DATA DISSEMINATION PROTOCOLS IN WIRELESS SENSOR NETWORKS : DESIGN, MODELING AND SECURITY

by

PRADIP DE

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2008

To my parents

and

Srirupa...

ACKNOWLEDGEMENTS

I am deeply indebted to my supervisors Prof. Sajal K. Das and Dr. Yonghe Liu for their expertise, motivation, constant encouragement, advice and all the persistent help and guidance that carved the path towards the completion of my dissertation. I would also like to extend my heartfelt gratitude to Prof Kalyan Basu for his insightful advice and suggestions. I acknowledge the support of my other committee members, Dr. Mohan Kumar and Dr. Matthew Wright, for their interest in my research and taking time to serve in my committee. Their invaluable comments and suggestions helped improve my dissertation quality immensely.

I would also like to extend my appreciation to TxTec funding office, Nokia, the Dean's office, and the CSE department at UT Arlington for providing the necessary financial support for my graduate studies. I am thankful to all the professors at UT Arlington under whom I took various courses that helped me equip myself with the much needed knowledge to pursue my doctoral studies.

Finally, I would like to express my deepest gratitude to my wife, Srirupa, whose love, perseverence, and patience have been a constant source of sustenance and inspiration for me. I am indebted to all my family members back in India for their enormous support and love that went a long way in sustaining me through my doctoral studies. Last, but definitely not the least, I thank my friends and fellow members at CReWMaN for those unforgettable moments of fun and camaraderie that we shared.

April 9, 2008

ABSTRACT

DATA DISSEMINATION PROTOCOLS IN WIRELESS SENSOR NETWORKS : DESIGN, MODELING AND SECURITY

Pradip De, Ph.D.

The University of Texas at Arlington, 2008

Supervising Professors : Sajal K. Das and Yonghe Liu

Wireless sensor and actuator networks have been one of the stepping stones towards realizing a pervasive computing infrastructure. However, in a post-deployment scenario, transferring critical updates and reconfigurations on a network-wide scale is a non-trivial proposition. Wireless techniques provide the only means of communication, and consequently, an in-depth study of the multihop broadcast based communication paradigm and the design of efficient and reliable data dissemination protocols for reprogramming a sensor network is of paramount importance.

In this dissertation, we initially focus on the formal modeling and performance analysis of broadcast-based data dissemination protocols in wireless sensor networks. The data propagating in the network could be either small configuration information to be shared by all the nodes or a large code image required for reprogramming the network. In order to better understand the propagation behavior, we construct a mathematical model that allows us to compare different dissemination protocols in terms of their speed of propagation and extent of network reachability.

Next, from a perspective of security, we investigate the potentially disastrous threat of node compromise originating from a single node infected with a piece of

malware, propagating to other nodes and gradually compromising the entire sensor network. Focusing on the possible epidemic breakout of such a propagation, we model and analyze this spreading process and identify key factors determining potential outbreaks. More importantly, we compare the propagation process based on different sensor deployment strategies, for instance, uniform and group-based deployment, thereby getting valuable insights for designing secure networks. Subsequently, we delve onto a specific case of a malware spreading over existing data dissemination protocols in sensor networks. In order to better understand these protocols' vulnerability to piggybacked virus attacks, we construct a mathematical model for the malware infection, incorporating important parameters derived from the communication patterns of the protocol under test. We further enrich our study by observing the effects of a simultaneous recovery process on the infection propagation. The overall result is an approximate but flexible framework to characterize a broadcast protocol in terms of its vulnerability to malware propagation.

Having focused on analyzing data dissemination techniques in static sensor networks, we observe that existing data dissemination protocols are inefficient in a mobile environment and require effective modifications to suit the uncertainties and demands of network mobility. Thus, we propose a novel wireless multihop data dissemination protocol, suitable to a mobile sensor network and evaluate it through extensive simulations as well as real testbed implementation on a network of SunSPOT devices. Our results indicate an improved performance of our protocol over existing reprogramming protocols, both in terms of completion time and total number of messages transmitted in the network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Wireless sensor networks have gained immense popularity in the last few years as arguably the first means of realizing a distributed and networked pervasive computing environment. These networks, composed of compact, inexpensive, battery-operated sensor units equipped with computational and wireless communication capabilities [38, 69] are the result of the symbiosis of ubiquitous RF-based wireless networking [88] and recent advances in low-power analog and digital electronics. Due to their increasingly favorable form and cost factors, it is feasible to connect together a large number of such sensors in order to support fault-tolerant, fine-grained monitoring of the physical environment and tracking applications [67]. Cheap and ubiquitous platforms of networked sensors will be the key to real-time delivery of large volumes of useful information and would support a variety of applications such as battlefield monitoring, structural health monitoring, intrusion detection and physical environment monitoring.

1.1 Scope and Challenges

Many of these applications require sensors to be deployed in a large scale in a fairly dense manner. Moreover, in most of these cases, the devices are deployed in places which are seldom physically accessible and thus expensive and time consuming to be regularly monitored by humans. Thus, a post-deployment control and management of the network as a whole is a critical issue. These networks require suitable protocols that would achieve this remotely and in a reliable, scalable and

efficient manner. Also, one cannot overlook the security issue associated with such a management and control method.

An important feature of a wireless sensor network is that the network is generally treated as a whole or as clusters of nodes. Individual nodes are relatively insignificant, and this prompts redundant deployments in most applications. Consequently, sensor nodes are seldom required to be individually addressible. Thus, the most important data flows in sensor networks is generally of the form of *convergecast* towards the base station or broadcast/multicast to many or all the nodes of the network. We thus observe that broadcast-based wireless multihop propagation of information is a very important paradigm of communication in a sensor network. Due to the necessity of wirelessly retasking or reprogramming the nodes of a sensor network in the post-deployment phase, several over-the-air code update protcols have been designed [36], [53], [49]. These protocols can be used to update all the nodes of the network with newer versions of installed code or to share useful information from a single node to the entire network in a wireless multihop manner. Not only are these protocols useful for retasking the whole network, but they also help in remotely debugging deployed nodes. Reliably disseminating a piece of data to every node is a fundamental primitive in wireless sensor networks. Apart from reprogramming the entire network, example uses include disseminating a pattern for in-network event detection, a communication schedule for radio duty cycling, or configuration constants for tuning operations.

Because wireless network-wide data dissemination is a critical requirement, security of this mode of communication in a sensor network is of prime concern. However, it remains as one of the most critical challenges yet to be fully addressed. Due to the unique characteristics of wireless sensors, such as their scarce resources and hence low defense capabilities, node compromises can be expected to be common phenomena for wireless sensor networks in unattended and remote, hostile environments.

Extensive research efforts have been conducted towards designing resilient network security mechanisms [11], [26], [13], [54]. However, the compromise itself, and in particular, the propagation of node compromise (possible epidemics) have attracted little attention. While node compromise, thanks to physical capture and succeeding analysis, is naturally constrained by the adversary's capabilities, software-originated compromises can be much more damaging. Recent emergence of viruses like Cabir[1] that spread over wireless interfaces in cellular networks indicate that wireless sensor networks are also extremely vulnerable to malware. It can be conjectured that, wireless sensor networks, with their inherent properties of high densities, large-scale deployments and data spreading nature, coupled with the fact that they are generally deployed in mostly unattended terrains, are undoubtedly vulnerable to possible virus/malware outbreaks spreading over the 802.15.4 interface.

The fact that it would be difficult to continuously perform human monitoring of a deployed network makes it even more vulnerable. Moreover, the hardware constraints could also restrict the use of sophisticated mechanisms like antivirus installations or complex cryptographic measures to fight virus attacks and, more disturbingly, attacks which might seem apparently simple and harmless in a conventional network might prove detrimental in the case of a sensor network. For instance, from the sensor network's standpoint, a piece of software that can spread across multiple nodes and repeatedly perform complex operations that deplete battery, can be malicious.

In particular, the broadcast or code dissemination protocols proposed for wirelessly reprogramming a sensor network can easily serve as vehicles in transfering a piece of malware across the whole network very quickly. Thus, malware that does not have the ability to transfer itself to other nodes can exploit these protocols by piggybacking atop them and transferring themselves to the whole network. The density

[1] http://www.f-secure.com/v-descs/cabir.shtml

and large scale nature of wireless sensor networks would only make matters worse and facilitate fast malware propagation over these protocols.

Malware spreading over the Internet has been widely studied, and notably by means of epidemic theory [3]. However, the marked difference in the topological aspects of the Internet with that of a sensor network often render existing models for the former unusable in the latter. The Internet is typically characterized as a scale free network showing the properties of *preferential attachment* and *growth*, both of which are largely missing in wireless sensor networks. Although there are several algorithms and protocols for data dissemination and routing in sensor network that are based on epidemic principles [9], a consolidated formal model to quantify the propagation rate and other important parameters is yet to be designed.

1.2 Contributions

A deep and insightful study of the behavior of the broadcast based communication paradigm in sensor networks is essential towards not only understanding the information propagation processes, but also to assess their shortcomings and vulnerabilities to different kinds of security breaches. This analytical study, as we perform in this dissertation, would equip us with better insight into building efficient, robust and secure data dissemination protocols for sensor networks.

This dissertation seeks to analyze broadcast based information propagation in sensor networks under different application scenarios and different deployment strategies, focusing on both static and mobile sensor networks. We construct models for the propagation process under each scenario and analyze those models in the light of the extent of coverage and the speed of propagation of the information, be it useful data/code or harmful malware.

We now describe the contributions of this dissertation.

1.2.1 Analytical Model for Performance Analyses of Data Dissemination Protocols

Several protocols for code update or data dissemination in sensor networks have been proposed in the recent literature including Trickle [53], Firecracker [52], Deluge [36] or MNP [49]. We construct a mathematical model to analyze the performance of different data dissemination protocols in sensor networks in terms of speed of propagation and coverage of the network. Specifically, our contribution is a novel framework based on epidemic theory [3], which serves as a common and flexible platform for capturing and characterizing the spread of information over the different protocols (e.g., Trickle, Deluge. etc.), thus facilitating a comparative analysis of their performance. The results provide valuable insights toward designing better protocols that can be used for disseminating executable code to the entire network and other network management tasks. We construct our epidemic model for data propagation based on the local spatial interaction of nodes in a neighborhood. Then we map each broadcast protocol onto this model by expressing an important parameter of this epidemic framework, viz., the infection rate, as a function of the communication rate of the protocol under test after incorporating the physical communication constraints of the wireless network. Subsequently, we use this rate in our epidemic model to observe the dynamics of the information propagation over the particular protocol. This framework, thus provides a flexible analytical tool by which we can assess, within close approximations, the performance of different data dissemination protocols by expressing their communication rates in terms of the infection rate of the underlying epidemic framework. Our results indicate that our model can fairly approximate the data propagation behavior of different broadcast protocols accounting for their speed of transfer and network coverage.

1.2.2 Analysis of Node Compromise Propagation

Having developed an epidemic model for wireless sensor networks performed a performance analysis of different broadcast protocols, we next focus on an investigation of the spreading process of *node compromise* in a large scale sensor network [18]. We assume that the sensor network uses secure communications framework. Nodes use pre-distributed shared secret keys to securely communicate among themselves. Compromise of a node could veritably reveal all the keys used by that particular node. Starting from a single compromised node, we assume that the adversary can effectively compromise some of its neighboring nodes through wireless communication and thus can threat the whole network without engaging in large-scale physical attacks. In particular, due to security schemes employed by the sensor networks, we assume that communication can only be performed when neighboring nodes can establish mutual trust by authenticating using a common key. Therefore, node compromise is not only determined by the deployment of sensor nodes which in turn affects node density, but also determined by the pairwise key distribution scheme employed therein.

By incorporating these factors of the networks, we propose an epidemiological model to investigate the probability of a network-wide breakout (compromise of the whole network) and if not, the sizes of the affected components (compromised clusters of nodes). Furthermore, we analyze the effects of node recovery in an active infection scenario and obtain critical values for these parameters that result in an outbreak. We focus our analysis on two specific types of node deployment scenarios, namely *uniform random deployment* and *group-based deployment* of nodes. In the latter, locations of the nodes of a group are assumed to follow a particular spatial distribution about the *group deployment point*.

Through extensive simulations, we show that our analytical results can closely capture the effects for a wide range of network setups. Our analysis also provides deeper insights on the temporal dynamics of the epidemic process under each deployment scenario.

We, subsequently, move on to analyze the special case of the threat of node compromise spreading over data dissemination protocols in sensor networks which are particularly vulnerable. As much as these protocols provide reconfigurability to the sensor network, they could also serve as easy vehicles for any malware to propagate through the entire network. This way, the malware need not devise any strategy to target any specific node based on some knowledge of the topology, but simply rely on the dissemination protocol's propagation mechanism to transport itself to the entire network. In particular, we look at a scenario where a source node has been compromised and is being used alongwith the communication mechanism of the broadcast protocol to compromise the rest of the nodes by propagating a piece of malware to the entire network.

A set of recent works have focused on authentication mechanisms for securing broadcast protocols, particular for code-update [27, 24, 51], using a combination of hash trees or hashchain based schemes and digital signatures. Generally, they require the first packet of the data-stream to be digitally signed at the source and verified by nodes along the propagation paths. The rest of the packets are either secured by a hash chain or a hash tree over the entire set of subsequent packets.

However, the complexity of the security mechanism may increase significantly in the presence of multiple sinks or base stations [78, 24] with hash trees being generated for each one. Moreover, if the source of a network-wide broadcast is a regular sensor node, then it might be prohibitive to use digital signatures as it would have to perform both signing and verification functions. More importantly, if a regular sensor node indeed serves as a broadcast source using digital signatures for authentication,

its undetected compromise could reveal all the security related information to the attacker rendering the broadcast protocol vulnerable. Consequently, an adversary, on capturing that node, can employ the protocol as a vehicle for propagating malicious code.

We use an epidemic theoretic framework similar to the one used previously for comparing the performance of these dissemination protocols, to analyze their vulnerability [19, 20]. However, in addition, we also consider a simultaneous recovery process that is active in the network and assess the comparative vulnerabilities of each protocol even under the influence of a simultaneous recovery. We thus arrive at critical values pointing at, not only the rate of infection of the network, but also the maximum fraction of the network compromised under each protocol before the recovery procedure is able to take over the malware breakout.

1.2.3 Design of a Reprogramming Protocol For Mobile Sensor Networks

Having analyzed the performance and security aspects of data dissemination protocols in *static* sensor networks, we observe that there is no suitable protocol for efficiently disseminating data in a mobile sensor network. Mobility of nodes in a network presents significant uncertainty around critical factors like node locations and neighborhood densities. Existing broadcast protocols, if used as-is in a mobile environment, would function inefficiently in terms of both speed and energy savings. We thus propose a new reprogramming protocol for mobile sensor networks.

Our dissemination mechanism primarily revolves around periodic advertisements of code metadata in the neighborhood. When a conflict is discovered, the code pages are propagated in an ordered manner across the network in a pipelined, spatially multiplexed fashion. The reason behind maintaining page order during pipelined transfer in a static network is the notion that the code is propagating from a source in the form of a wave. Therefore, in the absence of any kind of unicast

routing, there is very low probability of pages arriving at a node out of order. Thus, these protocols reduce the overhead of nodes contending for different pages by enforcing an ordered transfer. However, the random and continuous mobility of the nodes could render the feature of acquiring pages in-order, as the existing protocols for static sensor networks are designed, extremely inefficient. Furthermore, node mobility causes considerable uncertainty on the location of a node at any given time. Thus, a code update protocol running at a node should not only consider the issue of optimally utilizing the node's resources, but also engage in tackling the uncertainties of neighboring nodes' locations brought about by their mobility.

