The activity-based models indicate a much slower spread than RWP. This is due to the nature of the mobility where people (and devices) remain at locations for the duration of their activities and and then move to other locations. Thus, the network is dynamic, and under such a scenario, the device density at locations tends to vary and thus the propagation is slower. The sensitivity analysis on the various malware model and network parameters indicates that the spread is very strongly influenced by the market share of the susceptible device. Market share of devices has a significant impact on the speed of the spread. We perform a formal statistical test, ANOVA, to determine the interaction between the malware related parameters and the network parameters. Our findings indicate that there is significant interaction between the parameters and that the most interaction is between the idle time between infection cycles and the market share of the susceptible devices.

## 5.1 Comparison with RWP

In this section, we attempt to compare the malware spread with the most commonly used mobility model, random waypoint (RWP) model. Before going into the details of the malware spread, we look at the differences in the structure of the network created by these models. The RWP mobility model uses several parameters for generating the mobility—number of nodes, location area, minimum and maximum speeds, and pause times. The activity-based model on the other hand uses the synthetic population and their activities to generate occupancy in a location. The occupancy of a location in activity-based mobility is time varying and so for the comparison with RWP, we obtain average occupancy as the number of devices for RWP. RWP model generates the mobility information with minimum speed 0.5 $m/s$ and maximum speed 1.5 $m/s$ with pause times

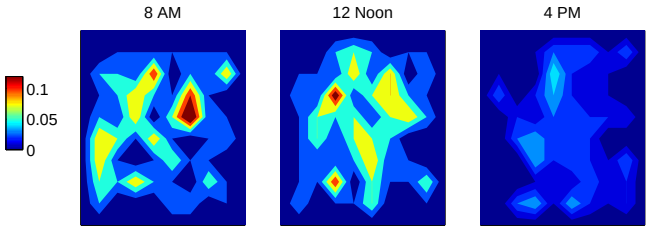



(a) (b)

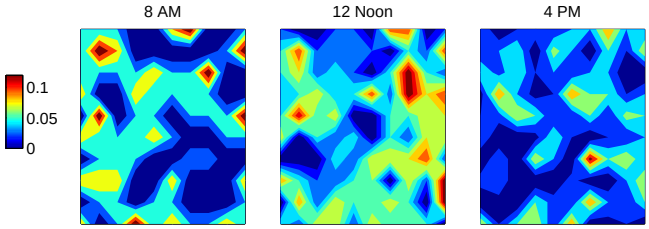
Figure 5.1: Graph Characteristics obtained from activity-based mobility models. **(a):** Comparison of degree distribution between activity-based (top) and RWP models (bottom) at different snap shots during the 8 hour duration of simulation; **(b):** Clustering co-efficient distributions for activity-based (top) and RWP (bottom) for the same time instants.

of 300 $s$ and 600 $s$. Table 5.1 shows the parameters we use for the mobility model comparison. We generate the mobility using both models for the duration of 8 hours and take hourly snapshots of the network.

Figure 5.1 shows the network structure in terms of the degree and clustering at different instants of the simulation and compares the two models. Clearly, the network structure is different. The hourly degree distribution for the two networks is shown in Figure 5.1a. Note that the x-axis in Figure 5.1a is different for each sub-plot. This clearly shows that the RWP model results in nodes with larger degrees than the activity-based model even when the location has a lower number of nodes. The maximum degree in case of activity-based model is 7. From Figure 5.1b we can see that the clustering in RWP models is slightly more than ABMM and also results in a faster spread of the malware. It is well known that RWP exhibits the property of high device density at the center of any given area under stable conditions. Figure 5.2 shows the density at the locations using RWP (Figure 5.2a) and activity-based model (Figure 5.2b). We can observe that in the RWP model the nodes are concentrated at the center of the location whereas for the activity-based model, the nodes are distributed throughout the location. This concentration at the center causes the higher




(a) Random Waypoint Mobility Model




(b) Activity-based Mobility Model

Figure 5.2: Node density comparison of RWP and ABMM. Node density distribution snapshots at a location of RWP and activity-based mobility models at different times of the day. **(a):** Node density at the location at 8 AM, 12 Noon, and 4 PM for RWP model; **(b):** Node densities for the same instants for activity-based mobility model.

Table 5.1: Simulation parameters for RWP and activity-based mobility models for the comparative study. RWP scenario was simulated with NS-2 and the activity-based mobility model was studied using EpiNet.

| **Parameters** | **Random Waypoint Model** | **Activity-based Model** |
|---|---|---|
| Number of Locations | 1 | 1 |
| Node Number | 109 | 91–147 |
| Node Velocity | 0.5–1.5 $m/s$ | – |
| Pause Time | 300 $s$, 600 $s$ | – |
| Node arrival & departure | No arrivals/departures | Every 300 $s$ nodes arrive or depart |
| Initially infected | 1 infected device | 1%, 5% & 10% |

degree and slightly higher clustering observed in RWP network. This property has been observed by Bettstetter [14].

We will now look at the worm spread dynamics in the two networks. Making comparisons between activity-based and RWP models is not straight-forward as they depend on different parameters. The instantaneous occupancy of the location (computed for every 300 $s$ interval) for the activity model ranges from 91–147. Because the arrivals and departures are rounded to a 300 $s$ interval, the occupancy does not change in the interval we consider. The total occupancy of the location is 572 individuals. For RWP we consider 109 devices, the average occupancy for the duration of the simulation. Because the scenarios have a different number of nodes, we cannot perform a direct one-to-one comparison. We varied the parameters of the initial infection sizes in case of ABMM (1%, 5%, and 10% devices initially infected) and the pause time (300 $s$ and 600 $s$) and speeds ranging from 0.5 $m/s$ and 1.5 $m/s$ in case of RWP. The simulations were conducted using EpiNet and NS-2 for ABMM and RWP, respectively.

We simulate the scenario for an hour (between 8 AM and 9 AM) and observe the number of nodes infected by 9 AM. In case of the activity models, we randomly select 1%, 5% and 10% of the location's total occupancy as the initial infection size and consider a single randomly infected

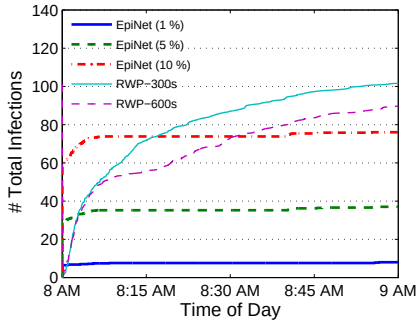



Figure 5.3: Infection spread comparison with RWP at a single location. Comparison between the infection spread with RWP and activity-based mobility models within a single location. RWP shows shows a much faster rate of propagation due to a property of RWP that causes a high density of devices at center of the area.

device for RWP. Because there is no direct relationship between devices in RWP and activity models, we cannot select the same device to be infected in both cases. We compare the number of devices infected at the end of the hour. Figure 5.3 shows the infection growth (averaged over 5 seeds) comparison between EpiNet (EpiNet 1%, EpiNet 5%, and EpiNet 10%) and RWP models (RWP-300s and RWP-600s). For the RWP model, the infections surge initially and infect almost all of the devices present in the location within the first hour. This can be attributed to the higher degree of the RWP network as seen in Figure 5.1a and the mixing of the devices in the location. In fact, we observe a higher rate of infection spread for the case when pause times are lower (300 $s$). Clearly, the mixing of the devices with other infected devices causes the higher rate. However, the initial surge of infections quickly saturates for activity models. Note that the initial infection size is different in both cases. When 1% of devices are infected, we have actually infected 5 devices. Yet the infection spread does not go beyond 10 devices. This can be attributed to the activities of the individuals and the duration of contact with other infected devices. For example, a person carrying an infected device may leave the location without interacting with a lot of people or devices due to a density variation in the location. Random models as shown in Figure 5.3 can predict a very high level of growth in the infection when in reality, the growth saturates. The activity-based mobility is much more accurate in representing human mobility and is extremely important to consider in

evaluating mobile epidemics.

## 5.2 Data set, Assumptions, and Experimental Design

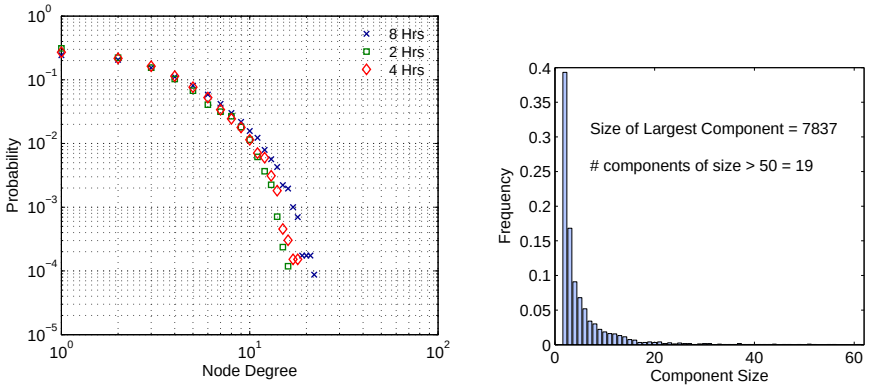
We use the methods outlined in Section 3.2.1 to construct a Bluetooth network for a region in Chicago downtown corresponding to the zip code 60602 and select the locations in this area. The size of the population is 30474 device/people and 2200 activity locations. The activity extracted is for a duration of 8 hours, from 8 AM to 4 PM, at these locations. Figure 5.5 shows some properties of the resulting proximity network; notice the non-Poisson nature of these distributions. People within an age range of 20–50 are selected to own digital devices. For some experiments we use the smart phone penetration or market share of a device as a parameter for people in this demographic class. In all others we consider that the entire network is composed of similar devices, i.e., all of them having the same vulnerability (or a market share of 100%). This provides the worst-case estimates of the dynamics and the final infection size as there is no heterogeneity. In a real scenario, market share of the devices rarely reaches 50–60%. Although in future the market share is expected to consolidate on a few popular phones – when this issue is going to bring some homogeneity in the device distributions. Each experiment takes approximately 2 hours when no response schemes are implemented and is repeated for 5 seeds. We report the average spread values in the plots.

Figure 5.4 shows the high level design and the parameters of the experiments. We study the dynamics of worm spread using two measures: (i) the cumulative number of infections, as a function of time of day, and (ii) a function $T(q, .)$, that denotes the time taken for a $q$-fraction of devices to get infected, and (.) indicates the variable that is being altered in the study. For example, when we study the spread by varying the idle time of the worm $T_{idle}$, then (.) represents $T_{idle}$.

Now, we study the malware propagation characteristics and look at some of the malware and network properties and study their impact. The following experiments are performed on the Chicago network. Some of the results obtained are specific to the network and its underlying structure. First, we look at the effect of the malware parameters' idle time ($T_{idle}$) and probability of timeout occurring after the inquiry request with no responses ($p_{to}$). Malware implementations can use the

1. **Network:** Synthetic population from the city of Chicago (zip 60602). Activities and activity durations are rounded to 5 minute intervals (300 $s$).
2. **Network Size (People/devices and locations):** 30474, 2209
3. **Computing resources:** 1 CPUs of 1GHz Pentium III Linux with 1 GB RAM
4. **Number of initial infections:** 1%, 5%, 10%
5. $T_{idle}$: 20 $s$; $T_{to}$: 12.80 $s$, $N_{resp}$: 4
6. **Infection seed time:** 8 AM (at the beginning of simulation)
7. **Simulator used:** EpiNet (an initial version of EpiNet was used in these experiments.)
8. **Seeds:** 5 (for each combination of input parameters)
9. **Simulation duration:** 8 hours, from 8 AM to 4 PM
10. **Studies:**
    (a) *Influence of worm parameters:* $T_{idle}$, $p_{to}$
    (b) *Influence of network parameters:* Market share ($m$), Location Density ($d$)
    (c) *Responses to Malware Spread:*
    - Graph metrics: Degree and Betweenness
    - Device-based detection: Passive self-detection ($sd_{th} = \{1, 2, 4, 6, 8, 10\}$)
    - Detection with Signature dissemination: Local and Centralized signature dissemination
11. **Average runtime:** $\approx$ 2 hours (much shorter when responses are implemented)

Figure 5.4: Experimental setting and parameters studied




(a) (b)

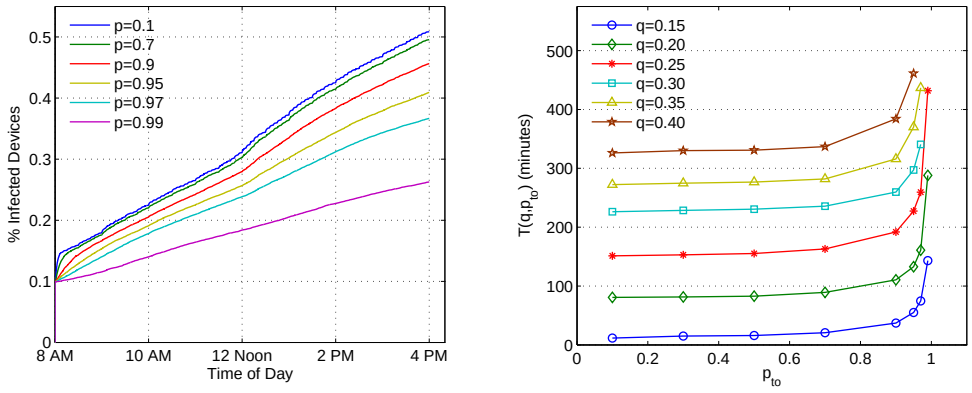
Figure 5.5: Network characteristics for the data set used in the experiments. Network characteristics obtained as a result of the activity-based mobility models and the *sub-location modeling*. We use a union graph for determining these measures from individual networks constructed every 300 $s$. **(a):** Degree distribution of the network; **(b):** Histogram of the component sizes in the network.

idle time parameter to create stealth while propagating to susceptible devices. These techniques can make it difficult to detect the malware. $p_{to}$ is the probability that an inquiry request does not discover any neighboring Bluetooth devices. This parameter is interesting as it can help us understand the effect of configuring a Bluetooth device as non-discoverable. Next, we will vary the network level parameters that change the underlying network. In this evaluation, we consider the location density ($d$) and market share ($m$) to observe the effect.

## 5.3 Sensitivity of Bluetooth and worm parameters

Figure 5.6a shows the spread characteristics. Comparing the studies performed in earlier work, we find that the worm spreads much more slowly in our model than in [99, 94]. Additionally, we find that a smaller fraction of devices is infected; e.g., at most 50% of devices become infected during 8 hours starting with 10% initially infected devices. This is clear evidence of the specific mobility model underlying our data-set, which mixes much slower than uniform mobility, and is much more

heterogeneous.

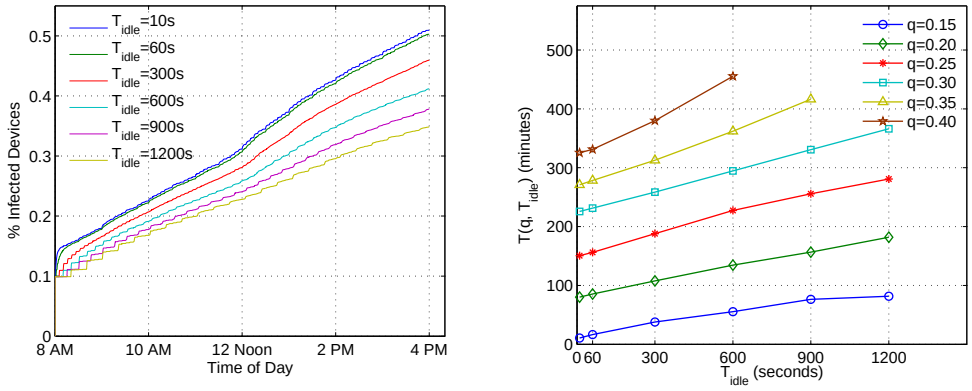



(a) (b)

Figure 5.6: Infection spread dynamics with variation in $p_{to}$. **(a):** Infection spread with varying $p_{to}$ starting with 10% initially infected devices; **b:** $T(q, p_{to})$ for the varying $p_{to}$ values in the same scenario.

We now study the sensitivity to some of the parameters in the worm model in Figure 3.8, namely the timeout probability ($p_{to}$) and the idle time ($T_{idle}$). First, we look at the effect of varying the probability that an inquiry request timeout occurs without a single inquiry response. We denote this probability by $p_{to}$. Surprisingly, as shown in Figure 5.6, there is a very limited effect of $p_{to}$ on the total infections seen. Until $p_{to}$ becomes 0.9—meaning that the infected devices' timeout occurs in 90% of inquiry requests—the infection still spreads to a large number of devices (about 45%). Beyond this, the successful completion of inquiry with a response reduces and the infection does not seem to take off. Only about 20% of the devices become infected starting with 10% initially infected devices. Clearly, this shows that disabling the *discoverable* mode in Bluetooth devices can cause a large slowdown in the spread of the malware. Figure 5.6b shows the plot of $T(q, p_{to})$, i.e, the time taken to infect q-percentage of devices when varying $p_{to}$.

Next, we vary the idle time of the worm to observe the effect of a stealthier worm on the spread. Figure 5.7 shows the effect on the infection spread. Figure 5.7a shows the cumulative percentage of infected devices as $T_{idle}$ is increased. Intuitively, it would seem that an intelligent worm can




(a) (b)

Figure 5.7: Infection spread dynamics with variation in $T_{idle}$. **(a):** Infection spread with varying $T_{idle}$ starting with 10% initially infected devices; **(b):** $T(q, T_{idle})$ for the varying $T_{idle}$ values in the same scenario.

adapt its idle time to maximize the spread. However, we find that the spread obtained with 20 $s$ idle time denotes the upper limit of the worm propagation. Reducing the idle time to 10 $s$ does not result in an increase in the total infected devices. As expected, the idle time does affect the initial speed of the spread as the devices wait longer in each infection cycle. Nevertheless, once a certain number of devices become infected, eventually the speed increases to infect more devices. Figure 5.7b shows the $T(q, T_{idle})$ values for the variation in $T_{idle}$. For most values of idle time at least 35% of devices become infected starting with 10% initially infected devices.

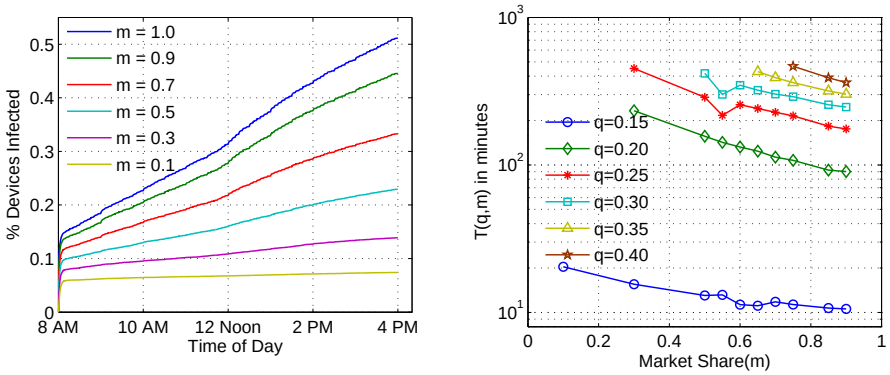
## 5.4 Sensitivity to Network parameters

We study the effects of network structure on the worm dynamics by comparing with the dynamics based on random waypoint mobility and by altering the network structure in a controlled manner. We conduct two studies that alter the network structure. Market share of smart phones have a large impact on the malware spread. Because device susceptibility depends on the operating system (OS) and applications on a device, market share of the OS and applications directly impact the malware

spread. So, we evaluate the effect of market share on the spread characteristics. We denote market share by $m$. The other aspect that we are interested in studying is the effect of density of locations. The sub-location modeling we perform is dependent on the area allocated to locations. Because we do not have data on the actual areas, we make assumptions by keeping a certain density at the locations and based on the occupancy, determine the area assigned. We denote the density of locations by $d$.

### 5.4.1 Effect of market share

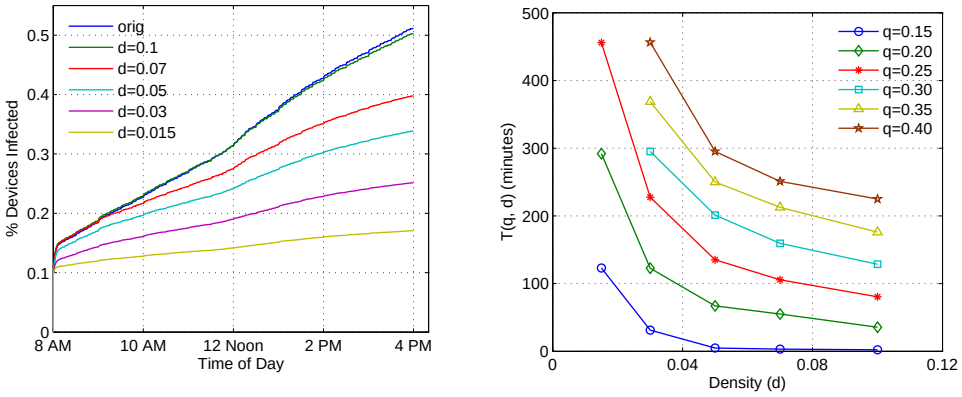
In this section, we evaluate the effect of the market share on the infection spread. We consider different market share values for the devices. The market share, $m$, represents the number of devices that have similar characteristics to the infected device and are susceptible to the same malware (e.g., a market share $m = 0.1$ means that only 10% of the devices are susceptible among the entire device population). The market share is a simple parameter that allows us control over the susceptible fraction, and here we study its impact on the worm dynamics. We consider a range of $m$ values from 10% to 90%.




(a) (b)

Figure 5.8: Effect of market share on the infection spread. **a:** Infection spread with varying market share of the susceptible devices; **b:** $T(q, m)$ for different values of $q$ and varying market shares of the susceptible devices.

Figure 5.8a shows that the speed of the worm is directly proportional to $m$, as expected, and both the speed and the final outbreak size are reduced with $m$. However, we find that the $T(q, m)$ function (Figure 5.8b) shows very different characteristics, when compared with the results of [94]; in particular, we find a steeper variation in $T(q, m)$, and this suggests effects arising out of our detailed mobility model, because [94] assumes a uniform distribution of susceptible devices within a cell. We can also see a threshold effect. The time taken to infect at least 20% of the devices starting with 10% initially infected changes drastically for all values of market share. This kind of behavior indicates that there is a significant amount of time to implement response mechanisms to control the spread during this interval.




(a) (b)

Figure 5.9: Infection spread dynamics with variation in location density ($d$). **(a):** Infection spread with varying $d$; **b:** $T(q, d)$ changes with varying density of locations.

### 5.4.2 Effect of location density

Next, we control the network structure by altering the density within the locations in our dataset. Clearly, a higher density implies higher degree, and would suggest faster spread (analogous to the impact of higher speed in [99]). This is indeed borne out in Figure 5.9a, where we reduce the density by a parameter $d$, and examine the rate of spread. A surprising observation is that there seems to be some kind of threshold effect, and the dynamics do not change until the density has

been altered quite a bit (by a factor of 10).

## 5.5 Characterizing interactions between parameters

An important research question important to address in any study with a large number of parameters is whether there is any interaction among them. The worm model parameters such as $T_{idle}$ and $p_{inf}$ and network parameters such as market share $m$ and density of locations are of particular interest. Understanding how they couple together and change the response, in this case the final % of devices infected, is important. We set up a *balanced*, *factorial experiment design* for these parameters with 5 levels each and measure the *response variable*, the final % devices infected for 10 replicates. We use analysis of variance (ANOVA) to perform statistical analysis and present the results in this section. These experiments are performed using the EpiNet simulator on the NRV1 network.