We introduce *ReMo* [21], a reprogramming protocol specifically designed for mobile sensor networks. ReMo tries to infer relative distance with neighboring nodes and the link qualities with them from parameters measured from received packets. In contrast with previous reprogramming protocols, in ReMo, we relax the constraint of in-order propagation of pages and try to take advantage of mobility by allowing nodes to download pages out-of-order. We address two main issues for code exchange between neighbors. First, nodes try to ascertain which neighbor has the best link quality as well as a higher chance of staying within communication range for a sufficient duration. Second, a node also tries to ascertain which neighbor has the highest potential for providing pages for download. The primary local goals of ReMo running at a node are to optimally choose a neighbor based on the above two requirements while simultaneously trying to minimize its energy usage by intelligently limiting the number of control messages transmitted in a neighborhood. Since each node takes local decisions based only on neighborhood information, ReMo can scale to large network sizes. Moreover, the optimal choice of a neighbor to exchange pages with, based on link quality and potential for page exchange, helps it extract the most out of the uncertain mobile environment. This means it is efficient in terms of speed of propagation and energy savings. Furthermore, the epidemic style of page propagation

automatically achieves the desired reliability to correctly disseminate the code to the entire network.

We have conducted extensive simulations to evaluate ReMo against other existing protocols like ones proposed for static networks [36, 49]. Our results, under different mobility scenarios, show significant improvement of ReMo both in terms of time taken to reprogram the network and also the number of messages transmitted to achieve the dissemination.

We have also implemented ReMo [22] on a testbed of SunSPOT sensors [86] and evaluated its performance in comparison with Deluge [36].

1.3 Organization of Dissertation

In chapter 2, we review existing literature and provide some necessary background on a few related topics that have been used in this dissertation. In chapter 3, we focus on a comparative performance analysis of different data dissemination protocols from a data propagation and network reachability standpoint. In chapter 4, we delve into the analysis of node compromise propagation in a securely communicating sensor network and model and study the effects of different deployment strategies on this process. We further analyze the behavior of malware propagation over different data dissemination protocols in chapter 5. In chapter 6, we identify the drawbacks of existing dissemination protocols when applied in a mobile scenario and propose a novel reprogramming protocol for mobile sensor networks and discuss its design. We elaborate on the implementation of this protocol in chapter 7 and finally conclude this dissertation in chapter 8.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we discuss existing works related to this dissertation. In chapter 3, we discuss the performance analysis of several dissemination protocols based on a common framework of evaluation. Many dissemination protocols have been proposed in the literature. Data dissemination protocols in sensor networks can be broadly classified as the category of protocols that aid the process of data transfer from a single or a small number of sensor nodes to the rest of the network or a significant fraction of the network. These protocols are critical for various purposes where various levels of control and management is necessary over a sensor network for operations ranging from remotely upgrading software on the nodes to troubleshooting them and fixing bugs. The necessity for this capability is primarily accentuated by the fact that sensors in their post-deployment phase are generally assumed to be physically inaccessible. This assumption automatically applies to sensors deployed in irrigation fields, military grounds, etc. However, they also apply to a pervasive computing environment where the sensors are embedded in our environment and, although accessible, it is an extremely expensive proposition to uproot them from the environment for purposes of upgrading them or changing applications running on them.

One of the prominent works of data dissemination in sensor networks is SPIN [47, 46] where the authors proposed the concept of meta data or data descriptors to eliminate the chance of redundant transmissions in sensor networks. Their main contribution was based on the basic deficiencies of classic flooding, viz., *Implosion*, *Overlap*, and *Resource Blindness*.

For the reliable dissemination of data in sensor networks, the authors of *Infuse* [48] proposed a TDMA based data dissemination protocol for sensor networks. Since TDMA ensures a deterministic slot when a sensor node should transmit its packet, it offers a degree of reliability which is used by the data dissemination strategy adopted in Infuse. The authors tackle the problem of random message losses in the presence of channel errors by considering recovery algorithms based on sliding window protocols, modified to use implicit acknowledgments.

In [29], the authors performed an experimental and empirical study of the epidemic style algorithms in large scale multihop wireless networks.

Specifically, dissemination protocols for code update in sensor networks have been given special focus in recent years. All these protocols have concentrated on reprogramming a set of static nodes scattered in a terrain. Trickle [53] is a code maintenace algorithm which works on a polite gossip based periodic advertisement of metadata. Deluge [36] is based on Trickle and is meant for transferring bulk code by dividing it into fixed size pages and pipelining the pages across the network. However, Deluge suffers from the hidden terminal problem in dense networks. MNP [49] improves on Deluge's hidden terminal problem by implementing a sender selection algorithm in a neighborhood by which one single node would transmit data. It transfers the whole image in a phase-by-phase manner across single hops. Although it saves on energy, the code propagation process takes longer. An extension of MNP was proposed in Gappa [77] where parts of a code can be communicated to a subset of sensors on multiple channels.

We have, subsequently in section 2.1, dealt with some of these dissemination protocols for code update in further detail as we have used them in our dissertation for evaluating our framework.

Security of these protocols is an area of major concern and a few other works [27, 24, 51] have been proposed which try to secure the code transfer over these protocols.

These protocols discuss authentication techniques for securing data dissemination in sensor networks. However, they have a few drawbacks when these protocols are used generically for disseminating data from any sensor node acting as a source. We discuss this vulnerability of these protocols in chapter 5. Although most of these protocols provide a reliable transfer of data across the network and can handle occasional node failures, they are unsuitable or inefficient when used in a mobile sensor network scenario where nodes are dynamic all the time. We have not come across any existing work that proposes a protocol for disseminating data efficiently in a mobile sensor network. This is the basis for our work in chapters 6 and 7.

Moreover, the fundamental model on which our framework formulations are based, relies heavily on a branch of mathematics for modeling the spread of an infection in a susceptible population, namely, *Epidemic Theory*. We have used this branch of biologically inspired technique for solving our formulated problems.

Subsequently, we briefly discuss this subject in this chapter in section 2.2 before embarking on using it to address the key problems discussed in this dissertation in subsequent chapters.

In chapter 4, we delve into the threatening aspect of node compromise spreading across a sensor network from a few initially compromised nodes. We have proposed models to analyze and derive critical parameters to aid in the design of more robust and secure sensor networks. Node compromise in sensor networks and the need for their security has received immense attention in the research community[34]. A large portion of current research on security in sensor networks has been focused on protocols and schemes for securing the communication between nodes [54, 28, 55]. In [28], the authors propose a random key distribution scheme for secure communication among sensor nodes. In [54], the authors improve on the work in [28] by taking advantage of node location information to improve key connectivity. In [26], the authors discuss a key management scheme based on node deployment knowledge.

They consider a group based deployment where the resident points of nodes in each group follow a two-dimensional gaussian distribution around the deployment point of the group. In [65], the authors provide critical values of the size of the keyring and the key pool such that the network is not only connected but also resilient against the capture of a fixed fraction of the nodes and their keys. However, we consider a dynamic process whereby the adversary acquires more keys by propagating the node compromise process from a small set of nodes. Revocation of keys of compromised nodes has been studied in [12] where the authors define basic properties that distributed sensor-node revocation protocols must satisfy and present such a protocol that satisfies these properties under general assumptions and a standard attacker model. In [34], the authors demonstrate the ease with which a sensor node can be compromised and all its information extracted. Unfortunately, little work has been done on the defense strategies when the compromise of a single node could be used to compromise other nodes over the air. Our work takes the first step towards modeling this potentially disastrous propagation[18].

2.1 Data Dissemination Protocols for Code Update in Wireless Sensor Networks

Several data dissemination protocols in wireless sensor networks for purposes of reprogramming and maintenance of code have been proposed in the recent literature. We have adopted a few of them to showcase the models and associated analyses. In this chapter, we would review some of the key protocols and briefly discuss them and their mechanism of operation.

2.1.1 Trickle

Trickle [53] is one of the first protocols proposed for propagating and maintaining code updates in wireless sensor networks. The basic algorithm of Trickle is

based on a "*polite gossip*" policy where nodes periodically broadcast code summary or metadata advertisements to local neighbors for maintenance. Trickle assumes that nodes can describe their code with metadata in a concise manner and by comparing two different pieces of metadata, can determine which node needs an update. In Trickle, each node would periodically transmit its code metadata if it has not heard a few of its neighbors transmit the same thing. However, a node would choose to remain silent if it has recently heard a metadata identical to its own. When a node hears old gossip, it triggers a code update, so that the gossiping node can be updated. Trickle also regulates its rate of gossiping based on received gossip information. In other words, a node will gradually reduce its gossip rate if it does not hear new information. However, when it indeed overhears any new gossip, the rate will be increased automatically.

Thus, the local and stateless feature of Trickle's periodic maintenance mechanism allows it to scale to thousands of nodes and different folds of network density, quickly propagate updates, distribute transmission load evenly, be robust to transient disconnections, handle network repopulations, and impose a maintenance overhead on the order of a few packets per hour per node. Thus, the result of a metadata broadcast could be either that every neighbor is up to date or a recipient node detects the need for an upgrade. It does not matter who transmits first and as long as there is some minimal communication in each neighborhood and the network is connected, nodes will stay up to date. Formally speaking, Trickle divides time into a series of rounds. Each node maintains a counter c, a threshold k, and a timer t randomly chosen in the range $[0, \tau]$, τ being a fixed time constant. k is a small fixed integer. When a node hears metadata identical to its own, it increments c. At time t, the node broadcasts its metadata if $c < k$. The random selection of t uniformly distributes the choice of who broadcasts in a given interval. When the interval of size τ completes, c is reset to zero and t is reset to a new random value in the range [0, τ].

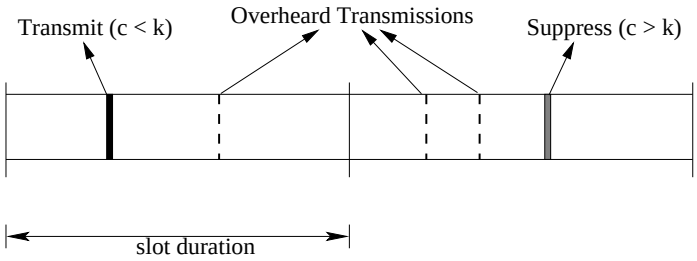


Figure 2.1. Working Principle of Trickle.

If a node with code ϕ_x hears a metadata for ϕ_{x-y}, it broadcasts the code necessary to bring ϕ_{x-y} up to ϕ_x. If it hears a summary for ϕ_{x+y}, it broadcasts its own summary, triggering the node with ϕ_{x+y} to send updates. Fig. 2.1 shows a visual depiction of the working principle of Trickle for two intervals of length τ with $k = 1$.

The random selection of t uniformly distributes the choice of who broadcasts in a given interval. This evenly spreads the transmission energy load across the network. If a node with n neighbors needs an update, the expected latency to discover this from the beginning of the interval is $\frac{\tau}{n+1}$. Detection happens either because the node transmits its metadata, which will cause others to send updates, or because another node transmits a newer metadata. A large τ has a lower energy overhead (in terms of packet send rate), but also has a higher discovery latency. Conversely, a small τ sends more messages but discovers updates more quickly.

For further details, the reader is referred to the original paper [53].

2.1.2 Firecracker

Firecracker [52] is another data dissemination protocol in wireless sensor networks. In Firecracker, the authors have combined routing and local broadcast to rapidly transfer a piece of data to all nodes of the network. Broadcasting is favorable when a piece of data needs to be sent to all the nodes. However, it needs to be done in an energy efficient way. In order to minimize the amount of energy spent in delivering the data, nodes either reserve the channel or rebroadcast carefully such that collisions

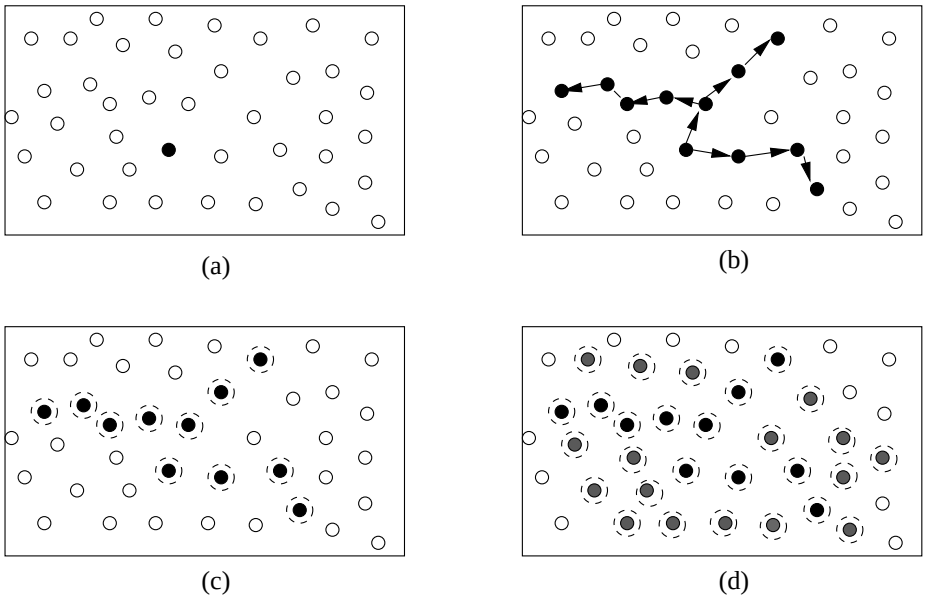


Figure 2.2. Working Principle of Firecracker: (a) Source initiation, (b) Routing to Seeds, (c) Broadcast from Seeds, (d) Further Propagation.

are unlikely. Explicit reservation of channel requires additional energy in the form of control messages, which further introduce latency. The latter technique requires suppression timers, which imposes delay on each hop.

However, routing is favorable when the network needs to send data quickly to a single destination. Nodes can forward data along the single path without having to worry about suppression. Unfortunately, it is difficult to address each and every node in a sensor network. Thus, although this technique is faster than broadcast, routing to every node is less energy efficient. Firecracker combinines these two techniques of routing and broadcast to achieve dissemination rates close to routing while still maintaining the energy efficiency of broadcasting. To start the dissemination process, the data source first routes data to certain distant points termed as *seed points* in the network. Once the data reaches the respective destinations, broadcast-based dissemination begins along the path like a string of firecrackers. In other words, nodes along the route also cache the data they forward and neighboring nodes along the route who overhear can also do the same. The network can then start the broadcast based

dissemination from each of the nodes that received the data. Thus, it is noteworthy that the most efficient end-to-end route is not necessarily the best one. Taking a long circuitous route along the network can possibly propagate data more quickly than a purely broadcast based dissemination.

Thus, the first essential part of Firecracker is the seed selection. The farther the seed points from the origin, the faster data can propagate. Moreover, these seeds should also be distant from one another, or traffic along their routes would be redundant.

The routing protocol must allow nodes to address arbitrary nodes in the network. Since the purpose of the routing phase is to spread the data to distant points in the network, a naming scheme that allows nodes to choose such points is helpful. Random selection of seed points is beneficial provided the random points are far away from the source. The routing mechanism can be any of the standard mechanisms for routing in sensor networks[37], [16], [81], [47]. For instance, in a routing protocol such as GPSR [40], this would involve routing to a geographic location well outside the area of the network.

Fig. 2.2 shows the working principle of firecracker where the source seeks distant seed points from where local broadcast based dissemination starts. Thus, with the correct selection of seeds, Firecracker can speed up the data dissemination process across the network. For further details, the reader is referred to [52].