Our findings are that these factors interact very strongly with each other; specifically, $T_{idle}$ and $m$ interact the most among the three factors.

### 5.5.1 Analysis of Variance Study

We set up a statistical study to evaluate the outcome of the following three independent factors (variables):

- Idle time between infection cycles, denoted by $T_{idle}$,
- probability of infection, $p_{inf}$, and
- market share of the devices, $m$.

Some of these factors are derived from malware characteristics and the networks that are being evaluated. Each of these factors have five levels (values each variable takes). Each experiment in the design is repeated 10 times (replicates). See Table 5.2 for the levels for each factor. $T_{idle}$

indicates how stealthily the malware propagates and evades detection. The larger the $T_{idle}$, the slower the malware spreads and the harder it becomes to detect. $p_{inf}$ indicates the probability with which a susceptible device exposed to an infected device becomes infected. It basically indicates the efficacy of the propagation method. The higher the probability, the more effective the infection propagation becomes and the higher the total number of infected devices. The market share $m$ decides the proportion of susceptible devices that are present in the network and indirectly represents the community within which a particular malware can spread. The higher the market share, the larger the susceptible device population and probability an infected device will come into contact with a susceptible device.

Table 5.2: Levels for the factors in the ANOVA study. There are 5 levels for each factor and 10 replicates for each combination of these parameters.

| **Factors** | **Levels** |
|---|---|
| $T_{idle}$ | 30s, 5min, 10min, 20min, 30min |
| $p_{inf}$ | 0.1, 0.3, 0.5, 0.7, 0.9 |
| $m$ | 10%, 30%, 50%, 70%, 90% |

The mathematical model for 3-factor ANOVA is as shown below:

$$y_{ijkl} = \mu + \alpha_j + \beta_k + \gamma_l + (\alpha\beta)_{jk} + (\alpha\gamma)_{jl} + (\beta\gamma)_{kl} + (\alpha\beta\gamma)_{jkl} + \epsilon_{ijkl} \tag{5.1}$$

where

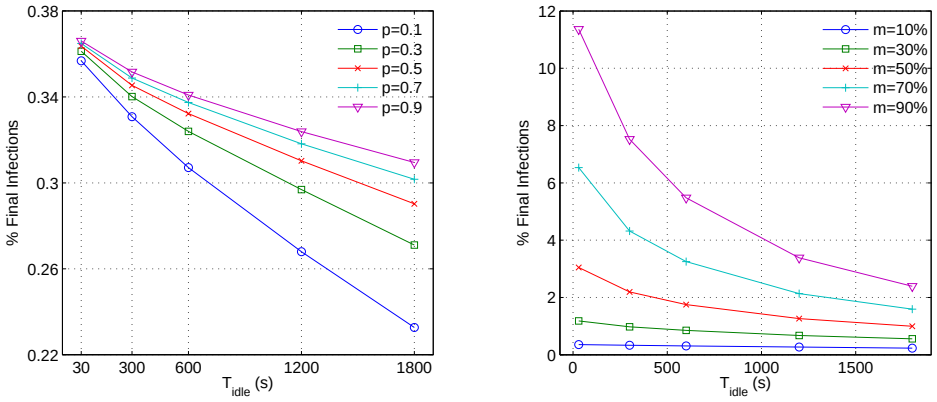
1. $y_{ijkl}$ is the measurement of the response variable; in this case, it is the percentage of devices infected at the end of the simulation (or the % Final Infections)
2. $\alpha_j$, $\beta_k$, and $\gamma_l$ are the effects of the $T_{idle}$, $p_{inf}$, and $m$ on the outcome.
3. $i$ is the number of replicates in the experiments; in this case, it is 10 as we repeat the experiment 10 times for different seed values.

In addition to the effect of individual factors, we have also computed the interactions among them. Here we present the interactions between the probability of infection ($p_{inf}$), idle time ($T_{idle}$), and the market share of the devices ($m$). $p_{inf}$ is a malware parameter and identifies a more virulent malware and how effectively it can spread. The market share $m$ denotes the available susceptible device population and reflects the network structure. It is expected that there is a big influence of both these parameters, and we are trying to understand which parameter actually has a higher influence and under what conditions.

Table 5.3: Interaction study for $T_{idle}$, $p_{inf}$, and $m$. 3-factor balanced ANOVA to determine the interaction between idle time $T_{idle}$, probability of infection $p_{inf}$, and market share $m$. T, P, and M represent the $T_{idle}$, $p_{inf}$ and $m$, respectively. There is a high level of interaction between all the parameters and $T_{idle}$, and $m$ interact the most (Model 3).

| No. | Interactions | Source | SS | DF | F test |
|---|---|---|---|---|---|
| 1 | All 1-way | [T][P][M] | 12053 | 1237 | 3791.923* |
| 2 | 2-way | [TP][TM] | 4669 | 1205 | 4063.270* |
| 3 | 2-way | [TP][PM] | 7514 | 1205 | 6880.722* |
| 4 | 2-way | [TM][PM] | 1002 | 1205 | 431.778* |
| 5 | All 2-way | [TP][TM][PM] | 566 | 1189 | 120.075* |
| 6 | All 3-way | [TPM] | 71 | 1125 | - |

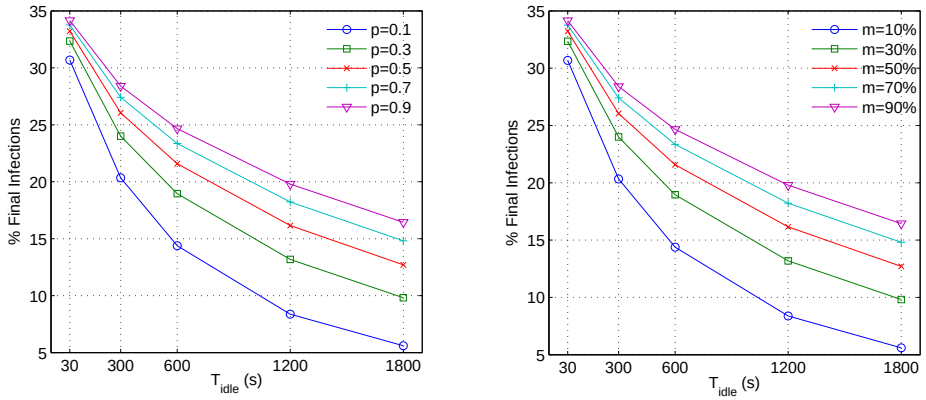
Table 5.3 shows the results for 3-factor ANOVA to determine the interaction between $T_{idle}$, $p_{inf}$, and $m$ study for malware propagation. Column 1 shows the model number for each of the interaction models indicated in column 2. The parameters that interact are listed in column 3. Columns 4-6 show the interaction study on the response variable, the final % of devices infected. The F-statistic shows the interaction and the level of interaction from among the factors. From the table, we see that there is significant interaction between the 3 factors. The interaction in the 3-way automatically indicates this. Among the factors, $T_{idle}$ and $m$ interact the most (model 3 in row 4). The two way interaction that is dropped from model 5 is `[TM]` and so it shows the interaction between the 2 factors.




(a) (b)

Figure 5.10: Interaction of idle time $T_{idle}$ with $p_{inf}$ and $m$ for the NRV1 network. We plot the interaction as a function of the percentages of devices infected at the end of the day (denoted as "% Final Infections"). **(a):** Shows the interaction between $T_{idle}$ and $p_{inf}$ for a low market share ($m = 10\%$); **b:** Plots the interaction between $T_{idle}$ and market share, $m$ for a low probability of infection ($p_{inf} = 0.1$).

Figure 5.10 shows the interaction of $T_{idle}$ with $p_{inf}$ and $m$ for the NRV1 network. The x-axis shows the variations in $T_{idle}$ and the y-axis shows the percentage of devices finally infected at the end of the day.




(a) (b)

Figure 5.11: Interaction of idle time, $T_{idle}$ with $p_{inf}$ and $m$ for the NRV1 network. We plot the interaction a function of the percentages of devices infected at the end of the day (denoted as "% Final Infections"). **(a):** Shows the interaction between $T_{idle}$ and $p_{inf}$ for a low market share ($m = 10\%$); **b:** Plots the interaction between $T_{idle}$ and market share, $m$ for a low probability of infection ($p_{inf} = 0.1$).

## 5.6 Study of SMS/MMS Malware and their propagation

In this section, we study the dynamics of the SMS/MMS malware spread and differentiate it from the proximity based Bluetooth malware. The social network we use for this study is derived from the SG module discussed in Section 3.2.2. Here we conduct two simple experiments to determine the speed and dynamics of a purely SMS/MMS malware that propagates using the SMS/MMS message. We refer to such malware as *sms-only*. We also study hybrid malware that jump from device to device over proximity Bluetooth link as well as a link over the cellular infrastructure through SMS/MMS messages. We refer to these malware as *hybrid*. We find that the dynamics of

Table 5.4: Networks studied using the EpiNet framework. Characteristics of the proximity networks studied in this section. The networks have varying sizes and structural properties that make it necessary to study the spread and control of mobile malware for each network. The EpiNet framework is not able to study these networks as efficiently as EpiNetdoes.

| | NRV1 | NRV2 | Miami1 | Miami2 |
|---|---|---|---|---|
| Demographics (age group) | 20-50 | 13-80 | 20-50 | 13-80 |
| # Devices | 77,659 | 126,800 | 1,269,650 | 1,642,565 |
| # Links | 284,941 | 1,507,328 | 4,495,755 | 17,266,216 |
| # Social Links | 699,984 | 1,478,686 | 22,652,784 | 35,796,647 |
| 13-20 years | – | 10.28% | – | 12.85% |
| 50-80 years | – | 29.92% | – | 32.48% |

the SMS*sms-only* malware under some conditions are almost similar to the Bluetooth malware in nature. When hybrid malware spread, their dynamics are different from *sms-only* malware. They spread quickly and infect almost the entire susceptible population within a few hours.

### 5.6.1 Implementation of SMS/MMS malware

The EpiNet framework has the capability to study communication networks in addition to proximity networks created by Bluetooth. In this section we highlight this aspect and perform some simple studies on the propagation characteristics of malware that use the communication infrastructure on mobile devices. SMS malware is implemented in the framework as stealthy malware that intermittently sends SMS messages to the people in the address book of an infected device. These malware can be of differing kinds and can select the victim number at random or select the user based on call statistics. A more frequently called individual is targeted before an infrequent one and so on. We evaluate the two approaches and find that in a fast spreading SMS malware the randomness or preferred selection strategy results in similar dynamics.

#### 5.6.1.1 Comparing the propagation dynamics

In this section, we observe the propagation under 3 conditions – (1) only Bluetooth malware, (2) only SMS malware and, (3) hybrid malware that propagates on both mediums. We use this to determine network structures that cause change in the patterns of infections.

Figure 5.12 shows the propagation of the 3 kinds of malware on the NRV1 and NRV2 networks. We observe a strange behavior in the NRV2 network. The propagation dynamics for the two networks are different because the NRV2 network has more social links (as shown in Table 5.4) than the NRV1 network. As expected, the hybrid malware propagates faster than either the Bluetooth malware or the SMS malware alone (Figure 5.12a and Figure 5.12b). There is an interesting dynamic for the SMS malware in the NRV2 network as shown in Figure 5.12b. We observe that the Bluetooth malware propagates slowly initially and still achieves a significant growth beyond 12 Noon and infects almost the same % of devices as the hybrid malware. This is an interesting behavior and is due to the number of social links shown in Table 5.4. The social links indicate the contacts in the SMS network (or the social network of the individuals). It appears that the devices are coming in proximity more than communicating through the SMS links. This can be the result of having high density locations, where devices are in range of other devices. This is an interesting case where we see that the proximity malware spreads much faster than the SMS malware. Accurately modeling the network is a necessity to obtain the right conclusions and depends heavily on aspects of human mobility.