2.1.3 Deluge

Deluge [36] is a reliable data dissemination protocol for propagating large data objects from one or more source nodes to many other nodes. The density aware epidemic properties makes it reliable against unpredictable wireless environments and robustness against varying densities of nodes. The large data object that needs to be transported is broken down into manageable chunks of fixed size called *pages*.

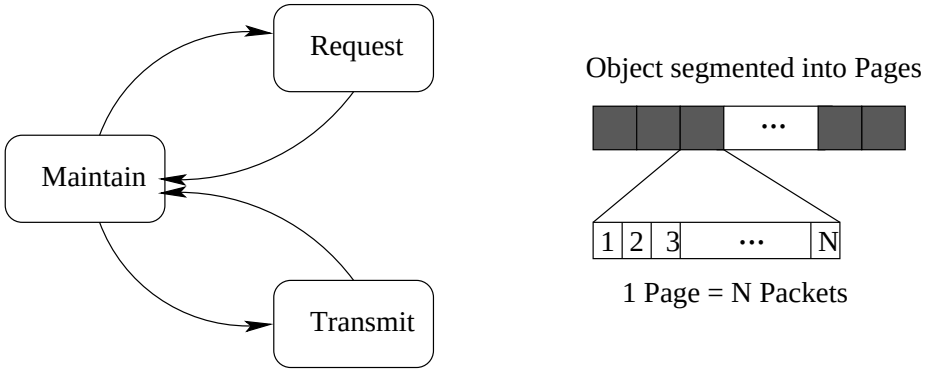


Figure 2.3. Deluge Protocol.

This representation of the data object allows for spatial multiplexing and supports efficient incremental upgrades. It also limits the state that a receiver must maintain while receiving data.

Deluge utilizes Trickle as a basic maintenance mechanism. In Trickle, as explained earlier, nodes stay up-to-date by periodically broadcasting a code metadata to their neighbors. A node suppresses its own broadcast if it recently overhears a similar code metadata. When the nodes are not up-to-date, the broadcast rate is reduced, but is otherwise increased upto a specified limit.

Deluge is an epidemic protocol operating as a state machine where each node follows a set of strictly local rules to achieve the global goal of reliably disseminating large data objects. A node operates in one of three states at any given time: MAINTAIN, RX or TX. The set of local rules an individual node follows is a function of its current state and specify what actions and state transitions to take in response to events. A node in the MAINTAIN state is responsible for ensuring that all nodes within communication range have (i) the newest version of the object profile and (ii) all available data for the newest version. To maintain this property, each node uses Trickle to periodically advertise a summary describing the current version of its object profile and the set of pages from the object which are available for transmission.

Trickle's suppression mechanism helps to minimize the set of senders and simplifies the decision making process of nodes at any given time. The only trigger that causes a node R to request data from another node S is the receipt of an advertisement stating the availability of a needed page. A significant contribution of Deluge is its emphasis on spatial multiplexing. Deluge advertises the availability of received pages even before all pages in the object are complete, allowing faster propagation of the object. Consequently, overall throughput is increased by pipelining the transfer of pages across the network. In order to realize the full benefit of spatial multiplexing, Deluge takes special care to ensure that transfers of different pages do not interfere with each other. First, Deluge constrains nodes by requesting pages in sequential order, that is, a request for page i cannot be made unless data for all pages in the range $[0, i)$ are also up-to-date.

A node in the RX state is responsible for actively requesting the remaining packets required to complete a page. Each request operates as a selective negative acknowledgment (SNACK), where a bit-vector specifies which data packets in the page are needed.

A node in TX is responsible for broadcasting all requested packets for a given page (continuing to service any subsequent requests for data from the same page) until all requested packets have been broadcast and then transition back to MAINTAIN. Deluge services requests by taking the union of any new requests with previous requests not yet serviced.

Deluge thus provides a reliable and scalable mechanism of propagating a set of pages constituting an entire data object in a spatially multiplexed manner across the whole network.

2.1.4 MNP

MNP [49] is another broadcast-based multi-hop reprogramming protocol for sensor networks. In particular, it addresses the issue of message collisions and hidden terminal problems in previous reprogramming protocols like Deluge. It is similar to Deluge in that it uses periodic advertisement of metadata for maintaining code synchronization with neighbors. However, it also implements a *sender selection algorithm* which attempts to guarantee that, in a neighborhood, there is at most one source transmitting the program at a time. Each source node competes with one another based on the number of distinct requests they have received. Contrary to Deluge, where the image is broken down into pages and pipelined across the network, in MNP, the whole image is sent in a phase-by-phase approach across each hop.

Recognizing the importance of preserving energy in sensor networks, MNP incorporates some power saving techniques to do its reprogramming. It reduces radio transmission by preventing too many source nodes from transmitting concurrently, as such concurrent transmissions create excessive contention. MNP achieves this by using a sender selection and suppression scheme.

The working principle of MNP is divided into 4 distinct phases, namely *Advertisement/Request*, *Forward/Download*, *Query/Update* and *Reboot*. In the *Advertisement/Request* phase, sources advertise the new version of code they have, and all interested nodes request the code. Depending on the requests it receives and advertisement messages it overhears from other sources, a node decides whether it should start forwarding code or go to "sleep". In this stage, nodes apply the sender selection algorithm. Based on a counter of the number of requests that a source node receives, the sender is elected based on its potential to serve the highest number of requesters. after the election is done, the nodes that yielded to the chosen sender go to sleep to save energy.

Once a source has been selected, it broadcasts a "StartDownload" message to inform all the receivers in its neighborhood to get prepared for the arrival of new code. The program image is divided into segments, each of which fits into a single packet. The sender sends the program code packet by packet to the receivers. In the *Query/Update* phase, once the new program code has been transmitted, the sender broadcasts a "query" message to its receivers, which respond by requesting the packets they are missing. The missing packets are retransmitted by the source node using broadcast. In the *Reboot* phase, the node transfers downloaded code into program memory and reboots with the new code. Thus, MNP succeeds in saving energy by having only one sender per neighborhood transmitting its code. However, this selection of a sender increases the overall latency of code transfer across the network.

2.2 Epidemic Theory and its Applications

In order to appreciate the epidemiological models applied in wireless sensor networks, we need to first understand the concept of epidemic theory. In this section we provide a terse description of the theory and its applications. Epidemic Theory is the study of the dynamics of how contagious diseases spread in a population, resulting in an epidemic and it is well documented [3, 56, 35, 4]. Primarily, the theory mathematically models the propagation process of an infection and measures its outcome in relation to a *population at risk*. The population at risk basically comprises of the set of people who possess a susceptibility factor with respect to the infection. This factor is dependent on several parameters like exposure, spreading rate, previous frequency of occurrence, etc., which define the potential of the disease causing the infection. Among the different models characterizing the infection spread, two are quite popular. They are the *Susceptible Infected Susceptible* (S-I-S) Model and the *Susceptible Infected Recovered* (S-I-R) Model. In the former, a susceptible

individual acquires the infection and then, after an infectious period (i.e., the time the infection persists), the individual becomes susceptible again. In the latter, the individual recovers and becomes immune to further infections.

An approach to modeling the propagation of an infection is to assume that the probability (per unit time) for a susceptible individual to acquire infection is equal to the average rate at which new infective partners are acquired multiplied by the probability of being infected by any one such partner. In the general deterministic S-I-R model, if *N(t)*, *S(t)*, *I(t)* and *R(t)* denote the total population, the susceptible, the infected and the recovered or immune individuals, respectively at time t, we can say

$$N(t) = S(t) + I(t) + R(t) \tag{2.1}$$

Let us assume that β denotes the infection rate and γ denotes the removal rate of infected individuals. Assuming a homogeneous mixing model i.e., each of the susceptibles can get in contact with any of the infectives, we can see that in time Δt, there are $\beta SI\Delta t$ new infections and $\gamma I\Delta t$ removals. Therefore, the basic differential equations that describe the rate of change of susceptible, infective and recovered individuals are given by:

$$\begin{aligned} \frac{dS(t)}{dt} &= -\beta S(t)I(t) \\ \frac{dI(t)}{dt} &= \beta S(t)I(t) - \gamma I(t) \\ \frac{dR(t)}{dt} &= \gamma I(t) \end{aligned} \quad .$$

The above equations can be solved either approximately or precisely based on some boundary conditions. For example, at the start of the epidemic, when $t = 0$, (S, I, R) can take the values $(s_0, i_0, 0)$. Note that, in particular if i_0 is very small, s_0 is approx-

imately equal to N. It also follows that only if the *relative removal rate*, $\mu = \gamma/\beta$, is greater than s_0 can an epidemic start to build up as this condition will result in $[dI(t)/dt]_{t=0} > 0$, i.e. $I(t)$ will have a positive slope. Therefore, the relative removal rate $\mu = s_0$ gives a threshold density of susceptible nodes.

On the other hand, the S-I-S model does not have the recovered subset $Z(t)$ and those who are infected fall back into the susceptible subset $S(t)$ after their infectivity duration.

Of particular interest in epidemiological studies is the phenomenon of phase transition of the spreading process that is dependent on a threshold value of the epidemic parameter, i.e., if the epidemic parameter is above the threshold, the infection will spread out and become persistent; on the contrary, if the parameter is below the threshold, the infection will die out. Identification of this threshold value is critical in the study of how an epidemic spreads and how it can be controlled.

Apart from the continuous differential rate equation based modeling technique, the study of epidemics has often been performed by treating the population as a network graph, with the nodes representing each individual and the edges their interaction. This form of analysis [60] has mainly been used in scenarios where the end result of the epidemic spread is more important than the temporal dynamics of the propagation.

In our work in chapter 4, we adopted some of the directions presented in [60] where the author proposes a percolation theory based evaluation of the spread of an epidemic on graphs with given degree distributions. However, their work is a generic analysis of epidemics in random graphs. In our work, we have considered the specific characteristics of sensor networks including distance, deployment and key constrained communication patterns.

Connectivity issues in random ad hoc networks are extremely important as a pre-requisite before any epidemic-like propagation process is analyzed. In [6, 63], the authors derive threshold values of the transmission range of the nodes that ultimately make the network k-connected with a given probability. The thresholds for the monotone properties of random geometric graphs have also been dealt with in [30].

In fact, visualizing the population as a complex network of interacting individuals has resulted in the analysis of epidemics from a network or graph theoretic point of view[57, 61, 62]. Specifically, the scale free topology has been of keen interest [61, 80, 5, 60] and this model has been the basis for the analysis and extensive study of virus and worm spreading in the Internet [74, 44, 42, 43, 70, 32].

Several works have spawned [61], [62], [56], [23], [57], [31] where the spread of infectious diseases in a human population have been studied by modeling the social network of humans as a scale free topology.

Epidemic Theory has found special attention in the design and modeling of several phenomena and protocols in sensor networks wherever there is a scope of information distribution in a large scale preferably from a small number of sources to a large number of recipients. Among the popular phenomena in sensor and ad-hoc networks where this theory has been adopted are data dissemination, broadcast protocols and routing.

Since data dissemination primarily deals with the transfer of messages from one node to all nodes of a network, algorithms based on epidemiological formulations are a perfect fit. Accordingly, these algorithms have been successfully used in disseminating information in sensor networks and depending on the application, the dissemination can start at a single node, such as a base station, or at multiple sensor nodes. The decentralized and distributed nature of wireless sensor networks fits the context of epidemic algorithms aptly.

In general, epidemic algorithms for data dissemination follow the model of nature to spread information and define simple rules for information to flow between nodes of a network. The authors in [1] have done a comparative study of epidemic algorithms for data dissemination based on the style of communication between neighboring nodes, i.e., pull based, push based or both.

Although flooding has been used with some optimization to route packets in an ad hoc network, many routing messages are propagated unnecessarily. The authors in [33] have proposed a *gossip*-based approach where each node decides to forward a message to another node based on some probability. The authors in *Geographic Gossip* [25] propose an alternative gossiping scheme, thats exploits geographic information and build a completely randomized and distributed algorithm that requires substantially less communication. The idea is to include geographic routing to gossip with random nodes far away in the network.

In a network of n sensors, a basic solution to the *averaging problem*, i.e., to compute the average of all n sensor measurements, is based on the *Gossip* algorithms where each node randomly picks a one-hop neighbor and exchange their current values. This is performed in an iterative fashion and ultimately all nodes converge to the global average in a distributed manner. The key issue here is the number of iterations it takes for such a gossip algorithm to converge to a sufficiently accurate estimate. Recent works [7], [8], [14], [39], [41] have dealt with variants of this problem. The convergence time of this algorithm is closely linked with the mixing time of the Markov Chain defined by a weighted random graph on the network. In [8], the authors showed how to optimize the neighbor selection probabilities for each node in order to find the fastest mixing Markov chain. However, for sensor network graphs, even an optimized gossip algorithm can result in excess energy consumption.

The authors of *Smart Gossip* [50] propose an adaptive form of gossiping in sensor networks. They propose techniques by which a gossip based protocol can au-

tomatically and dynamically adapt to the network topology. Smart Gossip copes well with wireless losses and unpredictable node failures that affect network connectivity.

In [45], the authors develop a *topologically-aware worm propagation model*(TWPM) in wireless sensor networks where worms spread by localized scanning to select random targets.

Probabilistic broadcasting in a mobile environment have been dealt with in [76]. In [84], the authors proposed a dynamic scheme to change the rebroadcasting probability based on node distribution and movement.

2.3 Summary

In this chapter, we have reviewed some of the associated topics which are pertinent to the research performed in this dissertation. In particular, we have delved into various epidemiological models and protocols employed in wireless ad hoc and sensor networks. Starting from data dissemination and gossip protocols to security issues in sensor networks such as propagation of compromise of sensor nodes, we observe that there have been several works inspired by this powerful concept of epidemic theory. The density and scale of a sensor network coupled with the objective of a one-to-many data transfer from a few nodes to the rest of the network, unleash the efficiency with which this theory can effectively model and provide solutions to several problems in ad hoc and sensor networks.

We have, specifically, provided an overview of each of the existing data dissemination protocols that we have adopted for showcasing the working of our model. We have also briefly explained Epidemic theory, which is a method for the mathematical modeling of infectious diseases on which our framework is based.

In subsequent chapters, we discuss the usage of epidemic models to formulate specific problems related to data dissemination and their solutions.

CHAPTER 3

PERFORMANCE ANALYSIS OF DATA DISSEMINATION PROTOCOLS

Multihop broadcast based data dissemination protocols such as Trickle [53], Firecracker [52], Deluge [36] and MNP [49] have emerged as convenient means for distributing data and code to the entire sensor network. These protocols are essential for remotely retasking or reprogramming large numbers of sensors, particularly when deployed in inaccessible terrain.

In this chapter, we conduct a comparative analysis of the performance of data transfer by these different broadcast-based dissemination protocols. We construct a mathematical framework of these protocols wherein we focus on the speed of transfer of information and coverage across the network. We are able to map different protocols onto this framework so as to study their propagation behavior on a common platform. Our analyses provide useful insights into the rate of propagation of data over these protocols under given conditions of network connectivity.

This chapter is organized as follows : In section 3.1, we present the sensor network topology model. In section 3.2, we present the epidemic theoretic analytical framework. In section 3.3, we fit in each broadcast protocol into the framework. We perform our simulation study in section 3.4 and summarize the chapter in section 3.5.

3.1 Sensor Network Model

In this section, we present our model of the physical topology of the sensor network used for our analysis. We model a wireless sensor network as an undirected geometric random graph $G_{p(d_{uv})}(N)$ [64] of N nodes, based on the unit disk model [17]

where $p(d_{uv})$ is the probability of having a link between nodes u and v at a distance d_{uv} from each other. The expected number of links in the network is then given by

$$E_d = \sum_{u=1}^{N} \sum_{v=u+1}^{N} p(d_{uv}) \tag{3.1}$$

Correspondingly, the mean degree, η, of a node is given as:

$$\eta = \frac{2E_d}{N} \tag{3.2}$$

The link existence probability $p(d_{uv})$ is based on the transmission radius R_c of each node, which is computed using suitable radio propagation models for wireless sensor networks. For example, if the received power at distance r from the transmitter is denoted by $\mathbf{P}(r)$, then in the log-normal shadowing model [68], the assumption is that the logarithm of $\mathbf{P}(r)$ is normally distributed.

3.2 An Analytical Framework based on Epidemic Theory

In this section, we propose a novel framework based on epidemic theory for analyzing the propagation of information over a sensor network broadcast protocol. The framework captures both the local spatial interaction in a static network scenario and the temporal dynamics of the propagation process. Our idea is to model the communication between a node having the information and the one requiring it, as a contact between an infected individual and a susceptible one. Just as a susceptible individual might get infected with a certain probability once it is in contact with an infective one, the node requiring the information would get infected if it receives the propagating piece of data through communication with an already infected node. Incidentally, we would denote a node that already contains the propagating information

as an infected node and the node requiring the propagating piece of information as a susceptible node.

Although it is a data spread across the whole network that we are focusing on, for the sake of consistency, we have adhered to the same terminology as used in epidemic theory, i.e., assigning the term *infection* to a data update.

3.2.1 System Model

The population in our model is the total number of nodes, N, in the sensor network which are assumed to be stationary and uniformly randomly distributed with the node density denoted by σ. The number of infected nodes $I(t)$ at time t are those that have been infected by the information spreading over the broadcast protocol. Likewise, $S(t)$, denotes the set of susceptible nodes at time t.

The rate of infection, β, represents the probabilistic rate at which an infective node communicates with a susceptible one through a broadcast protocol, thus infecting the latter. Here β depends on two factors: (i) probability ρ representing the infectivity of the information which is a measure of how essential or critical it is, and (ii) the rate of communication of the protocol. The degree of susceptibility of a node depends on its average degree η, the rate of communication between nodes, and the probability ρ.

Since the sensor nodes are assumed to be stationary, they cannot *homogeneously mix* with any other node in the network. This implies that when all the neighboring susceptible nodes around an infective node i acquire the infection, then i is rendered inoperative and does not contribute further to the infection spread. Moreover, we assume that an infected sensor node uses the normal operation of a broadcast protocol to spread the information to its neighbors. Thus, the infection rate is dependent on the communication rate of the broadcast protocol.

Table 3.1. Performance Model Parameters

Model Parameter	Description
N	Total number of nodes
η	Average node degree
σ	Node density
R_c	Communication radius of node
S(t)	Susceptible nodes at time t
I(t)	Infective nodes at time t
β	Data infection rate
ρ	Data infectivity potential

The working principle of a broadcast protocol states that once a node has new data, it updates its surrounding neighbors by first sending an advertisement. This implies that there is a circular region of infected nodes centered at the source node which grows with time as the infection spreads outwards riding on top of the broadcast protocol. We approximate this observation into our model by having nodes on the periphery or wavefront of the infected circular region trying to infect their susceptible neighboring nodes lying outside this circle. These susceptible neighbors reside in a circular strip of width equivalent to a node's communication radius R_c, outside the infected circle as illustrated in Fig. 3.1.

We derive analytical expressions for each sub-population function $S(t)$ and $I(t)$ for the information spread.

In our analytical study, we have made certain assumptions that we highlight in this section. They are as follows :

- The model does not assume channel contention delay when an infective node is communicating with a susceptible one.

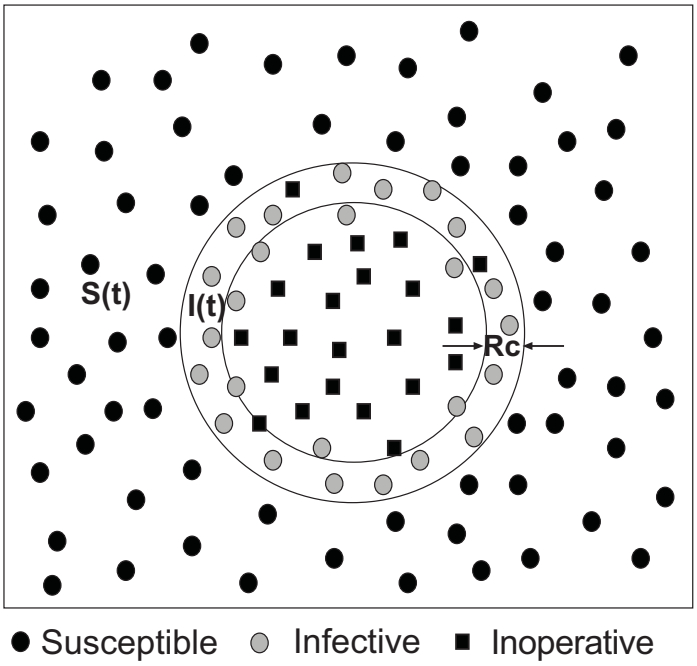


Figure 3.1. The Spreading Phenomenon in a Sensor Field.

- Packet loss is predominantly assumed to be caused by packet collisions. Existing links between two nodes is assumed to be of fairly good quality so that packets are negligibly lost due to a failing link.

3.2.2 Model Analysis

In this section, we present a detailed analysis of the propagation mechanism by deriving the functions describing the dynamics of each sub-population with time. The following lemma is used to calculate the number of nodes in a circular strip of radius h hops where one hop length is dependent on the density of nodes.

Lemma 3.1. *Given that sensor nodes are uniformly randomly distributed in a field, the number of nodes which are h hops away from a source node is $O(h)$.*

Proof. As we assume N nodes to be uniformly randomly distributed in a square of unit area, the number of nodes along each side of the unit square is $O(\sqrt{N})$ with an average hop length of $O(1)$. The average distance between a source and destination

node in this square is $O(\sqrt{N})$. Thus, if the average distance between any two nodes is $O(N)$, then there are $O(N^2)$ nodes present in the unit square. With uniform random deployment of nodes in a circle having a radius of h hops, the total number of nodes present is $O(h^2)$. Thus, the number of nodes which are h hops away from a source situated at the center, along the circumference, is $\psi\pi(h^2 - (h-1)^2) = O(h)$ where ψ is the node density. □

We now present our model for the information propagation in the network.

For modeling information propagation using epidemic principles, we adopt the formulation of infection spread where there is no infection recovery. Thus, $R(t) = 0$ and $\gamma = 0$ in Eqs. (4) and (5). Consequently, with time there is a gradual increase in the number of infected nodes, ultimately reaching the whole network. In this case, we have non-homogeneous mixing because only the infected nodes that lie within distance R_c from the periphery of the circle of infected nodes can communicate with the susceptible nodes, and thus have the potential to infect them. For instance, in Fig. 3.2 the infected node k cannot infect a susceptible node because all the susceptible nodes fall outside its communication range. Thus, all the nodes, such as k, that lie in the interior of the infected circle are essentially inoperative and do not spread further infections. The number of infected nodes $I'(t)$ that lie in the circular strip of thickness R_c from the circumference is given by:

$$I'(t) \cong I(t) - \sigma\pi(r(t) - R_c)^2,$$

where σ is the uniform density of nodes and $r(t)$ is the radius of the circle that contains the infected nodes. Note that, $\sigma\pi r(t)^2 \cong I(t)$. After simplification we obtain:

$$I'(t) \cong (2\sqrt{\sigma\pi}R_c)\sqrt{I(t)} - \sigma\pi R_c^2. \tag{3.3}$$

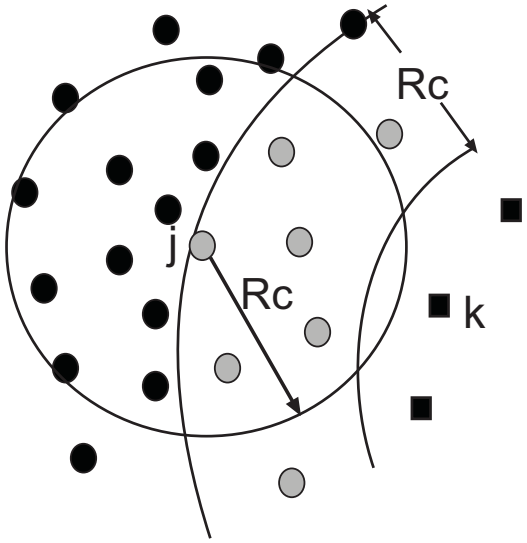


Figure 3.2. Circular strip of thickness R_c. Only a fraction of neighboring nodes of an infected node are the potentially susceptible ones. .

We observe that I' is of the order of $O(\sqrt{I})$ as per Lemma 1 and hence can be approximated as $I' = c\sqrt{I}$, where $c = 2\sqrt{\sigma\pi}R_c$ is the proportionality constant. Now, the set of susceptible nodes that are able to communicate with I' is a small fraction of $S(t)$. In particular, if η is the average degree of a node, then each node in $I'(t)$ is able to communicate with only η neighbors on the average. The radio transmission range, R_c, defines a node's neighborhood and its degree η. However, not all of the η neighbors of an infected node are susceptible. As illustrated in Fig. 3.2, for example, only the susceptible nodes that lie within the circle of radius R_c can potentially be infected by node j. As the data update propagates, we observe that for each infected node j in the peripheral circular strip, it tries to infect the susceptible fraction of its η neighbors. Thus, we can write the mass balance equation as:

$$N(t) = S(t) + I(t) \tag{3.4}$$

and the differential equations as:

$$\frac{dI}{dt} = \beta c\sqrt{I}\frac{(N-I)}{N}\eta \tag{3.5}$$

$$\frac{dS}{dt} = -\beta c\sqrt{I}\frac{(N-I)}{N}\eta \tag{3.6}$$

Substituting $U = 1/\sqrt{I}$, the first equation can be simplified into the following:

$$\frac{dU}{U^2 - 1/N} = -\frac{\beta c\eta}{2}dt \tag{3.7}$$

which after integration on both sides and applying the boundary condition $I(0) = 1$, i.e., initially only one node was compromised, leads to the following:

$$I(t) = N\left(\frac{2}{1 + (\frac{\sqrt{N}-1}{\sqrt{N}+1})\, e^{-\frac{\beta c\eta}{\sqrt{N}}t}} - 1\right)^2 \tag{3.8}$$

Note that in the above equation when $t = \infty$, $I(t) = N$, i.e., asymptotically all the nodes will be compromised. Eq. (3.8) gives the rate at which the infection of compromised nodes spreads across the network. For mapping a broadcast protocol onto this model, we would derive β in terms of the communication rate of the protocol.

3.3 Analysis of Individual Broadcast Protocols

Given the above framework, in this section, we address each of the broadcast protocols, Trickle, Firecracker, Deluge, and MNP. For details about the working mechanism of these protocols, the readers are referred to chapter 2. Our methodology is to apply the derived framework by investigating the key parameters specific to each of the protocols. Our goal is to derive the infection rate, β, for each of them.

Although there are possible similarities in the functional operation of these protocols, they are fundamentally different in their design and target applications.

While Trickle is mainly concerned with maintaining code and propagating small data updates around the network with minimum overhead, Deluge focuses on propagating large chunks of data in a pipelined fashion across the network. Firecracker, on the other hand, combines routing and local broadcast for quickly transferring information to different parts of the network. MNP propagates large data chunks like Deluge, as well as employs a sender selection mechanism to enforce a single data source in a neighborhood.

3.3.1 Trickle Protocol

In Trickle, we derive the probability of the periodic metadata broadcast and then use it to compute the average rate of updating a neighbor with data.

Lemma 3.2. *In the Trickle protocol, if the expected number of communication neighbors of a node i is denoted by η, then the probability p_k of i broadcasting metadata in each time interval is given by $p_k = \frac{k}{\eta+1}$ where k denotes the advertisement threshold.*

Proof. A node broadcasts advertisements at most once per period T_p at a random time $t \in [0, T_p]$. However, if the number of received advertisements is less than the threshold k, it will choose to transmit its own advertisement or suppress it. Assuming that t is uniformly randomly distributed in the interval $[0, T_p]$, the expected time between successive advertisements is $\frac{T_p}{\eta+1}$. Thus, the expected time E_k, for k advertisement transmissions, is $\frac{k \cdot T_p}{\eta+1}$. The probability p_k for a node to transmit its metadata is, therefore, directly proportional to E_k. Normalizing p_k by dividing with the period duration T_p, we have $p_k = \frac{k}{\eta+1}$ □

Theorem 3.1. *In Trickle, the expected time for a node to receive metadata is given by $E[T_{adv}] = \frac{T_p}{2} \cdot \sum_{i=1}^{k} \binom{\eta+1}{i} {p_k}^i (1-p_k)^{\eta+1-i} \cdot \frac{1}{1-l_r}$ where l_r is the packet loss rate.*

Proof. If the packet loss rate is denoted by l_r then the expected number of transmissions for a given packet is $\frac{1}{1-l_r}$. Given that the expected number of neighbors of a

node is η and the probability of a node in a neighborhood to transmit metadata is p_k, the probability that at least one node transmits metadata is given by the binomial expression $\sum_{i=1}^{k} \binom{\eta+1}{i} {p_k}^i (1-p_k)^{\eta+1-i}$. Furthermore, since a node selects a random time in the interval $[0, T_p]$ to transmit metadata, the expected delay before transmitting a metadata is $\frac{T_p}{2}$. Thus the net expected delay to successfully transmit metadata in a single hop neighborhood is given by $E[T_{adv}] = \frac{T_p}{2} \cdot \sum_{i=1}^{k} \binom{\eta+1}{i} {p_k}^i (1-p_k)^{\eta+1-i} \cdot \frac{1}{1-l_r}$

□

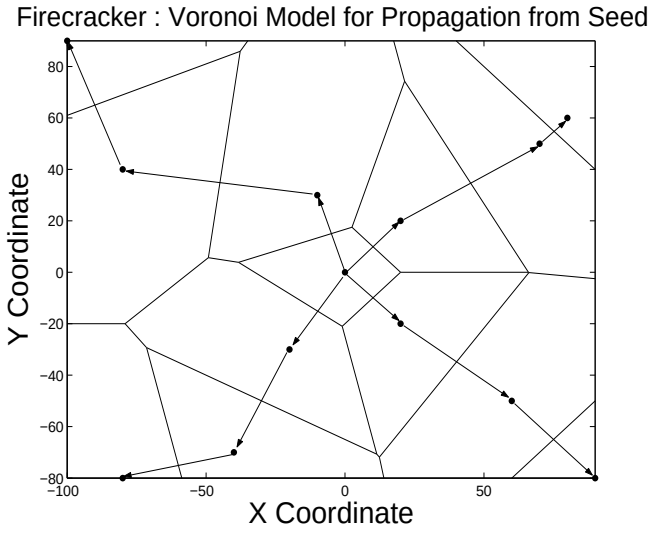


Figure 3.3. Firecracker Protocol Model : Voronoi Partitioning.