## 5.7 Summary

In this chapter, we have presented the study of the Bluetooth malware using the EpiNet and EpiNet simulation infrastructures. We compare the results of malware propagation with RWP mobility model and find that an accurate representation of the network is important to obtain correct results. Using the EpiNet simulator we have studied the influence of various malware properties such as idle time and probability of inquiry timeout. Further, we have looked at the

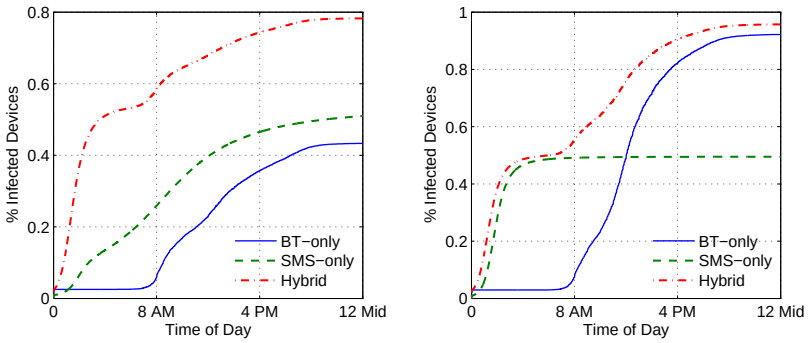



Figure 5.12: Comparing the dynamics of Bluetooth, SMS, and hybrid malware in the NRV network. Propagation dynamics in the NRV1 networks for the 3 kinds of malware. We consider Bluetooth alone, SMS alone, and hybrid malware that spreads through both means.

influence of network aspects such as market share of susceptible devices and density of activity locations on the spread dynamics. We find that each of these parameters have a significant influence on the spread. We conduct a more formal statistical test, analysis of variance, to determine the interaction of parameters. For this, we create a factorial experiment design with 3 factors – $T_{idle}$, $p_{inf}$, and $m$ – with 5 levels for each factor. From this analysis we determine that the 3 factors idle time between infectious cycles and the market share have a significant interaction.

The comparison results between RWP and the activity-based mobility models in addition to the preliminary results with the EpiNet environment on the Chicago network were published in [22]. This work was performed in collaboration with Dr. Madhav Marathe, Dr. Anil Vullikanti, Dr. Keith Bisset, and Dr. Deepti Chafekar. The later studies on the NRV and Miami networks with the scalable EpiNet simulator appear in [23] and was submitted on October 1st. This work was performed in collaboration with Dr. Madhav Marathe, Dr. Anil Vullikanti, Dr. Keith Bisset and Mr. Shrirang Yardi.

In the next chapter, we study and analyze methods and strategies to control the malware spread on these networks.

# Chapter 6

# Controlling the spread of mobile malware

Responses to malware are primarily initiated by the service provider to protect the infrastructure and mobile devices from the malware. It is possible to detect malware that spread through SMS or MMS messages as they use the infrastructure and remedial measures such as black-listing and throttling network bandwidth to slow the spread. Not all manifestations spread in this manner. Proximity malware that spread locally (in the immediate neighborhood of the infected device) are hard to detect and respond to. Mobile-based detection rather than network-based detections can perform better in detecting and responding to such worms.

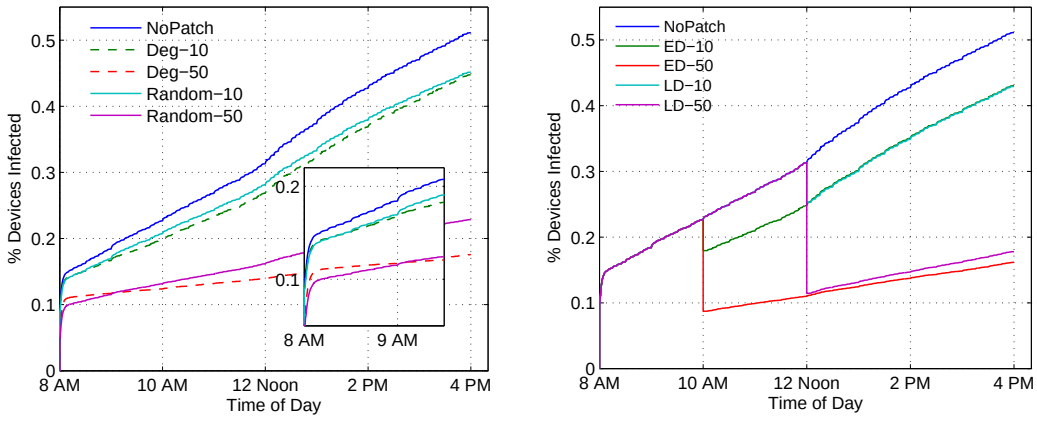
In this chapter, we study several mobile-based detection and mitigation schemes and measure their effectiveness in detecting and responding to malware. We classify the response mechanisms into (a) static and (b) mobile-based. In the static mechanisms, we study the influence of simple graph metrics such as degree and betweenness. Here, we are assuming that the service provider has some means to compute the proximity network for devices and then applies responses online based on the metrics computed on them. The provider uses these measures to identify the best candidates for applying the patches. The other and probably more effective response strategy in such ad hoc proximity networks is mobile-based detection and response, where devices can themselves detect their anomalous behavior [19, 54, 33, 72] and take some action: (1) disable the sub-system that is infected, (2) generate signatures that can help other infected devices know about their infection,

and (3) inform a centralized server of the infection to take remedial action. We are interested in determining the effective parameters values that can control the malware spread. Here again we are making an important assumption that the software and/or hardware involved in device-based detection is not affected by the malware. Further, in some of the response mechanisms, we are assuming that the sub-systems can be disabled temporarily and that on effective removal of the malware (through human interventions) these systems will be back online.

## 6.1 Graph Centrality Measures

Epidemiology studies for humans have long considered graph metrics based interventions or responses for epidemics. In this set of experiments, we are evaluating these strategies with the assumption that such networks can be constructed by the service provider. We study the effect of choosing devices that have a higher rank based on two graph centrality measures—*degree* and *betweenness*. Degree is defined as the number of links incident upon a node and defines in some ways the risk of infection from the malware propagating through the network. Betweenness is a measure of the number of shortest paths between any two nodes a given node is present on. It indicates the importance of a certain node in the topology. For all the studies involved in the graph measures, we consider the *union graph* for determining the rank of the devices. A union graph at a particular instant of time $t$, $(G(t))$ is the graph constructed by considering the union of all graphs created at the locations at $t$, i.e., $G(t) = \bigcup_{l \in L} G_l(t)$, where $L$ is the set of all locations and $G_l(t)$ is the graph in location $l$ at time $t$. The union graph $G$ is the union of all $G(t)$, i.e., $G = \bigcup_{t \in D} G(t)$, where $D$ is the duration of the simulation and time is incremented in steps of 300 $s$, the interval for new arrivals and departures at locations. We use $G$ to calculate the graph metrics for individual devices and rank them in decreasing order of the metric. Devices are selected to be patched in rank order of the metrics and the change in spread characteristics are observed. Devices so selected are not susceptible to the malware at the instant the patch is applied. We consider applying this intervention before and after the malware starts spreading.

Here, we study the effects of patching 10% or 50% of devices selected based on the rank of de-




(a) (b)

Figure 6.1: Comparison of degree-based response strategy. **(a):** Comparing random device selection to selection based on degree rank from the union graph; **(b):** Infection spread with devices patched early and late with devices selected with degree rank.

gree and betweenness of devices. We compute the union graph $G$ from the activity-based mobility model and compute the metrics on $G$. Human epidemic studies have indicated that degree based responses outperform betweenness based measures. Figure 6.1a shows a comparison between randomly selecting devices for patching and selecting devices based on the degree rank (the total devices patched remain same). 'Deg-10', 'Deg-50' indicate the result for patching 10% and 50% of devices by selecting the devices based on degree rank. 'Random-10' and 'Random-50' indicate 10% and 50% of devices being selected at random, respectively. The final infection size clearly indicates that degree based responses are better than random selection. The inset plot in Figure 6.1a shows that initially the random is slightly better than degree. From 9 AM onwards, the graph based metrics start to perform better. This effect is observed because high degree devices are determined from the union graph and are patched initially. Patching 10% devices does not have a significant impact on the spread characteristics and there is only a slight reduction in the final infection size. Any response strategy based on degree requires more than 10% of devices to be patched in order to be effective in controlling the spread.

Surprisingly, the effect of selecting devices based on the betweenness of devices has a similar effect on the control. Contrary to human epidemics where degree-based response mechanisms are more effective than betweenness, we find the effect with patching devices based on betweenness metrics identical to degree-based metrics.

## 6.2 Mobile device based Responses

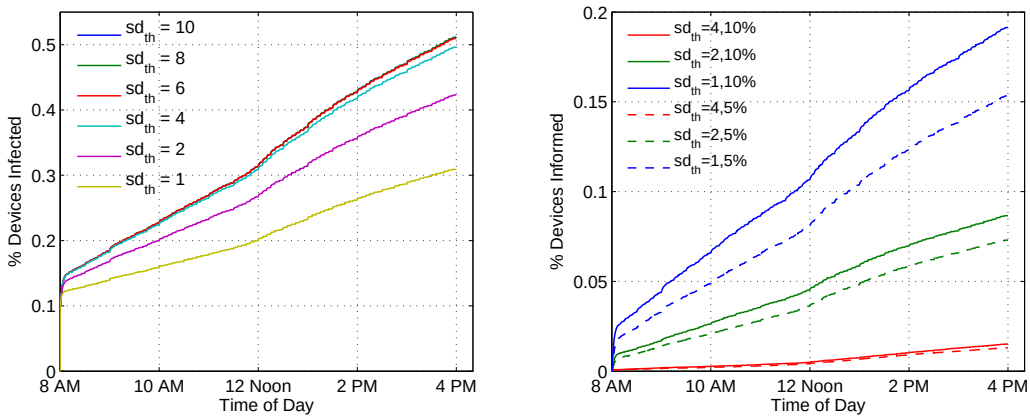
The studies conducted in the earlier sections observe the spread characteristics with heterogeneous networks and device market share. We also studied some static random and graph measure based responses and provide a base line for comparison with more advanced and practical responses. In this section, we explore the response schemes that use device-based detection mechanisms with and without service provider involvement in patching devices. In these responses, we mostly study *self detection* and other responses such as *local* and *centralized signature dissemination* that depend on self detection.

### 6.2.1 Passive Self Detection

Traditional network defenses are centralized and depend on information from packet traces to identify anomalous behavior. Such defense mechanisms cannot be applied to proximity malware as observations tend to be local and in the proximity of the infected devices. Recent research direction for recognizing local effects involve deployment of sensors in locations where device densities are expected to be high [43, 77]. In other, more recent schemes the potential victim serves as a detector. Monitoring sequence of system calls [33], power signatures [54] and behavior [19, 44] have the potential to be implemented in mobile devices to sense new infections.

We consider such detection mechanisms which allow devices to detect their own infection using behavior identification methods; this is passive in the sense that it uses the knowledge of the infection to prevent further spreading, but does not inform neighboring devices or service provider of the infection. Here we are interested in evaluating (a) whether such schemes reduce infection

spread and (b) the effect of varying the sensitivity—number of infections required to trigger the self detection—of the self detection system. We define the number of infections that trigger the self-detection as *self-detection threshold* and denote it by $sd_{th}$. In these experiments we vary $sd_{th}$ between 1 and 10 to determine the best threshold value to control the spread. The devices that become aware of their own infection are *informed* and have a means to stop further infections (by either alerting the user or disabling the sub-system or application spreading the malware).