In Trickle, the moment that metadata has been transmitted in a neighborhood, the nodes immediately update themselves by broadcasting the new code update packet. If we denote T_{pkt} as the transmission time of the code update packet, then the expected successful transmission time $E[T_{tx}]$ is given by

$$E[T_{tx}] = T_{pkt} \cdot \frac{1}{1-l_r} \tag{3.9}$$

Thus the total expected delay for a code update is given by

$$E[T_{CU}] = E[T_{adv}] + E[T_{tx}] \tag{3.10}$$

Apart from the rate of transfer established by the physical characteristics of the network and the working principle of the broadcast protocol, there is another important factor affecting the transfer rate of data. This is the inherent characteristic of the piece of data propagating, its type and properties, i.e., how critical and necessary it is for the other nodes to acquire it. We capture this property of the data parametrically by a factor ρ, the infectivity of the data. This parameter ρ differentiates one kind of data from another. Incorporating this factor into our model, the infection spread rate over Trickle protocol is given by

$$\beta = \frac{\rho}{E[T_{CU}]} \tag{3.11}$$

3.3.2 Firecracker Protocol

We recall that Firecracker first routes the data to distant points in the network before starting the local broadcast based dissemination. In order to model the protocol from an epidemiological standpoint, we need to derive the spreading rate β of the protocol. For this, we visualize the end of the routing phase as the beginning of the epidemic process making each of the end nodes of the routing phase (seeds) to be a source of a sub-infection process.

Thus, instead of a single node being the source of the infective spread, we have a set of nodes initiating the process. Moreover, since the nodes are uniformly randomly deployed, the spreading rate from each of the routing nodes is the same. These sub-processes work in the same manner as the broadcast mechanism in Trickle. The population of nodes for each of these sub-processes would be the nodes that are

closer to each seed than to any other seed. Therefore, in our formulation, we divide up the whole network into voronoi partitions where each voronoi cell corresponds to each seed node, as shown in Fig. 3.3. At any given time, the total number of nodes that have been infected is the sum of the number of nodes infected in each voronoi partition.

If there are V seed points in the network which are the destination points of the routing phase, then we have V corresponding voronoi regions with the voronoi points at each seed point. Let n_i denote the number of nodes in the i^{th} region, where $i = 1, 2, \ldots, S$. The spreading in each of these regions is based on a local broadcast method similar to Trickle. Therefore, if $v_i(t)$ denotes the set of infected nodes in region i at time t, then the fraction $f(t)$ of infected nodes at time t is given by

$$f(t) = \frac{|\bigcup_i^S v_i(t)|}{N} \tag{3.12}$$

where $N = \sum_i^S n_i$.

3.3.3 Deluge Protocol

Deluge builds off Trickle, using suppression and dynamic adjustment of the broadcast rate to limit transmissions among neighboring nodes. Similar to the previous two protocols, we identify the parameter in Deluge that allow it to be mapped onto the proposed epidemic model framework. Since the basic unit of transfer in Deluge is a page, we approximate this page transfer rate as the rate at which infection could spread over Deluge.

Given a lossy wireless environment with packet loss rate l_r, the expected number of transmissions for a packet is given by $E[N_{pkt}] = \frac{1}{1-l_r}$. We recall that each page is composed of a constant number of P packets. Since Deluge uses the same maintenance mechanism as Trickle for advertising pages, the expected backoff time is also the

same and equal to $\frac{T_p}{2}$. Similar to Trickle, a node in the MAINTAIN state suppresses advertisements if it has already heard advertisements in its neighborhood for a number of times larger than some constant k. As derived earlier, p_k is the probability that a node transmits advertisements in one time period T_p. Similar to the analysis done for Trickle, the expected time that a node waits for an advertisement to be transmitted in its neighborhood is given by

$$E[T_{adv}] = \frac{T_p}{2} \cdot \sum_{i=1}^{k} \binom{\eta+1}{i} {p_k}^i (1-p_k)^{\eta+1-i} \cdot \frac{1}{1-l_r} \tag{3.13}$$

Moreover, a node also does a random backoff in the RX state before sending a request packet. If $E[N_{req}]$ is the expected number of requests made by a node for acquiring a page, then the time spent for making the requests is given by

$$E[T_{req}] = \frac{T_p}{2} \cdot E[N_{pkt}] \cdot E[N_{req}] \tag{3.14}$$

The transmission time of P packets of a page is given by

$$E[T_{tx}] = P \cdot E[N_{pkt}] \cdot T_{pkt} \tag{3.15}$$

where T_{pkt} is the transmission time for a single packet.

At the same time, when a node in the RX state exceeds its limit by λ requests, it transits to the MAINTAIN state and thus has to wait for advertisements. This time is denoted by $E[T_{fallback}]$ and given by

$$E[T_{fallback}] = \lfloor \frac{E[N_{req}]}{\lambda} \rfloor \cdot E[T_{adv}] \tag{3.16}$$

Thus, the expected time to transmit a page in a neighborhood is given by

$$E[T_{page}] = E[T_{adv}] + E[T_{req}] + E[T_{tx}] + E[T_{fallback}] \quad (3.17)$$

The average rate of page transfer is thus $\frac{1}{E[T_{page}]}$. We assume a page transfer is enough for a malware to compromise a node. Thus, if the infectivity of the malware is ρ, then the average infection rate over the Deluge protocol is denoted by $\beta_D = \frac{\rho}{E[T_{page}]}$.

3.3.4 MNP Protocol

The previous three protocols had similarities between them in parts of their operational methodologies. The reason to choose such similar protocols was to see how the model could capture the subtle differences between them. At the same time, we are also interested in looking at a broadcast protocol (viz. MNP) which is designed differently. This protocol also works to propagate code across the network in a pipelined manner. Alongwith that, it employs a sender selection algorithm to circumvent the hidden terminal problem faced by protocols like Deluge when the network density increases.

Similar to previous analyses, we formulate the expression of the average page transfer rate from a source node to recipients in a neighborhood. We argue that the propagating piece of data uses the data transfer rate of the protocol to spread itself. We assume the same lossy environment as in Deluge and assume that there is a constant number P packets in a page. We simplify our analysis by considering only the basic functioning of MNP without the *query/update* phase. This phase of MNP generally accounts for lost packets. By already taking the lossy wireless characteristics into consideration we can safely neglect this protocol feature of MNP. However, contrary to Deluge, MNP does not use similar maintenance mechanisms as Trickle. A node in the *Advertise* state broadcasts an advertisement message every random

interval. It has a threshold value κ for the maximum number of advertisement messages before servicing requests made by neighbors. This duration of κ advertisement messages is used by nodes in a neighborhood to select an appropriate sending source and allow nodes not interested in the transmission to go to sleep.

We assume that the inter-arrival time of the advertisement messages is negative exponentially distributed with average arrival rate δ. As defined in Deluge, if $E[N_{pkt}]$ denotes the expected number of packets transmitted based on the error rate l_r and T_{pkt} denotes the transmission rate, then the time for a successful packet transmission is $E[N_{pkt}] \cdot T_{pkt}$. Thus, the effective expected inter-arrival time of successfully transmitted advertisement messages is given by

$$T^i{}_{adv} = \frac{1}{\delta} + E[N_{pkt}] \cdot T_{pkt} \tag{3.18}$$

The effective advertisement arrival rate is then given by $\delta_{eff} = \frac{1}{T^i{}_{adv}}$.

Accordingly, the average duration for κ advertisement messages is given by

$$E_\kappa[T_{adv}] = \frac{\kappa}{\delta_{eff}} \tag{3.19}$$

We are analyzing the propagation rate that MNP offers to the data riding on it when the protocol is disseminating new code. Therefore, similar to Deluge, we focus on a situation when there is a new version of code propagating in the network. Subsequently, during this interval $E_\kappa[T_{adv}]$, the advertising node would have received at least one request from a neighbor. Moreover, like Deluge, a requesting node might have to make $E[N_{req}]$ requests to acquire a page. We make a simplifying assumption that κ is chosen in a way such that during the interval $E_\kappa[T_{adv}]$, the advertising node has received at least $E[N_{req}]$ requests. This means that at the end of $E_\kappa[T_{adv}]$, it is ready to service a request.

After $E_\kappa[T_{adv}]$, the source sends the P packets of a page. It also sends a *Start Download* and an *End Download* message signifying the start and end of each page. Thus, the total expected time for a page transfer is given by

$$E[T_{pg}] = E_\kappa[T_{adv}] + E[N_{pkt}] \cdot T_{pkt}(P + 2) \tag{3.20}$$

With ρ denoting the infectivity of the data, the infection rate over MNP is then given by $\beta_M = \frac{\rho}{E[T_{pg}]}$.

3.3.5 Analysis Discussion

There are several important parameters in our model for the derivation of the infection rate β that require careful evaluation for the propagation model to achieve the desired accuracy. For MICA2 motes, the maximum packet transmission rate is around 36 packets/sec with a packet size of 32 bytes. This results in a packet transmission time of 0.027 sec. The average packet loss rate due to effects such as packet collisions, etc., is assumed to lie between 0.1 and 0.2 which is the average value derived from simulation data. Thus, in our formulation, $l_r = 0.1$ and $T_{pkt} = 0.027$ sec. Simulation results of Deluge [36] have shown that the average number of requests for acquiring a page $E[N_{req}]$ is approximately equal to 5.4.

Figures 3.4 illustrates the analytical plots depicting the propagation dynamics for each protocol in a network of 1000 nodes. In Fig. 3.4, sub-figures (a) and (b) show the dynamics of spread over Trickle, with varying infectivity, for average degrees 5 and 8 respectively. The value of the metadata advertisement bound k is equal to 2. We observe that the change in degree from 5 to 8, even for the least infective data, increases the speed of infection by more than 20%. The next two sub-figures depict Firecracker's performance. The source node is situated at the center and the routing destinations are points situated close to the other corners of the field. The effect of

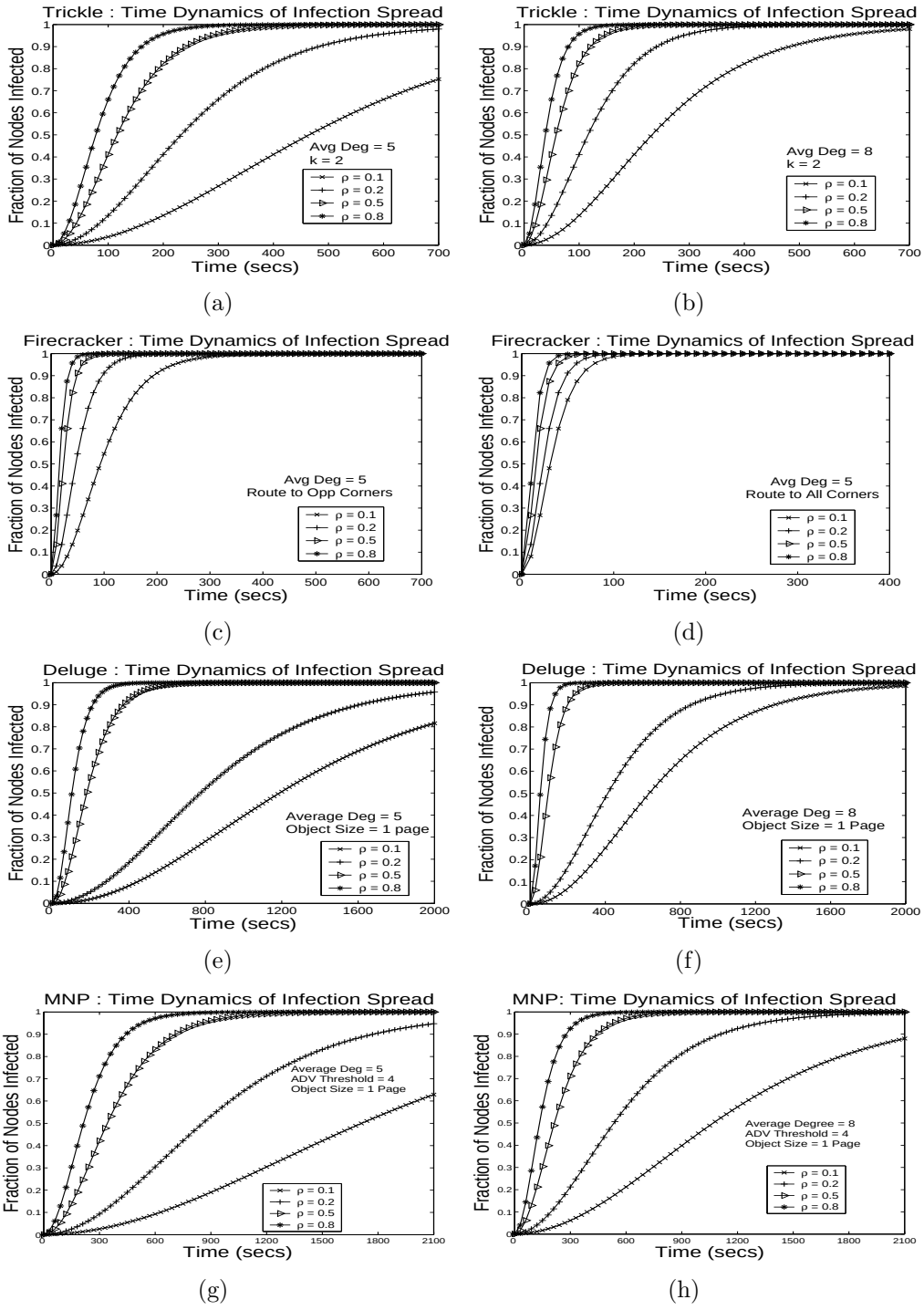


Figure 3.4. Growth of infected nodes, $I(t)$, with time for different values of ρ (data infectivity) : (a) Trickle (avg degree = 5), (b) Trickle (avg degree = 8), (c) Firecracker (avg degree = 5, Route to Opp Corners), (d) Firecracker (avg degree = 5, Route to All Corners), (e) Deluge (avg degree = 5), (f) Deluge (avg degree = 8), (g) MNP (avg degree = 5), (h) MNP (avg degree = 8).

increasing the number of strategically placed source nodes to spread the infection, has a significant impact on the subsequent rate of spread. We observe that routing the data to three corners instead of just the opposite two almost halves the compromise time of the whole network.

Deluge and MNP are meant for bulk transfer of data, and propagate one page at a time. This makes its spreading rate slower than Trickle or Firecracker. This is duly captured in our model as depicted in figs. 3.4 (e) and (f) for Deluge and figs. 3.4 (g) and (h) for MNP. From our model, we can get a fair picture of the temporal propagation when we compare protocols falling in the same class. Thus, comparing between Deluge and MNP, we observe how MNP's propagation process is slowed down by its sender selection algorithm. This is manifested in the fact that an advertising node collects requests and waits for a threshold number of advertisements before starting to service them. It is during this time that its sender selection procedure chooses a particular node as a sender.

An important observation of our model is that, contrary to networks with homogeneous mixing, in a sensor network with limited spatial interactions between nodes, there is no distinct phase transition point of the infection. This is probably because the spread rides on top of a controlled broadcast protocol and the propagation happens along a circular front which is spatially bounded. This means that, contrary to conventional epidemic flooding, the propagating data has to abide by the discipline imposed by the communication pattern of these protocols which try to minimize unnecessary transmissions to save energy.

As expected, our model captures the fact that Firecracker achieves the highest propagation rate among the protocols discussed and thus poses a very high threat for malware transfer in the event of an outbreak. Comparing figs. 3.4 (a) and (c), we observe that even with a lower node degree, Firecracker, with route points as opposite corners, achieves network compromise in about 25% to 40% of the time as

Trickle. However, comparing figs. 3.4 (e) and (g), we observe that MNP with an advertisement threshold of 4, takes almost 70% more time than taken by Deluge.

We thus observe that our model can successfully capture the propagation process and shed significant light on the temporal dynamics of how a piece of information could spread over current broadcast protocols. However, at this point we would also like to acknowledge a few limitations of our model. One of them is that our model fails to capture border effects in the network. As a result, it generally performs better when the spread is happening from the center outwards and loses accuracy when edge effects become significant.

Our model also shows inaccuracies when the density of the network becomes very high. Although, we have considered the physical effects of the network and the packet loss due to collisions, at very high densities, the hidden terminal problem becomes significant, especially in protocols like Deluge. Consequently, our model cannot capture it effectively. For instance, when we increase the average degree of the network from 5 to 8 our model observes an increase in the rate of propagation of the malware. This is in accordance with the fact that an infected node gets more susceptible nodes to infect. However, if the density and subsequently the node degree becomes very high, then the spreading rate would decrease because of the increase of the number of collisions. In such a scenario, our model would then require to be adequately tuned with the correct packet loss probability in order to accurately model the infection propagation.

Within its limitations, our model could be a handy tool to assess each broadcast protocol quickly to gather approximate knowledge of its vulnerability to malwares of different infectivity.

3.4 Simulation Study

In this section, we outline the simulation setup and details to implement the propagation process in each of the three broadcast protocols. The time dynamics of the data propagation is captured with varying degrees of infectivity on the whole network. We have used JProwler [89], a probabilistic, event-driven wireless network simulator written in Java, for our experiments.

In the simulation experiments, we assume $N = 1000$ sensor nodes with uniform random deployment in the network. We have used the default radio model in JProwler where all communication links in the network are symmetric and deterministic. Packets are lost only when there is a collision. The maximum data rate of wireless links is set to be 32 Kbps. The maximum length of a packet is fixed at 40 bytes. The MAC protocol is based on a simple CSMA scheme like BMAC [66]. The metrics for evaluating the proposed framework is the time it takes for a malware to infect a given fraction of the network, spreading over each broadcast protocol. Each reported result is averaged over twenty simulation runs.

Our simulation works in two phases. In the first phase, we form the network where each node identifies its set of neighbors and entries are made into a neighbor table. By randomly choosing links to keep or delete, we control the average degree of the network. We perform this by modifying the transmit power of individual nodes so as to change the average number of total links in the network.The entry for each node in the neighborhood table can indicate whether a node is susceptible, infected or recovered. The average node degree of the network is set to typical values of 5 and 8.

In the second phase, we simulate actual data propagation over each broadcast protocol. Initially, at $t = 0$, the number of infected nodes, denoted by $I(0)$ is set to be 1. The initially infected node is selected as a corner node of the whole network.

The time period T_p, of Trickle, has been assumed to be the unit and is equal to 1 second in our simulation.

First, we performed the simulation study for the case where there is no recovery against different values of the malware infectivity, ρ. We simulate under different network connectivities and study the time dynamics of the infected population. For each susceptible neighbor of an infective node to which a data packet is to be transmitted, malware transmission is done based on the probability ρ, independently for each node. A node, once infected, stays infected for the rest of the simulation time. Fig. 3.5 shows the simulation results for the propagation dynamics over each broadcast protocol against different values for the malware infectivity. We observe that the nature of the curves closely match our analytical model.

We also observe some discrepancies between our simulation and analytical results as is evident from figs. 3.4 and 3.5. This is attributed to the fact that the differential equation based approach approximates the process to be continuous in time which is not the case in our simulation. Moreover, our model does not incorporate border or edge effects and the infection is assumed to propagate from the center outwards. With a considerable increase in the density of the network, our simulation results would deviate significantly from the analytical results. This is attributed to the error in the packet loss probability that creeps in such scenarios. Our model would then have to be tuned accordingly so that the packet loss probability can effectively capture the impact of high density.

3.5 Summary

In this chapter, we provide a common mathematical model to analyze the process of information propagation over different multihop broadcast protocols. Although approximately, our model successfully captures the ripple based propagation behavior

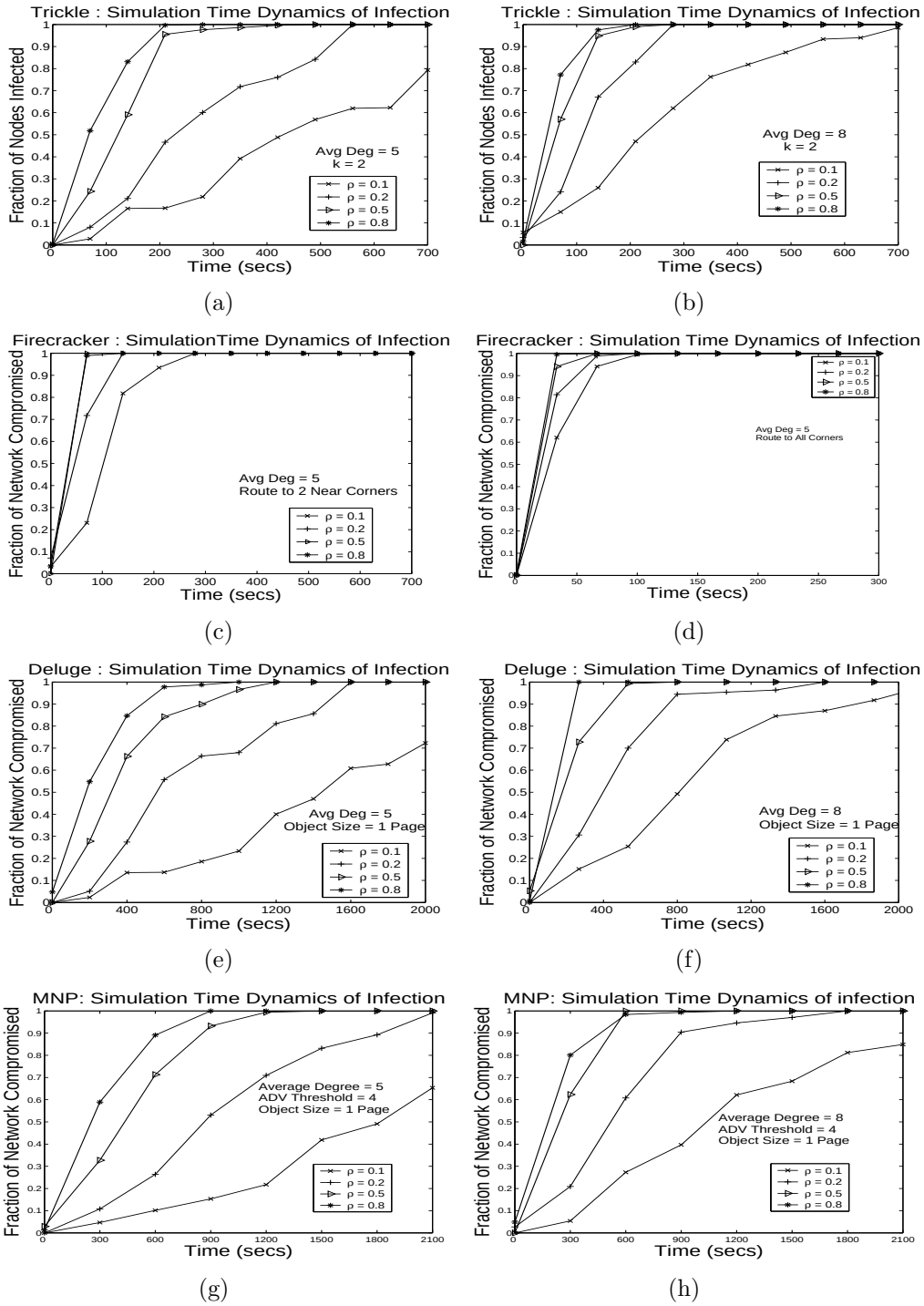


Figure 3.5. Growth of infected nodes, $I(t)$, with time for different values of ρ (data infectivity) : (a) Trickle (avg degree = 5), (b) Trickle (avg degree = 8), (c) Firecracker (avg degree = 5, Route to Opp Corners), (d) Firecracker (avg degree = 5, Route to All Corners), (e) Deluge (avg degree = 5), (f) Deluge (avg degree = 8), (g) MNP (avg degree = 5), (h) MNP (avg degree = 8).

of the wavefront of a broadcast protocol. Not only is the model capable of assessing the performance of each protocol for disseminating data, but it also helps in comparing their performances against each other. Its generic and flexible nature allows us to conveniently fit parameters of different broadcast protocols and analyze their susceptibilities. Despite the similarities in operation between some of the protocols discussed, the epidemic model successfully highlights their differences from a propagation standpoint. The model can also be extended to other complex broadcast protocols by successfully computing the infection rate β for that protocol. In other words, in this chapter, we have computed a model for data propagation over broadcast protocols and their temporal dynamics under different states of network connectivity. In chapter 5, we investigate the vulnerability of these protocols against piggybacked malware spread in a sensor network.

CHAPTER 4

MODELING OF NODE COMPROMISE PROPAGATION

In chapter 3, we discussed the data propagation behavior of broadcast based dissemination protocols. Our model helped us capture the spreading characteristics of each protocol in terms of spreading rate and network reachability by way of parametrically fitting the protocol's communication behavior into the epidemic infection spread model.

We now move on to consider the security aspects of this paradigm of communication and investigate different scenarios where the security of a sensor network could be breached. Moreover, we also analyse the drastic repercussions of how a small or negligible breach in security (compromise of a single node) could possibly spread across the network and gain epidemic proportions.

In this chapter, we look into the propagation process of malware throughout a sensor network where communication between the nodes is secure [18]. In particular, communication is only performed among neighboring nodes when they have established mutual trust by authenticating a common key. Starting from a single point of failure of a compromised node, we investigate the potential of an adversary to effectively compromise neighboring nodes through wireless communication and thus threat the whole network without engaging in full scale physical attacks.

Here, we do not assume the malware to possess the advantage of piggybacking itself on a broadcast protocol, but rather have the necessary communication component to infect a neighboring node. However, each link of communication is authenticated by pre-distributed secret keys. In this study of propagation of node compromise, we not only incorporate authenticated secure communication among the nodes but

also the effects of different kinds of deployment strategies of the sensor nodes in the terrain.

By incorporating these factors of the networks, we propose an epidemiological model to investigate the probability of a breakout (compromise of the whole network) and if not, the sizes of the affected components (compromised clusters of nodes). Furthermore, we analyze the effect of node recovery in an active infection scenario and obtain critical values for these parameters that result in an outbreak. We focus our analysis on two specific types of node deployment scenarios, namely uniform random deployment and group based deployment of nodes (where the actual resident points of the nodes of a group are assumed to follow a particular spatial distribution about the group deployment point).

This chapter is organized as follows : In section 4.1, we perform a preliminary discussion about pairwise key distribution in sensor networks. In section 4.2, we present our model and discuss it under different scenarios of with and without node recovery as well as under different node deployment strategies. In section 4.3, we discuss about a critical parameter in Epidemic Theory, namely, the Basic Reproductive Number. We discuss our simulation study in section 4.4 and summarize in section 4.5.

4.1 Pairwise Key Pre-distribution

In this section, we briefly overview the pairwise key scheme for securing the communication between neighboring nodes in a sensor network based on key pre-distribution. This shared secret key based secure communication is especially popular in sensor networks owing to the prohibitive resource consumption of most public key cryptographic techniques.

Due to the severe resource constraint of wireless sensor networks and limited networking bandwidth, proposed pairwise key schemes have commonly adopted the pre-distribution approach instead of online key management schemes. The concept of pre-distribution was originated from [28], where the authors propose to assign a number of keys, termed *key ring* randomly drawn from a key pool. If two neighboring nodes share a common key on their key rings, a shared pairwise key exists and a secure communication can be established. An enhanced scheme termed *Q-composite* was proposed in [13], where two neighboring nodes can establish secure communication only if at least Q keys are shared on their key rings. Pre-distribution schemes that rely on bivariate polynomials is discussed in [54]. In this scheme, each sensor node is pre-distributed a set of polynomials. Two sensor nodes with the same polynomials can respectively derive the same key.

Regardless of the specific key distribution scheme, a common parameter capturing the performance is the probability that two neighbors can directly establish a secure communication or, in other words, share at least one key. We denote this key sharing probability by q. Thus, two physical neighbors can communicate securely with probability q. The factors on which q depends, such as the key pool size or the individual key ring sizes, have been studied in previous works [28, 13]. The value of q is crucial in controlling the degree of connectivity of the securely communicating sensor network. As we will reveal later, q plays an important role in the spreading of node compromise, as direct communication (as explained subsequently, in the threat model) can result in propagation of malicious code. A high value of q would make the network highly connected while at the same time increase the network's susceptibility to compromise propagation.

4.2 Modeling and Analysis of Compromise Propagation

In this section, we analyze the propagation of node compromise originating from a single node that has been affected. Our focus is to study the outbreak point of the epidemic effect where the whole network will fall victim to the compromise procedure.

Our key method is to characterize the sensor network, including its key distribution, by mathematically formulating it as a random graph whose key parameters are precisely determined by those of the sensor network. Therefore, the investigation of epidemic phenomena can be performed on the random graph instead.

We perform our analysis on two types of sensor network topology models. In our first model, we assume that the sensor nodes are uniformly randomly distributed. In our second model, we assume a more realistic scenario where deployment knowledge is incorporated in the analysis. We assume that nodes are deployed in groups and the resident points of each node in a group follows a two dimensional gaussian distribution about the deployment point. Subsequently, given these two deployment approaches, we observe the epidemic process under two scenarios: without node recovery and with node recovery, depending on whether infected nodes will be recovered by external measures like key revocation, immunization, and so on.

Our goal is to analyze and compare epidemic spread progress and effect under different topological scenarios of the sensor network. In the following two subsections, we derive the degree distribution for the two random graph deployment models, viz. uniform random distribution and groups of gaussian random distribution.

4.2.1 Network Model

In this section, we will model the network topology of the overlay key sharing graph above the physical network. The outcome of our model is the degree distribution of the key sharing overlay topology. In our analytical derivation, we incorporate

the deployment knowledge of the sensor nodes by using different deployment models to finally derive the network degree distribution.

4.2.1.1 Uniform Random Distribution

Assume that sensor nodes are uniformly randomly deployed in a region with area A. Let $\sigma = \frac{N}{A}$ denote the node density of the network where N is the total number of the nodes. For a sensor node with communication range R, the probability that l nodes are within its communication range is given by

$$p(l) = \binom{N-1}{l} p^l (1-p)^{N-1-l} \tag{4.1}$$

where p is defined by

$$p = \frac{\pi R^2}{A} = \frac{\pi R^2 \sigma}{N}. \tag{4.2}$$

Thus p is the probability of a link existing at the physical level, i.e., whether the two nodes fall within their respective communication ranges.

We further assume that the probability that two neighboring nodes sharing at least one key in the random pre-distribution pairwise key is q. Notice that q is determined by the specific pairwise key scheme employed. For a particular node having l neighboring nodes, the probability that there are k nodes, $k \leq l$, sharing at least one key with it is given by

$$p(k|l) = \binom{l}{k} q^k (1-q)^{l-k} \tag{4.3}$$

Therefore, with uniform random deployment, the probability of having k neighboring nodes sharing at least one key is

$$p_u(k) = \sum_{l=k}^{N-1} p(l)p(k|l) \tag{4.4}$$

$$= \sum_{l=k}^{N-1} \binom{N-1}{l} p^l (1-p)^{N-1-l} \binom{l}{k} q^k (1-q)^{l-k} \tag{4.5}$$

4.2.1.2 Group based Deployment Model : Two Dimensional Gaussian Random Distribution

The other deployment model that we consider in this chapter is group based deployment [26, 82]. In this model, sensors are divided into groups where each group is deployed (e.g., dropped from an airplane) at a particular location. Due to the uncertainty of the deployment procedure, sensor nodes within each group are often randomly distributed around the targeted deployment point. Specifically, we make the following assumptions about this model.