(a) (b)

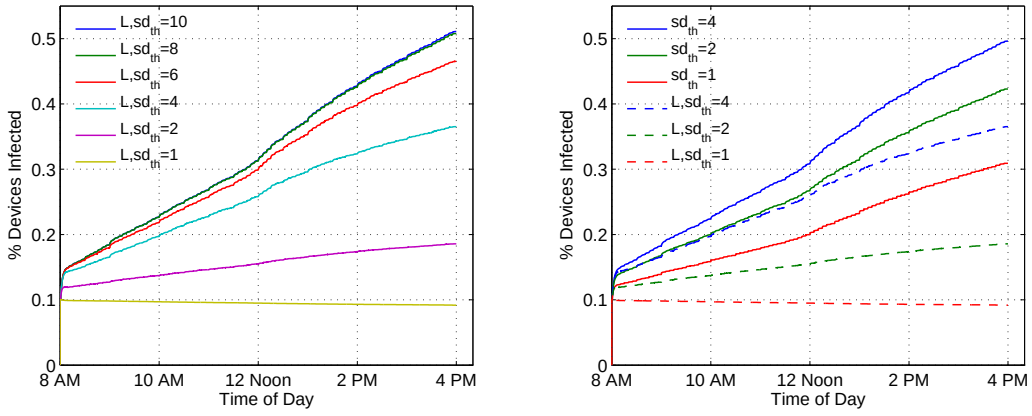
Figure 6.2: Passive Self-detection performance with varying detection thresholds. **(a):** Passive self-detection with 10% initially infected devices and $sd_{th} = \{1, 2, 4, 6, 8, 10\}$; **(b):** Total devices informed as the infection spreads with $sd_{th} = \{1, 2, 4\}$ for 10% and 5% initially infected devices.

Figure 6.2 shows the infection spread characteristics for 10% initial infection size (Figure 6.2a) with varying self detection thresholds ($sd_{th}$) from 1–10. $sd_{th}$ is defined as the number of infections caused by the infected device before the self detection system is triggered. The devices that become aware of the infection are called *informed* devices and assume that they do not cause new infections. From Figure 6.2a, we see that for any $sd_{th} > 4$ the spread does not slow and for $sd_{th} = 8$ and $sd_{th} = 10$ no devices self detect. This clearly shows that no single infected device causes more than 6 infections and only 2% infect more than 4 devices within the simulation duration. Further, even when $sd_{th} = 1$, the final infection size reduces by only 20%, implying that the self detection

mechanism has to be fairly accurate and extremely quick to detect an infection.

### 6.2.2 Detection with Signature Dissemination

Malware signatures are extensively used in existing anti-virus software to identify attacks and prevent infections requiring constant signature updates. Nevertheless, about 60% of malware developed to steal personal information are not detected by updated anti-virus software [3]. Automatic signature generation programs [72] or behavior identification systems [66] can help improve detection. Such approaches generate signatures online using the malware's characteristics and help keep the signatures updated. Here we evaluate such schemes implemented in mobile devices. We consider two approaches: (a) local dissemination of the signatures generated and (b) upload signatures to service provider for centralized dissemination. For this study we still use the same self detection discussed in Section 6.2.1, but improve the signature generation and dissemination process.




(a) (b)

Figure 6.3: Comparison between passive and local signature dissemination. **(a):** Local Signature dissemination for different $sd_{th}$ values; **(b):** Comparison between passive self detection and self detection with local signature dissemination for $sd_{th} = \{2, 4, 6\}$ values.

#### 6.2.2.1 Local Signature Dissemination

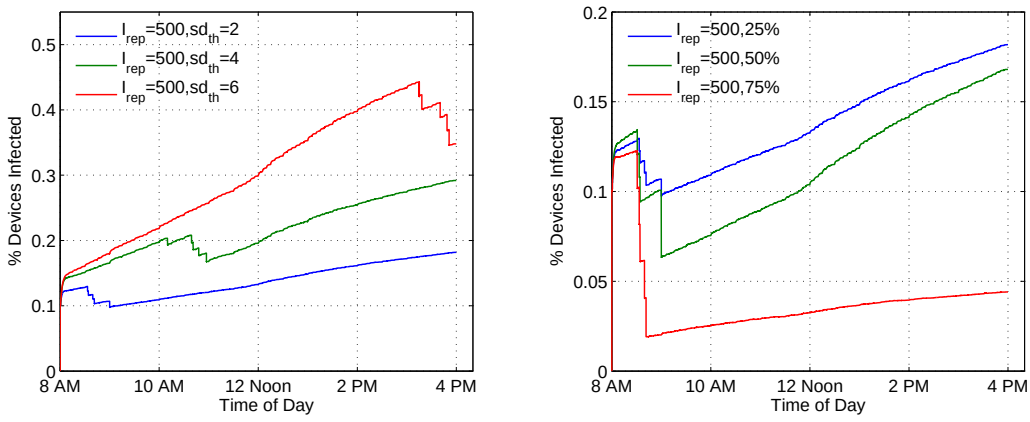
For local signature dissemination we consider that the *informed* device follows a reactive process—signature is shared only when an infected device tries to infect an *informed* device. We do not study a pro-active approach, as it becomes similar to the malware and would require significant resources from the mobile.

Figure 6.3a shows the results of performing local signature update to neighboring infected device by an informed device. 'L,$sd_{th}$=2' indicates the local (L) signature update with a self detection threshold $sd_{th}$ of 2. From Figure 6.3a it is clear that lower $sd_{th}$ values result in slowing the spread (similar to Figure 6.2a), but local signature updates slow the spread drastically when compared to passive self detection. Further, when $sd_{th} = 1$, some of the initially infected devices are informed of the infection and final infection size is smaller than the 10%. Thus, we find that in the current setting, local signature distribution (with passive self detection) is more effective than the passive detection approach. This approach requires the mobile to communicate with the neighboring infected devices and transfer the signature.

#### 6.2.2.2 Centralized Signature Dissemination

With local signature dissemination, the spread of the signatures is local to the position of the infected devices and does not have a spatial impact. So, here we evaluate the case where the informed devices send one *infection report* (or $I_{rep}$) to a centralized location. Infection reports are concise packets of information regarding the mobile's behavior or signatures generated after self-detection. The centralized entity can either generate the signature or serve merely to make decisions for distributing the signatures. We study the impact on the infection spread when the centralized entity applies patches on receiving a certain number of reports. We select different percentages of devices to patch and observe the spreading.

Figure 6.4 shows the results we have obtained for the different experiments with centralized signature dissemination. Here we are randomly deciding on the devices that receive the patches and




(a) (b)

Figure 6.4: Centralized signature dissemination for controling the malware spread. **(a):** Malware dynamics with centralized signature distribution with $sd_{th} = \{2, 4, 6\}$, $I_{rep} = 500$ and 25% devices patched; **(b):** Effect of patching 25%, 50%, and 75% of devices after receiving $I_{rep} = 500$ reports.

are not patching only the devices that are infected. For all the experiments presented here, we have used a fixed $I_{rep}$ threshold of 500 messages from the detectors. From Figure 6.4a we can see that lower detection threshold causes a faster response from the central server for patches. Even after application of the patches, for all the cases, the infection does not completely die out, but has significantly slowed down.

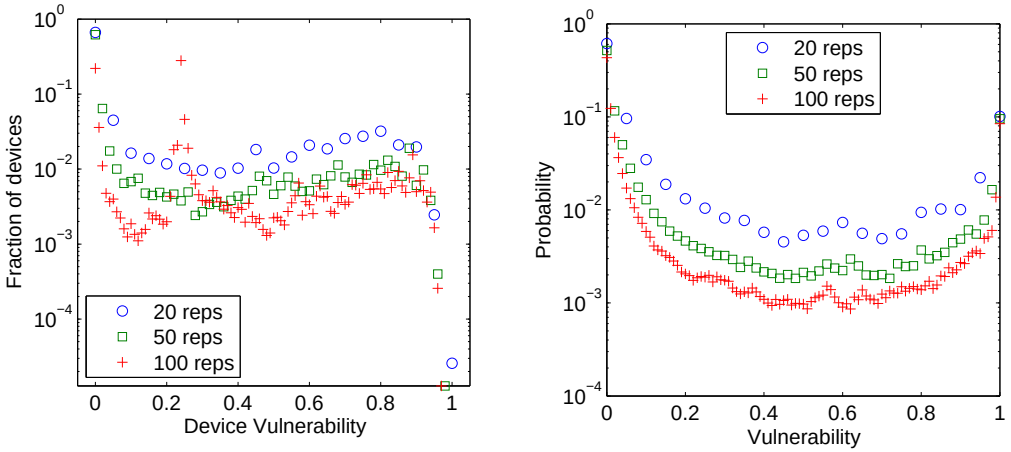
## 6.3 Device Vulnerability

The earlier sections targeted the responses based on graph metrics such as degree as they are found to be effective in human epidemiological studies. From our findings in the above sections, we note that static graph metrics such as degree and betweenness of the devices are not effective. Simply selecting devices based on the number of neighbors they have and ranking them according to this is not effective in slowing the spread. The other metric that has been shown to be effective is *vulnerability*. In this section, we explore the vulnerability of a device as a metric to decide which of

the nodes to patch and determine if that is effective in controlling the spread of mobile malware.

Vulnerability denoted by $V(v)$ for a node $v$ is defined as the probability with which $v$ gets infected under different initial infected devices. Vulnerability is a dynamical measure dependent on the temporal network and the dynamics of the malware. As we will show later, vulnerability is found to be an effective metric to control the malware spread. In a sense vulnerability of a device provides information regarding the probability of a node to be infected irrespective of which set of devices are initially infected. Because the Bluetooth device network is large and dynamic, we estimate vulnerability through simulation studies with a different set of devices initially infected and repeating the experiment for multiple replicates. For the purposes of this dissertation, we use 100 replicates while determining the vulnerability of a device. Figure 6.5 shows the effect of the vulnerability metrics when a different number of replicates are used to estimate vulnerability. Note that the initially infected devices are not counted while determining the number of replicates in which a given devices is infected. We can see that the metric varies largely depending on the number of replicates used to estimate the vulnerability of a device. Figure 6.5a and Figure 6.5b show the variation in the vulnerability for the NRV and Miami networks, respectively. Clearly the variations are different for each network.

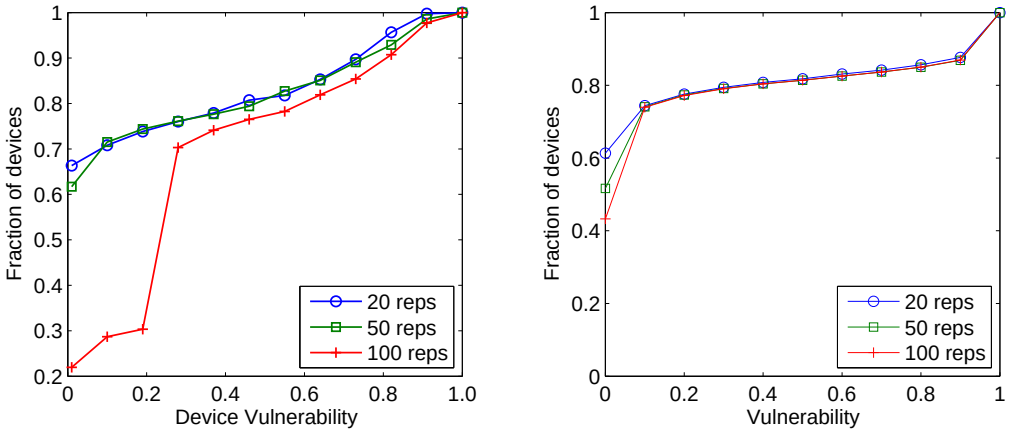
The cumulative probability distribution is shown in Figure 6.6 and indicates that as the number of replicates increases, there is a large variation in the percentage of devices that are highly vulnerable. This indicates that more replicates would yield a better understanding of the vulnerability of the device. From Figure 6.6a we can see that 75% of the devices have a vulnerability less than 0.2 for 20 and 50 iterations, while this number drops to 30% after 100 iterations. In Figure 6.6b, the variation is not that drastic. Although it is clear that the vulnerability metric varies with the number of iterations, it is not necessary that the variation is observed in all networks. It is important to identify the reasons why certain devices are vulnerable and others are not and also see if one can make general statements about the vulnerability across different networks. Next, we determine the effects of using vulnerability as a metric to identify the device to patch and observe the effect of such a response mechanism. In later sections, we make an attempt to obtain correlation of the vulnerability metric and explain the reasons why certain devices are more vulnerable than others.




(a) (b)

Figure 6.5: Device vulnerability as the number of replicates are varied. We use 20, 50, and 100 replicates to illustrate that a significant number of replicates are required for determining this metric and the number of replicates also depends on the structure of the network itself. **(a):** Probability distribution for device vulnerability on the NRV network as a function of the number of replicates used to determine the vulnerability of each node; **(b):** Probability distribution for the Miami network as the replicates are increased.

Without a scalable and expressive environment like EpiNet it is difficult to perform analysis on dynamic metrics such as vulnerability.




(a) (b)

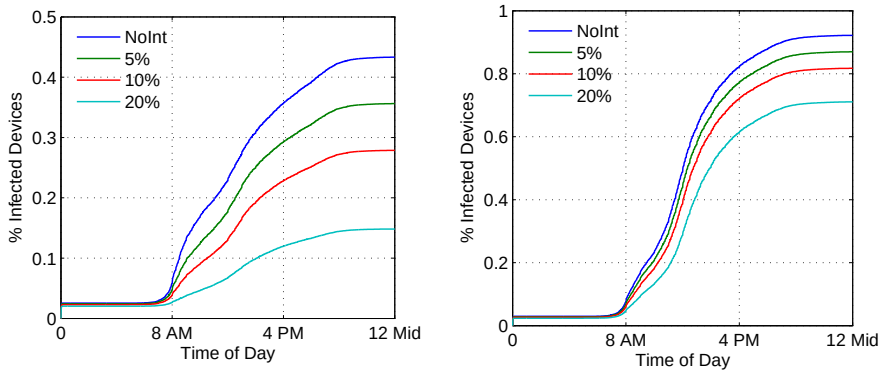
Figure 6.6: Cumulative distribution of device vulnerability as the number of replicates are varied for the NRV and Miami networks. **(a):** The cumulative distribution for device vulnerability of each device for the NRV network; **(b):** The cumulative distribution for the Miami network.

Next, we look at the convergence of the vulnerability metric on these networks. Because the vulnerability is a dynamic metric, the number of replicates required for each network varies and we need to determine whether the values have converged, or we need to consider more replicates.

### 6.3.1 Vulnerability based responses

Although there is variation in the vulnerability values for a node in different replicates, vulnerability indicates the importance of a certain device in the infection progression and thus can be used as a metric for controlling the spread of the malware. Here, we look at how effective vulnerability is in controlling the malware spread, if the metric was used to identify susceptible devices and apply targeted patching. Like degree based metrics, we need to know the network or at least conduct simulations on the network to determine if a device is vulnerable or not. Usually a significant number of replicates may be required to obtain the vulnerability irrespective of the initial infections.

In this section, we conduct these experiments for the NRV and Miami networks. The strategy is to identify the devices' vulnerability using a certain number of replicates. We use 100 replicates for each network and determine the vulnerability for each device. The devices are then ranked based on the vulnerability measured for the 100 replicates and a certain percentage of the total population (in this case, 5%, 10%, and 20%) are selected to apply the patch in the order of this rank. The highest vulnerability node is first, and so on, until the required percentage of devices has been selected.

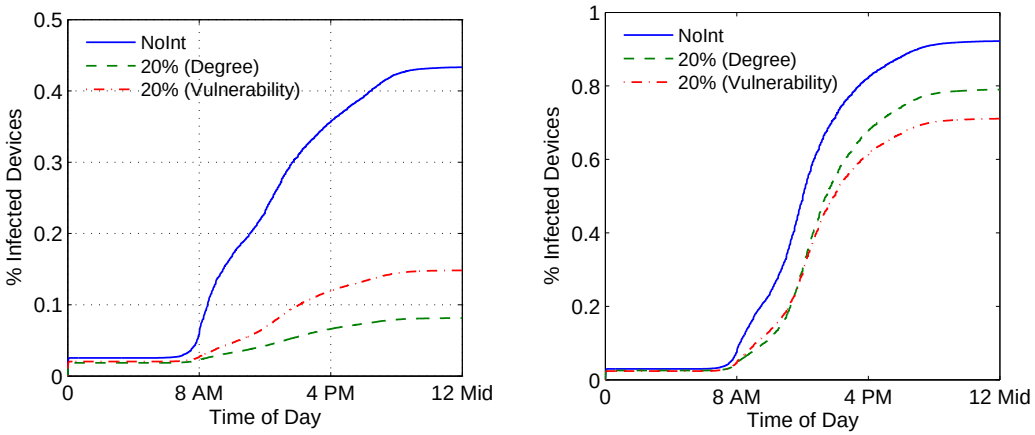



(a) (b)

Figure 6.7: Performance of the vulnerability based targeted patching of devices in NRV1 and NRV2 networks. **(a):** Patching devices based on the vulnerability rank has a very good effect on the reduction in the final infection size in the NRV1 network; **(b):** For the NRV2 network the increase in the same percentage of patched devices does not lead to increased gains. Only 20% of the devices patched devices remain uninfected at the end of the day in the NRV2 network.

Figure 6.7 shows the results we obtain for the two NRV networks. We observe in Figure 6.7a that selecting devices based on the vulnerability metric is effective in the NRV1 network and increasing the percentage of devices patched leads to increased gains. Patching slightly more than 20% of the devices actually stops the spread of the malware. The same does not apply to the NRV2 network (as shown in Figure 6.7b). The reduction in the number of final infections is not much and remains almost equal to the total percentage of the devices that are patched, i.e. 5%, 10%, and 20%, respectively.

### 6.3.2 Comparing Degree and Vulnerability Metrics



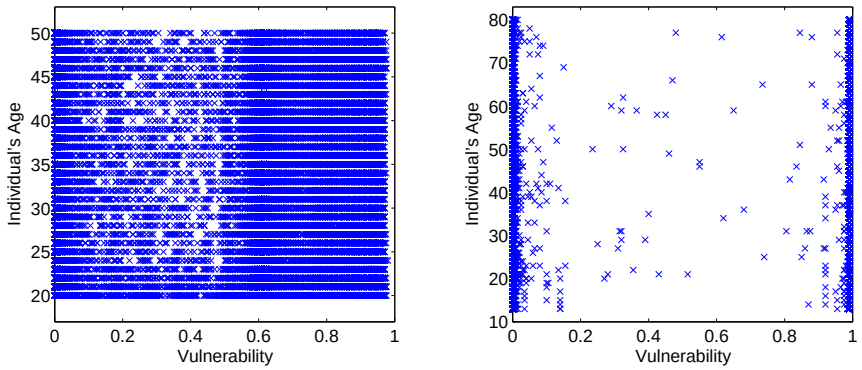

(a) (b)

Figure 6.8: Comparing degree and vulnerability metrics on the NRV networks. We find that the effect of the control mechanism is dependent on the network. Vulnerability works only slightly better in a more well connected network. **(a):** Control of malware on the NRV1 network. Selecting devices based on the degree rank is more effective than vulnerability; **(b):** NRV2 network has a slightly different outcome. Vulnerability metric fares better in controlling the spread.

Figure 6.8 shows the comparison between vulnerability and degree for the two NRV networks. It shows the difference in the effect of similar interventions on different networks. In Figure 6.8a, the effect of patching devices using degree- and vulnerability-based metrics are both effective in controlling the spread. The network they are deployed on is the NRV1 network. The same two metrics deployed on the NRV2 network indicate that the spread is not controlled effectively, as shown in Figure 6.8b. This indicates that the effectiveness of these metrics depends heavily on the network structure and it is important to have the ability to evaluate these metrics on device networks, and then deploy them on real networks. EpiNet provides an environment for performing such a detailed evaluation and analysis.

### 6.3.3 Estimating Vulnerability

According to Figure 4.8a, in the case study (presented in Section 4.3.1 of Chapter 4) we find that vulnerability is the upper bound on the effective intervention schemes. Here we are interested in investigating whether there are other metrics that are easier to obtain from than vulnerability. If some other metrics are correlated with it, we can then choose that metric and avoid trying to compute vulnerability for each network. We need to look at metrics that are dependent on the demographic metric rather than the network metrics. Network metrics such as degree would require a network to be generated to be calculated. This may not be possible. The demographic metric we look at is the age. Figure 6.9 shows the correlation between the two metrics in the NRV network. Recall that NRV1 and NRV2 contain slightly different demographics, 20-50 year olds and 13-80 year olds, respectively.




(a) (b)

Figure 6.9: Correlation between age and vulnerability obtained by the diffusion of Bluetooth malware through the proximity network. No correlation is seen in NRV1 when 13–20 and 50–80 year olds are not included. It is not possible to make a clean bifurcation of the devices into high, and low vulnerability devices for the NRV2 network.

For NRV1, there is no correlation between the vulnerability and age and almost every age value has the entire range of vulnerability values. This can also be attributed to the replicates that were used in the study. In Figure 6.10 we see vulnerability correlation of neighbors and see how the

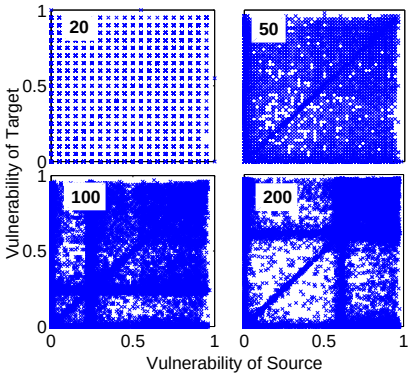



Figure 6.10: Vulnerability correlation between neighbors as the replicates are increased in the NRV1 network. The number in each subplot is the number of replicates used for the respective plot. Even after 200 replicates the correlation has not reached steady state.

correlation is changing as the number of replicates is changed. This shows that the vulnerability has not yet reached a steady state in the NRV1 network after 200 replicates. When people in the age groups of 13–20 and 50–80 are included in the population by assigning devices, the vulnerability of the people becomes higher. This is because the kids and the elderly are making the network more connected, thus dividing the nodes into highly vulnerable and not vulnerable.

In this section, we have investigated different demographic metrics and do not find a suitable metric that can replace vulnerability. Each network has a slightly different characteristic and it is extremely difficult to obtain a single metric that works on all the networks and effectively control the spread of proximity malware.

## 6.4 Summary

In this chapter, we have studied various control mechanisms for stopping the flow of the malware over realistic urban Bluetooth networks. There are several methods in the literature on applying patches to infected mobile devices. Some of them are device based detection and patch mechanisms, while others are surveillance approaches where infected devices are patched based on the surveillance information. Here we show that EpiNet environment is capable of evaluating these interventions

in addition to other new methods and evaluate the efficacy of particular schemes.

We have looked at non-adaptive responses based on static graph metrics such as degree and betweenness. We have studied device-based interventions where detection software on the device is in a position to detect a self infection and disable certain features to stop the spread. We have looked at policy-based centralized patch dissemination where the service provider applies patches to devices when the infection count reaches a certain number. We also evaluate how effective disseminating signatures in a local environment is in controlling the spread. We then evaluate the efficacy of dynamic graph metrics such as vulnerability in controlling the spread. We find that vulnerability performs well on some networks and performs better than other metrics on others. We try to find estimates for vulnerability from other metrics, but do not find any one metric that effectively maps to the vulnerability. In this chapter, we show how the framework can be employed to study aspects in the control of malware spread over mobile networks.

We find that it is very difficult to control the malware spread in a proximity network and there is no single metric that can effectively control the spread. Early detection will go a long way to show efficacy of some of the metrics and usually also requires a large fraction of the devices to be patched in order to control the spread.