- N sensor nodes to be deployed are divided into t equal size groups each with n nodes. Each group, G_i, for $x_i = 1, \ldots, t$ and $y_i = 1, \ldots, n$, is deployed from the deployment point (x_i, y_i).
- The deployment points are assumed to be arranged in a grid, which is commonly assumed.
- During deployment, the resident points of the node k in group G_i with deployment point (x_i, y_i) follow probability distribution $f_k^i(x, y|k \in G_i)$. An example of this distribution is a two-dimensional Gaussian distribution around the deployment point.

In other words, when the deployment point of group G_i is at (x_i, y_i), we have the mean position $\mu = (x_i, y_i)$ and the pdf for node k in group G_i as

$$f_k^i(x, y|k \in G_i) = \frac{1}{2\pi {\sigma_d}^2} e^{-[(x-x_i)^2+(y-y_i)^2]/2{\sigma_d}^2}, \tag{4.6}$$

where σ_d denotes the standard deviation. Given such a group based Gaussian deployment model, we formulate the degree distribution of the key sharing graph based on both the deployment and the key distribution mechanism.

Let us consider any node location $H = (x, y)$ in the rectangular deployment field. Therefore, the probability that a node n_i from group i with deployment point (x_i, y_i) resides within the rectangular area $dxdy$ centered at point H is given by

$$\frac{1}{2\pi {\sigma_d}^2} \exp -\frac{(d_i H)^2}{2{\sigma_d}^2} \cdot dxdy$$

where $(d_i H)^2 = (x - x_i)^2 + (y - y_i)^2$. We denote

$$f(d_i H|n_i \in i) = \frac{1}{2\pi {\sigma_d}^2} e^{-\frac{(d_i H)^2}{2{\sigma_d}^2}} \tag{4.7}$$

and have the following result.

Lemma 4.1. *If $g(x, y|j)$ denotes the probability that a node n_j from group j is within transmission radius R of point $H = (x, y)$, then $g(x, y|j) = \mathbf{1}\{h < R\} \left[1 - e^{-\frac{(R-h)^2}{2{\sigma_d}^2}}\right] + \int_{|h-R|}^{h+R} f(l|n_j \in j) \cdot 2lcos^{-1}\left(\frac{l^2+h^2-R^2}{2lh}\right) dl$, where $\mathbf{1}\{\cdot\}$ is the set indicator function.*

Proof. When a sensor node resides at the point $H = (x, y)$ as shown in Fig. 4.1, the probability that the sensor node n_j from group j resides within the circle centered at location H with radius R is defined as $g(h|n_j \in \text{group}j)$, where $h = d_j H$, is the

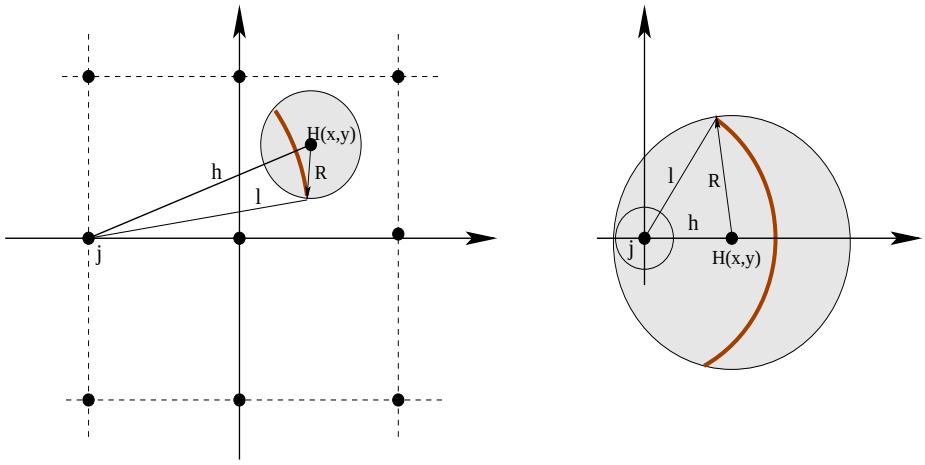


Figure 4.1. Deployment points and resident point distribution.

distance between H and the deployment point of group j. When $h > R$, as shown in the first diagram of Fig. 4.1,

$$g(x,y|j) = \int_{|h-R|}^{h+R} f(l|n_j \in j) \cdot 2lcos^{-1}\left(\frac{l^2+h^2-R^2}{2lh}\right) dl,$$

where the length of arc of the ring centered at j is calculated and then integrated over all possible values of l.

When $h < R$, as shown in the second diagram of Fig. 4.1,

$$g(x,y|j) = \int_0^{R-h} l \cdot 2\pi f(l|n_j \in j)dl + \int_{R-h}^{R+h} 2lcos^{-1}\left(\frac{l^2+h^2-R^2}{2lh}\right) f(l|n_j \in j)dl.$$

Thus,

$$g(x,y|j) = \mathbf{1}\{h < R\}\left[1 - e^{-\frac{(R-h)^2}{2\sigma_d{}^2}}\right] + \int_{|h-R|}^{h+R} f(l|n_j \in j) \cdot 2lcos^{-1}\left(\tfrac{l^2+h^2-R^2}{2lh}\right) dl,$$

where $\mathbf{1}\{\cdot\}$ is the set indicator function whose value is 1 when the evaluated condition is true and 0 otherwise, and $f(l|n_j \in j)$ is given by Eqn. 4.7. □

Theorem 4.1. *The probability distribution of the degree of the key sharing topology, $p_g(k)$, assuming that the nodes are deployed in groups and reside according to a two dimensional gaussian distribution around the deployment points, is given by* $p_g(k) =$

$\int_0^Y \int_0^X \sum_{l=k}^{N} \binom{l}{k} q^k (1-q)^{l-k} N_b(l,x,y)$ *where* $N_b(l,x,y)$ *is the probability that a node is at point* (x,y) *and it has* l *neighboring nodes.*

Proof. Let $d = g(x,y)$ denote the probability that a node resides within a radius R of point (x,y). Then from Lemma 1, we get

$$d = g(x,y) = \sum_j g(x,y|j) Pr[j] \tag{4.8}$$

where $Pr[j]$ is the probability that a deployed node belongs to group j. We assume that a sensor node is selected to be in each group with an equal probability and is equal to $\frac{1}{tn}$.

Let $p(l|x,y)$ be the probability that there are l nodes within radius R of (x,y). Therefore,

$$p(l|x,y) = \binom{N}{l} d^l (1-d)^{N-l} \tag{4.9}$$

where N is the total number of nodes deployed. The probability that a deployed node is at point $H = (x,y)$, is given by

$$\sum_i f(d_i H | n_i \in i) \cdot Pr[i] dxdy$$

Let $N_b(l,x,y)$ be the probability that a node is at (x,y) and it has l neighboring nodes. Thus,

$$N_b(l,x,y) = p(l|x,y) \sum_i f(d_i H | n_i \in i) \cdot Pr[i] dxdy \tag{4.10}$$

Now, let $p_g(k|l,x,y)$ be the probability that a node which is located at (x,y) and has l neighbors shares keys with exactly k neighbors. Hence,

$$p_g(k|l,x,y) = \binom{l}{k} q^k (1-q)^{l-k} \tag{4.11}$$

where q is the key sharing probability. Therefore,

$$p_g(k,x,y) = \sum_{l=k}^{N} p_g(k|l,x,y) N_b(l,x,y) \tag{4.12}$$

Thus, integrating over the entire region, the degree distribution of a group based deployed network with each group deployed in a gaussian manner, is given by

$$p_g(k) = \int_0^Y \int_0^X p_g(k,x,y) = \int_0^Y \int_0^X \sum_{l=k}^{N} p_g(k|l,x,y) N_b(l,x,y) \tag{4.13}$$

□

Thus, based on both physical proximity and the probability of key sharing between neighbors, we get a degree distribution $p(k)$ for each of the graphs representing the two different deployment strategies. We will now perform our epidemic propagation analysis on these two types of random networks under the two scenarios of node recovery and without node recovery. The random graph in our analysis is denoted by G, and $p_u(k)$ (for uniform deployment) and $p_g(k)$ (for group based deployment) characterize the degree distribution under the respective deployment strategies.

4.2.2 Network Connectivity

Before we consider our analysis of the epidemic processes on the overlay key sharing graph of the sensor network, it is essential to ensure that the graph is connected. We borrow the results from the works on connectivity in ad hoc networks

presented in [6, 63]. From their results, a geometric random graph with N nodes is k-connected with probability $P(k\text{-connected}) = \left(1 - \sum_{i=0}^{k-1} \frac{\left(\sigma\pi R^2\right)^i}{i!} \cdot e^{-\sigma\pi R^2}\right)^N$, where σ is the network density and R is the transmission radius. Since we are considering 1-connectivity, the probability that the graph is connected is given by

$$P(connected) = \left(1 - e^{-\sigma\pi R^2}\right)^N \tag{4.14}$$

Our goal is to make this probability almost equal to 1.

Without loss of generality on the deployment strategy, let $p(k)$ denote the degree distribution of the key sharing topology. Thus, $\eta = \sum_{k=1}^{N-1} kp(k)$ is the expected degree of the network. We observe that, in the expression for the aforementioned probability of connectivity of the network, $\sigma\pi R^2$ is the expected number of neighbors that a node has. In other words, it can be interpreted as the expected number of neighboring nodes that fall within the transmission range of a given node. Thus, for our network with expected degree denoted by η, the probability that the network is connected is given by

$$P(connected) = \left(1 - e^{-\eta}\right)^N \tag{4.15}$$

For our analysis, we consider the minimum value of the key sharing probability q to be such that it is *well above* the threshold in order to keep the network connected with very high probability.

4.2.3 Compromise Spread Without Node Recovery

Given the random graph construction based on the two deployment strategies, we now analyze the case of compromise spread when no node recovery is performed. In other words, a compromised sensor node will remain infectious indefinitely.

Let $G_0(x)$ be the generating function of the degree distribution $p(k)$ of a randomly chosen vertex in G and is defined by

$$G_0(x) = \sum_{k=0}^{\infty} p(k)x^k \tag{4.16}$$

The average degree z of G is denoted by

$$z = \sum_k kp(k) = G_0'(1) \tag{4.17}$$

Similarly, $G_1(x)$ denotes the degree distribution of the vertices at the end of randomly chosen edges. The distribution of degrees of vertices reached by following edges is proportional to $kp(k)$ and thus the generating function for those degrees is

$$\frac{\sum_k kp(k)x^k}{\sum_k kp(k)} = \frac{xG_0'(x)}{G_0'(1)} \tag{4.18}$$

To elucidate further $G_1(x)$ represents the distribution of the number of ways of leaving these vertices excluding the edge we come along, which is the degree minus 1 and is given by

$$G_1(x) = \frac{1}{z}G_0'(x) \tag{4.19}$$

If λ denotes the infection probability of a node being infected by communicating with a compromised node, then the number c of compromised edges around a randomly chosen vertex is generated by

$$\begin{aligned} G_0(x;\lambda) &= \sum_{c=0}^{\infty}\sum_{k=c}^{\infty} p(k)\binom{k}{c}\lambda^c(1-\lambda)^{k-c}x^c \\ &= \sum_{k=0}^{\infty} p(k)\sum_{c=0}^{k}\binom{k}{c}(x\lambda)^c(1-\lambda)^{k-c} = \sum_{k=0}^{\infty} p(k)(1-\lambda+x\lambda)^k \\ &= G_0(1-\lambda+x\lambda). \end{aligned} \tag{4.20}$$

Similarly, $G_1(x;\lambda) = G_1(1-\lambda+x\lambda)$.

Let $H_1(x;\lambda)$ be the generating function that denotes the distribution of the sizes of the cluster of vertices or components reached by following a randomly chosen edge that is compromised. The degree of such an end vertex can vary from 0 to $N-1$. Moreover, if the degree is at least one, then following each edge out of that vertex would lead to more vertices whose degree distribution is also $H_1(x;\lambda)$. If there are k edges emanating from the vertex at the other end of the random edge, then the distribution of the sum of the sizes of the k clusters that each edge from the end vertex leads to, is given by $H_1(x;\lambda)^k$.

The generating function $H_1(x;\lambda)$ for the total number of nodes reachable or compromised as a result of a single transmission along an edge of the network is, thus, generated by a self consistency relation of the form [59, 60]

$$H_1(x;\lambda) = xG_1(H_1(x;\lambda);\lambda). \tag{4.21}$$

We are interested in the distribution of the size of the component to which a randomly chosen vertex belongs. In other words, the distribution of the number of

nodes affected by an outbreak when the infection starts at a single infective node is generated by

$$H_0(x;\lambda) = x\sum_{k=0}^{N-1} p(k)[H_1(x;\lambda)]^k = xG_0(H_1(x;\lambda);\lambda). \tag{4.22}$$

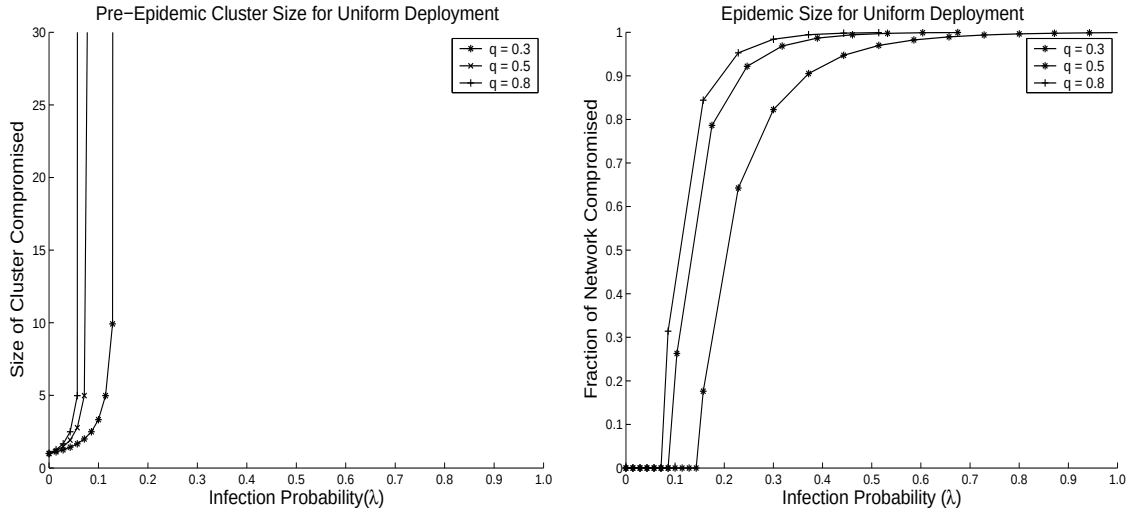
The average size of the outbreak cluster is derived as $s = H_0'(1;\lambda)$ and is given by

$$s = 1 + \frac{\lambda G_0'(1)}{1 - \lambda G_1'(1)}. \tag{4.23}$$

Infection probability λ essentially captures the spreading capability of the virus that could compromise the network: the larger it is, the stronger the virus is. We assume that its value can be obtained by means of measurement or analysis.

We remark here that in traditional epidemiology, the parameter λ, denoting the infection probability, generally represents the portion of the population that is susceptible to the infection. However, in a sensor network, it is typically assumed that all the nodes are homogeneous and therefore equally susceptible. Thus, in this case, instead of considering a fraction of the network as susceptible, we consider the whole network to be susceptible and subsequently, at t=0, all $N-1$ nodes are susceptible with one node being infected. The parameter λ, in this case, tries to capture the characteristic of the malware that is spreading and what technique it adopts, i.e., whether it has the properties of a worm, virus, or trojan, etc. In other words, using the variable λ, we try to parametrically capture the infectivity of a malware or the probability by which an infection spreads on a link between any pair of infected and susceptible node.