# Chapter 7

# Conclusions and Future Directions

## 7.1 Summary of Contributions

Large scale simulations of malware on mobile wireless networks have become important applications of high-performance computing. In this dissertation, we have proposed EpiNet, an individual-based, scalable high performance modeling environment, to study the propagation of mobile malware. It is designed specifically to work on commodity clusters. The environment serves as a evaluation environment for studying current and future generation malware. We summarize the contributions as follows:

1. We propose and implement EpiNet a highly expressive, scalable, parallel HPC-based wireless epidemiology framework. To our knowledge, this is the only framework for malware studies that scales to large device deployments of over 100K nodes and considers realistic high-fidelity human mobility models. EpiNet has the capability to represent realistic device contact networks created as a result of human mobility based on activity patterns.

2. EpiNet exports an expressive interface to configure and study complex interventions based on static and dynamic graph metrics applied non-adaptively or adaptively. The interface also allows for applying centralized policy interventions and individualized adaptive interventions. The ability to model interventions makes it possible to evaluate the efficacy of interventions

with respect to different malware parameters.

3. We propose and construct an abstract representation of Bluetooth malware to model within-host behavior. We calibrate the model using small scale packet level simulations using network simulators. We validate this model and find that the results obtained by using this model are accurate for small networks. We construct realistic proximity networks formed by mobile devices to study the propagation of Bluetooth malware.

4. Approximations implemented in the mobility resolution provide 100X reduction in messages exchanged and corresponding reduction in communication time. Reducing the model resolution helps obtain 10X speedup in computation from an initial high resolution model. Both these approximations contribute to $< 5\%$ error in the final infection size.

5. We implement and study the dynamics of SMS/MMS malware that spread over the cellular infrastructure and study hybrid networks that can spread over Bluetooth and through SMS/MMS messages.

## 7.2 Our Key Findings

In this dissertation we have studied the dynamics of mobile malware propagation and make the following conclusions.

1. Although it is strongly believed that Bluetooth based malware cannot propagate to a lot of devices, our finding suggest the opposite. The current market share of smart phones is a contributing factor. Once a large market share is gained by smart phones, we will see an increase in the number of malware outbreaks.

2. Access to communication and information allows devices to interconnect multiple networks together – proximity networks created around the devices and social contact and communication networks, created by e-mail, IM, and SMS/MMS communications. This interconnection provides multiple pathways for a sophisticated malware and is extremely difficult to control.

We find that an aggressive hybrid malware spread to 50% of the devices within a matter of 2-3 hours.

3. Access to real data of current and previous generation malware are limited. Very little information is available in public domain, making it extremely difficult to validate our modeling approach with respect to real malware. This will help in making the modeling more realistic.

## 7.3 Future Directions

There are several directions future work in this area can take.

1. *More accurate proximity networks for Bluetooth malware.* The proximity malware we study use the high fidelity model to construct intra-location Bluetooth networks. This model can be made more accurate by taking existing data on Bluetooth networks from sources such as [1]. This would make a high resolution network within the location and can offer more accurate networks.

2. *Consider the introduction of hybrid networks.* The current simulation environment considers proximity- and infrastructure-based propagation and a combination of the two. It does not consider the impact of crossover malware that spread between PC and mobile device. This would require the consideration of hybrid networks at locations in addition to proximity networks. This would result in a completely different dynamic.

3. *Building real-time simulations of mobile malware infections.* The current framework is implemented to execute a particular scenario of malware propagation and allows for implementing dynamic intervention. Dynamic changes to these strategies may be interesting to try out where an analyst can try dynamic interventions that change as a result of how a certain interventions is playing out in controlling the spread.

# Bibliography


[1] A Community Resource for Archiving Wireless Data at Dartmouth. http://crawdad.cs.dartmouth.edu/.

[2] Scalable Simulation Framework. http://www.ssfnet.org/homePage.html.

[3] AusCERT. Australian computer crime and security survey. http://www.auscert.org.au/images/ACCSS2006.pdf, 2006.

[4] Anand Balachandran, Geoffrey M. Voelker, Paramvir Bahl, and P. Venkat Rangan. Characterizing user behavior and network performance in a public wireless lan. In *ACM SIGMETRICS*, 2002.

[5] C. L. Barrett, R. J. Beckman, K. P. Berkbigler, B. W. Bush K. R. Bisset, K. Campbell, S. Eubank, K. M. Henson, J. M. Hurford, D. A. Kubicek, M. V. Marathe, O. Ramos Jr., S. Ree, P. R. Romero, J. P. Smith, L. L. Smith, P. L. Speckman, P. E. Stretz, G. L. Thayer, E. Van Eeckhout, and M. D. Williams. TRANSIMS 2.0: Transportation Analysis Simulation System. Technical Report LA-UR-00-1723–LA-UR-00-1767, Los Alamos National Laboratory, 2000. Online.

[6] Christopher Barrett, Richard Beckman, Karthik Channakeshava, Fei Huang, VSA Kumar, Achla Marathe, Madhav Marathe, and Guanhong Pei. Cascading Failures in Multiple Infrastructures: From Transportation to Communication Network. In *In Proceedings of Interacting Critical Infrastructures for the 21st Century*, 2010.

[7] Christopher L. Barrett, Harry B. Hunt III, Madhav V. Marathe, S. S. Ravi, Daniel J.

Rosenkrantz, and Richard Edwin Stearns. Complexity of reachability problems for finite discrete dynamical systems. *J. Comput. Syst. Sci.*, 72(8):1317–1345, 2006.

[8] Christopher L. Barrett, Keith R. Bisset, Stephen G. Eubank, Xizhou Feng, and Madhav V. Marathe. EpiSimdemics: an Efficient Algorithm for Simulating the Spread of Infectious Disease over Large Realistic Social Networks. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12, 2008.

[9] R. Beckman, K. Channakeshava, Fei Huang, V.S.A. Kumar, A. Marathe, M.V. Marathe, and Guanhong Pei. Synthesis and Analysis of Spatio-Temporal Spectrum Demand Patterns: A First Principles Approach. In *New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on*, pages 1–12, April 2010.

[10] R. Beckman, K. Channakeshava, Fei Huang, V.S.A. Kumar, A. Marathe, M.V. Marathe, and Guanhong Pei. Implications of Dynamic Spectrum Access on the Efficiency of Primary Wireless Market. In *New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on*, pages 1–12, April 2010.

[11] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experience with DETER: a Testbed for Security Research. In *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*, 2006.

[12] Rob Beschizza. iPhone Game Developer Accused of Stealing Players' Phone Numbers.

[13] Beselo. SymbOS/Beselo, January 2008. http://vil.nai.com/vil/content/v_144005.htm.

[14] Christian Bettstetter. Smooth is Better than Sharp: a Random Mobility Model for Simulation of Wireless Networks. In *MSWIM '01*, pages 19–27, 2001.

[15] Christian Bettstetter, Hannes Hartenstein, and Xavier Pérez-Costa. Stochastic Properties of the Random Waypoint Mobility Model. *Wireless Networks*, 10(5):555–567, 2004.

[16] Christian Bettstetter, Michael Gyarmati, and Udo Schilcher. An Inhomogeneous Spatial Node Distribution and its Stochastic Properties. In *MSWiM '07: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, pages 400–404, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-851-0. doi: http://doi.acm.org/10.1145/1298126.1298195.

[17] R. Bo and S. Lowen. Fractal traffic models for internet simulation. In *IEEE Computers and Communications (ISCC)*, 2000.

[18] Abhijit Bose and Kang G. Shin. Proactive Security for Mobile Messaging Networks. In *WiSe '06: Proceedings of the 5th ACM Workshop on Wireless Security*, pages 95–104, New York, NY, USA, 2006. ACM.

[19] Abhijit Bose, Xin Hu, Kang G. Shin, and Taejoon Park. Behavioral Detection of Malware on Mobile Handsets. In *MobiSys '08: Proceeding of the 6th international conference on Mobile Systems, Applications, and Services*, pages 225–238. ACM, 2008.

[20] Liane Cassavoy. iPhone Security Flaw: Separating Fact from Fiction. PCWorld, August 2010. http://www.pcworld.com/article/202538/iphone_security_flaw_separating_fact_from_fiction.html.

[21] Center for Disease Control. National Health Interview Survey (NHIS). http://www.cdc.gov/nchs/about/major/nhis/nhis_2007_data_release.htm.

[22] K. Channakeshava, D. Chafekar, K. Bisset, V. S. Anil Kumar, and M. Marathe. EpiNet:A Simulation Framework to Study the Spread of Malware in Wireless Networks. In *In Proceedings 2nd International Conference on Simulation Tools and Techniques (SIMUTools'09)*, March 2009.

[23] K. Channakeshava, K. Bisset, V.S.A. Kumar, M. Marathe, and S. Yardi. High Performance Scalable and Expressive Modeling Environment to Study Mobile Malware in Large Dynamic Networks. In *25th IEEE International Parallel & Distributed Processing Symposium (IEEE IPDPS 2011)*, 2011. Sumbitted.

[24] Credant Technologies. Survey demonstrates poor smartphone security attitude amongst users. http://www.prosecurityzone.com/Customisation/News/IT_Security/Mobile_computing_security/Survey_demonstrates_poor_smartphone_security_attitude_amongst_users.asp, June 2009. Online.

[25] Drever. Drever.C. http://www.f-secure.com/v-descs/drever_c.shtml, March 2005.

[26] A. Dwivedi and R. Wagner. Traffic Model for USA Long-distance Optical Network. In *IEEE Optical Fiber Communication Conference*, 2002.

[27] Daniel R. Ellis, John G. Aiken, Kira S. Attwood, and Scott D. Tenaglia. A Behavioral Approach to Worm Detection. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 43–53, New York, NY, USA, 2004. ACM. ISBN 1-58113-970-5. doi: http://doi.acm.org/10.1145/1029618.1029625.

[28] S. Eubank, V.S. Anil Kumar, M. Marathe, A. Srinivasan, and N. Wang. *Structure of Social Contact Networks, and Their Impact on Epidemics*. AMS-DIMACS Special Issue on Epidemiology, 2006.

[29] P. Ferrie, P. Szor, R. Stanev, and R. Mouritzen. Security Responses: Symbos.cabir. Symantec Corporation, June 2004.

[30] Kevin Finisterre. InqTana Through the Eyes of Dr. Frankenstein, 2006. http://www.digitalmunition.com/InqTanaThroughTheEyes.txt.

[31] Chris Fleizach, Michael Liljenstam, Per Johansson, Geoffrey M. Voelker, and Andras Mehes. Can you infect me now?: Malware Propagation in Mobile Phone Networks. In *WORM '07: Proceedings of the 2007 ACM workshop on Recurring malcode*, pages 61–68, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-886-2. doi: http://doi.acm.org/10.1145/1314389.1314402.

[32] Sally Floyd and Vern Paxson. Difficulties in Simulating the Internet. *IEEE/ACM Transactions on Networking*, 2001.

[33] Stephanie Forrest, Steven Hofmeyr, and Anil Somayaji. The Evolution of System-Call Monitoring. In *ACSAC '08: Proceedings of the 2008 Annual Computer Security Applications Conference*, pages 418–430. IEEE Computer Society, 2008.

[34] Priya Ganapati. Malware Sneaks Into Android Market, January 2010. http://www.wired.com/gadgetlab/2010/01/android-malware-fears/.

[35] Gartner, Inc. Gartner Says Worldwide Mobile Phone Sales Increased 16 Per Cent in 2007. http://www.gartner.com/it/page.jsp?id=612207, February 2008. Press Release.

[36] Gartner, Inc. Gartner Identifies the Top 10 Strategic Technologies for 2010. http://www.gartner.com/it/page.jsp?id=1210613, October 2009. Press Release.

[37] Georgia Tech Information Security Center. Emerging Cyber Threats Report for 2009. http://www.gtiscsecuritysummit.com/pdf/CyberThreatsReport2009.pdf, 2009.

[38] Georgia Tech Information Security Center. Emerging Cyber Threats Report for 2009. http://www.gtiscsecuritysummit.com/pdf/CyberThreatsReport2009.pdf, 2009.

[39] Daniel T. Gillespie. A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *Journal of Computational Physics*, 22(4):403 – 434, 1976.

[40] Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. Understanding Individual Human Mobility Patterns. *Nature*, 453(7196):779–782, June 2008. ISSN 0028-0836. URL http://dx.doi.org/10.1038/nature06958.

[41] GTNetS. The Georgia Tech Network Simulator (GTNetS). http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/.

[42] Mikko Hypponen. Malware Goes Mobile. *Scientific American*, November 2006.

[43] Gianluca Iannaccone, Christophe Diot, Derek McAuley, Andrew Moore, Ian Pratt, and Luigi Rizzo. The CoMo White Paper. Technical report, Intel Research, September 2004.

[44] Grégoire Jacob, Hervé Debar, and Eric Filiol. Behavioral Detection of Malware: from a Survey Towards an Established Taxonomy. *Journal in Computer Virology*, 4(3):251–266, 08 2008.

[45] Markus Jakobsson and Karl-Anders Johansson. Retroactive Detection of Malware With Applications to Mobile Platforms. In *HotSec 2010*. USENIX, USENIX, August 2010.

[46] Markus Jakobsson and Ari Juels. Server-Side Detection of Malware Infection. In *New Security Paradigms Workshop (NSPW)*. ACM, September 2009.

[47] Ryan Kairer. Smartphones Showed Strong Growth in 2007, February 2008. http://www.palminfocenter.com/news/9617/smartphones-showed-strong-growth-in-2007.

[48] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido. A Nonstationary Poisson View of Internet Traffic. In *INFOCOM '04*, 2004.

[49] Brendan P. Kehoe. *Zen and the Art of the Internet*. 1992. http://www.cs.indiana.edu/docproject/zen/zen-1.0_toc.html.

[50] Jeffrey O. Kephart and Steve R. White. Directed-Graph Epidemiological Models of Computer Viruses. *Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on*, 00:343–359, May 1991.

[51] Jeffrey O. Kephart and Steve R. White. Measuring and Modeling Computer Virus Prevalence. In *SP '93: Proceedings of the 1993 IEEE Symposium on Security and Privacy*, page 2, Washington, DC, USA, 1993. IEEE Computer Society.

[52] Jeffrey O. Kephart, Steve R. White, and David M. Chess. Computers and Epidemiology. *IEEE Spectrum*, 30(5):20–26, 1993. ISSN 0018-9235. doi: http://dx.doi.org/10.1109/6.275061.

[53] M.H. R. Khouzani, Saswati Sarkar, and Eitan Altman. Optimal Propagation of Security Patches in Mobile Wireless Networks: Extended Abstract. *SIGMETRICS Perform. Eval. Rev.*, 38(1):355–356, 2010.

[54] Hahnsang Kim, Joshua Smith, and Kang G. Shin. Detecting Energy-greedy Anomalies and Mobile Malware Variants. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 239–252, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-139-2. doi: http://doi.acm.org/10.1145/1378600.1378627.

[55] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. *Wireless Networks*, 2005.

[56] L. Kroc, S. Eidenbenz, and J. Smith. SessionSim: Activity-Based Session Generation for Network Simulation. In *Winter Simulation Conference*, 2009.

[57] M. Lactaotao. Security Information: Virus Encyclopedia: Symbos comwar.a. Trend Micro Incorporated, March 2005.

[58] Lasco. Cabir!lasco. http://vil.nai.com/vil/content/v_131604.htm, January 2005.

[59] George Lawton. Is It Finally Time to Worry about Mobile Malware? *Computer*, 41(5):12–14, 2008. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/MC.2008.159.

[60] Rafal Leszczyna, Igor Nai Fovino, and Marcelo Masera. Simulating malware with malsim. *Journal in Computer Virology*, 6:65–75, 2010.

[61] Liberty. Palm/Liberty. http://www.f-secure.com/v-descs/lib_palm.shtml, August 2000.

[62] Michael Liljenstam, David M. Nicol, Vincent H. Berk, and Robert S. Gray. Simulating Realistic Network Worm Traffic for Worm Warning System Design and Testing. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 24–33, New York, NY, USA, 2003. ACM.

[63] J. Ma, G. Voelker, and S. Savage. Self-stopping Worms. In *WORM '05*, 2005.

[64] Mabir. Mabir.a!sis. http://vil.nai.com/vil/content/v_132804.htm, April 2005.

[65] P. Mel, K. Kent, and J. Nusbaum. Guide to Malware Incident Prevention and Handling. http://csrc.nist.gov/publications/nistpubs/800-83/SP800-83.pdf, November 2005. NIST 800-83.

[66] Ellen Messmer. New Approaches to Malware Detection Coming into View. http://www.networkworld.com/news/2007/042507-malware-detection.html?page=1, April 2007. Online.

[67] James W. Mickens and Brian D. Noble. Modeling Epidemic Spreading in Mobile Environments. In *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*, pages 77–86. ACM, 2005.

[68] James W. Mickens and Brian D. Noble. Analytical Models for Epidemics in Mobile Networks. In *Wireless and Mobile Computing, Networking and Communication, IEEE International Conference on*, pages 77–84, Los Alamitos, CA, USA, 2007. IEEE Computer Society. ISBN 0-7695-2889-9.

[69] David Moore, Colleen Shannon, and k claffy. Code-Red: A Case Study on the Spread and Victims of an Internet Worm. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 273–284, New York, NY, USA, 2002. ACM. ISBN 1-58113-603-X. doi: http://doi.acm.org/10.1145/637201.637244.

[70] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the Slammer Worm. *IEEE Security and Privacy*, 1(4):33–39, 2003. ISSN 1540-7993. doi: http://dx.doi.org/10.1109/MSECP.2003.1219056.

[71] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring Internet Denial-of-Service Activity. *ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139, 2006.

[72] Susanta Nanda and Tzi-Cker Chiueh. Execution Trace-Driven Automated Attack Signature Generation. In *ACSAC '08: Proceedings of the 2008 Annual Computer Security Applications Conference*, pages 195–204. IEEE Computer Society, 2008.

[73] Maziar Nekovee. Worm Epidemics in Wireless Ad Hoc Networks. *New Journal of Physics*, 9 (6):189, 2007. URL http://stacks.iop.org/1367-2630/9/189.

[74] Maziar Nekovee. Epidemic Spreading of Computer Worms in Fixed Wireless Networks. *Bio-Inspired Computing and Communication*, pages 105–115, 2008. URL http://dx.doi.org/10.1007/978-3-540-92191-2_10.

[75] Zoran Nikoloski, Narsingh Deo, and Ludek Kucera. Correlation Model of Worm Propagation on Scale-Free Networks. *Complexus*, pages 169–182, March 2006.

[76] NS-2. Network Simulator (NS-2). http://www.isi.edu/nsnam/ns/.

[77] Terrence OConnor and Douglas Reeves. Bluetooth Network-Based Misuse Detection. In *ACSAC '08: Proceedings of the 2008 Annual Computer Security Applications Conference*, pages 377–391. IEEE Computer Society, 2008.

[78] V Paxson and S Floyd. Wide Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 1995.

[79] Cyrus Peikari. Analyzing the Crossover Virus: The First PC to Windows Handheld Cross-infector. http://www.informit.com/articles/article.aspx?p=458169&seqNum=5, March 2006.

[80] Lisa Phifer. Defeating Malicious Mobiles. *Business Communications Review*, April 2007.

[81] Redbrowser. Redbrowser.A. http://www.f-secure.com/v-descs/redbrowser_a.shtml, March 2006.

[82] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, and Song Chong. On the Levy-Walk Nature of Human Mobility. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 924–932, April 2009.

[83] C.J. Rhodes and M. Nekovee. The Opportunistic Transmission of Wireless Worms between Mobile Devices. *Physica A: Statistical Mechanics and its Applications*, 387(27):6837–6844, 2008.

[84] E. Van Ruitenbeek, T. Courtney, W. H. Sanders, and F. Stevens. Quantifying the Effectiveness of Mobile Phone Virus Response Mechanisms. In *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pages 790–800, 2007.

[85] Scalable Network Technologies. QualNet. http://www.scalable-networks.com/.

[86] Kirk Schloegel, George Karypis, and Vipin Kumar. Parallel Multilevel Algorithms for Multi-constraint Graph Partitioning. In *Euro-Par '00*, pages 296–310. Springer-Verlag, 2000.

[87] Mukund Seshadri, Sridhar Machiraju, Ashwin Sridharan, Jean Bolot, Christos Faloutsos, and Jure Leskove. Mobile Call Graphs: Beyond Power-law and Lognormal Distributions. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 596–604. ACM, 2008.

[88] Skulls. Skulls.A. http://www.f-secure.com/v-descs/skulls.shtml, March 2005.

[89] Jing Su, Kelvin K. W. Chan, Andrew G. Miklas, Kenneth Po, Ali Akhavan, Stefan Saroiu, Eyal de Lara, and Ashvin Goel. A Preliminary Investigation of Worm Infections in a Bluetooth Environment. In *WORM '06: Proceedings of the 4th ACM Workshop on Recurring Malcode*, pages 9–16, 2006.

[90] Symantec. SymbOS.Cardtrp.A. http://www.symantec.com/security_response/writeup.jsp?docid=2005-092215-2634-99, September 2005.

[91] D. Tang and M. Baker. Analysis of a Local-Area Wireless Network. In *ACM MobiCom*, pages 1–10, 2000.

[92] John Tang, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Vincenzo Nicosia. Analyzing Information Flows and Key Mediators Through Temporal Centrality Metrics. In *SNS '10*, pages 1–6. ACM, 2010.

[93] UCBT. UCBT: Bluetooth Extension for NS2 at University of Cincinnatti. http://www.ececs.uc.edu/~cdmc/ucbt/.

[94] Pu Wang, Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. Understanding the Spreading Patterns of Mobile Phone Viruses. *Science*, pages 1167053–, April 2009.

[95] Songjie Wei, Jelena Mirkovic, and Martin Swany. Distributed Worm Simulation with a Realistic Internet Model. In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, pages 71–79. IEEE Computer Society, 2005. ISBN 0-7695-2383-8.

[96] Songjie Wei, Alefiya Hussain, Jelena Mirkovic, and Calvin Ko. Tools for Worm Experimentation on the DETER Testbed. *Int. J. Commun. Netw. Distrib. Syst.*, 5(1/2):151–171, 2010.

[97] D. Willkomm, S. Machiraju, J. Bolot, and A. Wolisz. Primary Users in Cellular Networks: A Large-Scale Measurement Study. In *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*, pages 1–11, October 2008.

[98] Guanhua Yan and Stephan Eidenbenz. Bluetooth Worms: Models, Dynamics, and Defense Implications. In *ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference*, pages 245–256. IEEE Computer Society, 2006.

[99] Guanhua Yan and Stephan Eidenbenz. Modeling Propagation Dynamics of Bluetooth Worms. In *ICDCS*, pages 42–52. IEEE Computer Society, 2007.

[100] Guanhua Yan, Hector D. Flores, Leticia Cuellar, Nicolas Hengartner, Stephan Eidenbenz, and Vincent Vu. Bluetooth Worm Propagation: Mobility Pattern Matters! In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, Computer and Communications Security*, pages 32–44. ACM, 2007.

[101] Cliff Changchun Zou, Weibo Gong, and Don Towsley. Code Red Worm Propagation Modeling and Analysis. In *CCS '02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 138–147. ACM, 2002.

[102] Cliff Changchun Zou, Weibo Gong, and Don Towsley. Worm propagation modeling and

analysis under dynamic quarantine defense. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 51–60. ACM, 2003.

[103] Gjergji Zyba, Geoffrey M. Voelker, Michael Liljenstam, Andras Mehes, and Per Johansson. Defending Mobile Phones from Proximity Malware. In *INFOCOM 2009. The 27th Conference on Computer Communications. IEEE*, April 2009.