Thus, λ succeeds in differentiating between different malwares and their propagation characteristics and is assumed to be fixed for a particular spread but may



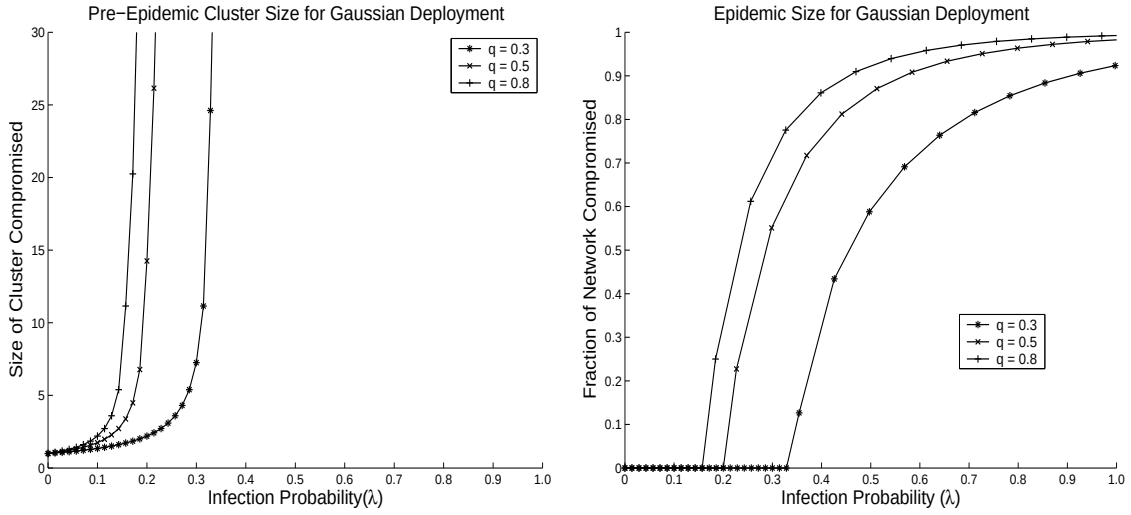
(a) (b)

Figure 4.2. Size of compromised node clusters with infection probability (λ) for uniform random deployment: (a) depicts the average size of infected clusters when there is no epidemic and (b) shows the epidemic size as the fraction of the entire network. The point where non-zero value appears indicates the transition from non-epidemic to epidemic.

vary from time to time based on the type of malware and what systemic technique it adopts to spread.

We, thus, use λ to connect the infection probability of the malware to the physical properties of the network (expressed in terms of p and q) and see if there is a resultant epidemic.

Given the above result, we can see that the outbreak point for the network is $\lambda = 1/G_1'(1)$ which marks the onset of an epidemic. For $\lambda > 1/G_1'(1)$ we have an epidemic in the form of a giant component in the random network. We observe that $H_0(1;\lambda)$ which is the distribution of the cluster formation is only valid below the threshold point beyond which it becomes invalid because in a giant component there could be loops and the recursive distribution of the end node degree of an edge as stated by Eqn. 4.21 would not hold. Thus, beyond the threshold point, we define $H_0(1;\lambda)$ to be the distribution of isolated clusters which do not have loop formations.



(a) (b)

Figure 4.3. Size of compromised node clusters for Gaussian Deployment: (a) depicts the average size of infected clusters when there is no epidemic and (b) shows the epidemic size as the fraction of the entire network. The point where non-zero value appears indicates the transition from non-epidemic to epidemic.

If Ψ_m denotes the cluster size distribution of size m, then we observe that the fraction of the network forming the giant component is given by $S = 1 - \sum_m \Psi_m = 1 - H_0(1; \lambda)$. Rearranging and substituting from Eqn 4.22, we have $S = 1 - G_0(u; \lambda)$.

Here u is the root of the self-consistency relation

$$u = G_1(u; \lambda).$$

Intuitively, the above conclusion reveals that if $\lambda \leq 1/G_1'(1)$, the component of compromised nodes is finite in size regardless of the size of the network and each node's probability of being compromised is zero for large networks. On the contrary, if $\lambda > 1/G_1'(1)$, there always exists a finite probability for a node to be compromised.

We plot the effect of different key sharing probabilities on the epidemic outbreak on our two deployment strategies, viz uniform random and a collection of group based deployment strategies. Fig. 4.2 depicts this effect for a uniform random network with

$N = 10000$ nodes deployed in a $600 \times 600 unit^2$ area with different key sharing probabilities q. The underlying physical topology is determined by the communication range of each node which is equal to 20 units. Given the physical deployment, we vary the probability of direct pairwise key sharing (q) and study the point of outbreak. As we can see in Fig. 4.2, while undoubtedly increasing q can facilitate communication in the network, the network also becomes more vulnerable to virus spreading. Specifically, when $q = 0.3$, network wide breakout is only possible when a compromised node has an infection probability (λ) larger than 0.15 to infect a neighbor. We note that in this case, we have an expected node degree of 15. On the contrary, λ only needs to be around 0.05 when $q = 0.8$ which subsequently makes the expected node degree around 30. Fig. 4.2(b) illustrates the fraction of the network that is ultimately infected as the infection probability is increased beyond the critical point of the onset of outbreak. For instance, we observe that when key sharing probability is high ($q = 0.8$), the whole network is compromised with a λ value of around 0.4. On the contrary, with $q = 0.3$, the network could be compromised with only a high value of $\lambda = 0.7$. Although Fig. 4.2 (b) is a continuation of Fig. 4.2 (a) when the epidemic spreads to the entire network, a separate depiction provides a clear picture of the extent to which the network is infected before and after an epidemic. In summary, Fig. 4.2 clearly indicates the tradeoff between key sharing probability among sensor nodes and the vulnerability of the network to compromise.

In Fig. 4.3, we depict the same process with the deployment scheme changed to a group based one. The deployment points are arranged in a 10×10 grid with 100 nodes per group. The nodes in each group reside in a two-dimensional gaussian manner about their mean deployment point with $\sigma_d = 10$. We observe that the potency of the propagation process is affected by the change in deployment. The λ values at which the epidemic starts to spread into the whole network has increased.

For instance, for $q = 0.3$, the transition point is around $\lambda = 0.32$ as compared to be around 0.15 for the uniform deployment case. The reason for the decrease in the epidemic effect is caused by the fact that the expected node degree of the network is lowered when the nodes are deployed in groups distributed in a gaussian manner. This, obviously, has a crippling effect on the propagation process and thus helps in delaying the onset of the epidemic. However, this effect, as expected, slowly diminishes with increase in the variance of the gaussian distribution of each group, which gradually pushes the distribution to a more uniform nature. Thus, when we increase σ_d to around 40 in our deployment scenario with 10000 nodes, there is practically little difference between the two deployment strategies. This analysis shows that tuning the deployment parameters to a certain extent could result in making the network robust against viral propagation without considerably hampering its connectivity.

4.2.4 Compromise Spread With Node Recovery

In this case, we assume that the network has the capability to recover some of the compromised nodes by either immunization or removal from the network. To capture this recovery effect, we assume that an infected node recovers or is removed from the network after an average duration of infectivity τ. In other words, a node in the sensor network remains infective for an average period τ after which it is immunized. During this infective period, the node transmits the epidemic to its neighbors with the infection rate β, denoting the probability of infection per unit time. Evidently, the parameter τ is critical to the analysis as it measures how soon a compromised node recovers. Naturally, we will perform our analysis following the SIR model in epidemic theory [31, 60].

First, consider a pair of adjacent nodes where one is infected and the other is susceptible. We define T as the compromise transmission probability, or in other words, the *transmissibility* of the infection. Given the above definitions for β and

τ, we can say that the probability that the disease will not be transmitted from the infected to the susceptible is given by

$$1 - T = \lim_{\delta t \to 0} (1 - \beta \delta t)^{\tau/\delta t} = e^{-\beta\tau}. \tag{4.24}$$

Subsequently, we have the transmission probability

$$T = 1 - e^{-\beta\tau}.$$

In other words, the compromise propagation can be considered as a Poisson process, with average $\beta\tau$. The outcome of this process is the same as bond percolation and T is basically analogous to the bond occupation probability on the graph representing the key sharing network. Thus, the outbreak size would be precisely the size of the cluster of vertices that can be reached from the initial vertex (infected node) by traversing only occupied edges which are occupied with probability T. Notice that T explicitly captures node recovery in terms of the parameter τ.

Replacing λ with T in Equation 4.23, and following similar steps, we get the size of the average cluster as

$$s = 1 + \frac{TG_0'(1)}{1 - TG_1'(1)}. \tag{4.25}$$

and the epidemic size is obtained by

$$S = 1 - G_0(u; T). \tag{4.26}$$

where u is obtained by

$$u = 1 - G_1(u; T), \tag{4.27}$$

and $G_0(u;T)$ and $G_1(u;T)$ are given respectively by

$$G_0(u;T) = G_0(1 + (u-1)T), \tag{4.28}$$

and

$$G_1(u;T) = G_1(1 + (u-1)T). \tag{4.29}$$

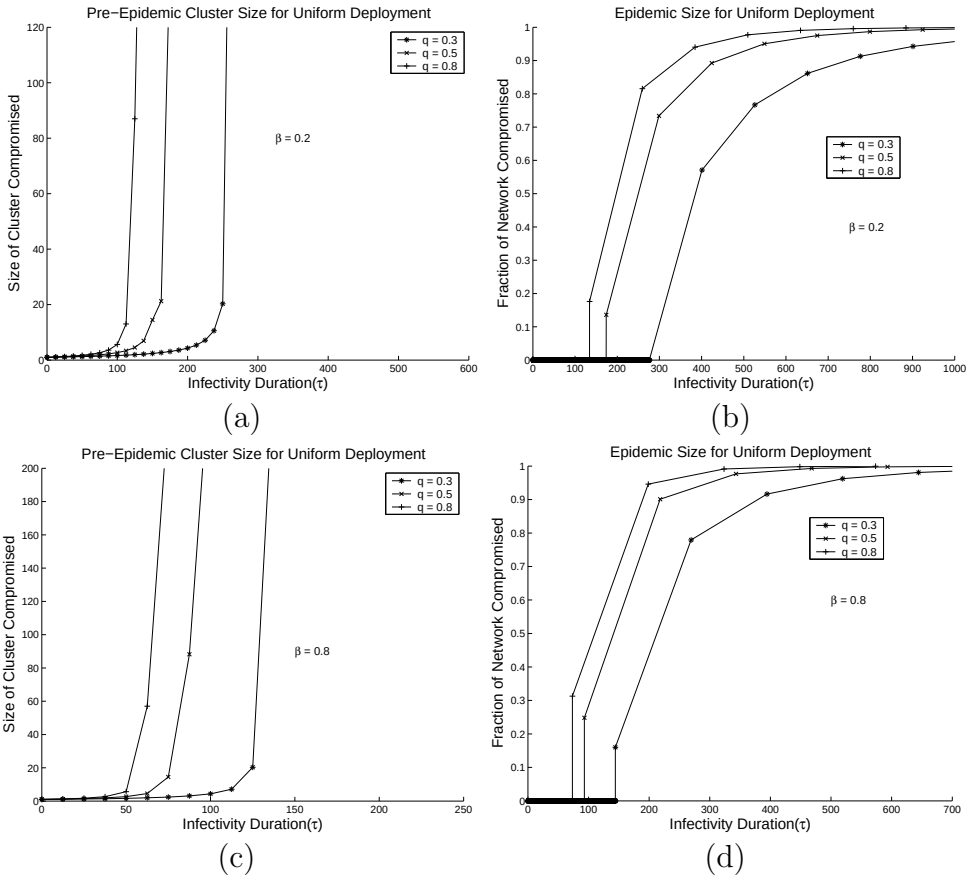


Figure 4.4. Extent of Epidemic Size with Varying Infectivity Duration (Uniform Deployment): (a) Pre-Epidemic Cluster Size with Low Infection Probability (b) Post-Epidemic Infected Fraction with Low Infection Probability (c) Pre-Epidemic Cluster Size with High Infection Probability (d) Post-Epidemic Infected Fraction with High Infection Probability.

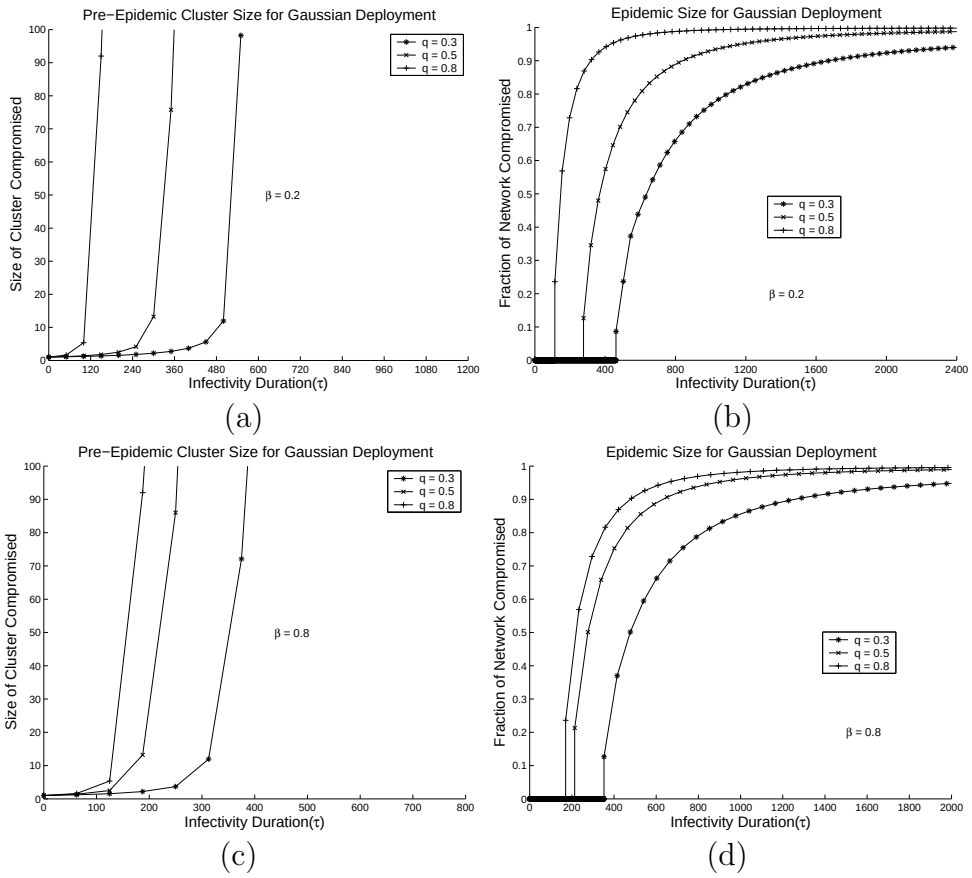


Figure 4.5. Extent of Epidemic Size with Varying Infectivity Duration (Gaussian Deployment): (a) Pre-Epidemic Cluster Size with Low Infection Probability (b) Post-Epidemic Infected Fraction with Low Infection Probability (c) Pre-Epidemic Cluster Size with High Infection Probability (d) Post-Epidemic Infected Fraction with High Infection Probability.

Figs. 4.4 and 4.5 summarize this effect for the uniform and group based deployment respectively. They depict the epidemic outbreak against the average recovery time τ for low and high infection rates $\beta = 0.2$ and $\beta = 0.8$. The network setup is the same as before with $N = 10000$ and a 10×10 grid for the group deployment scheme. For uniform deployment, Figs. 4.4(a) and (b) depict the pre-epidemic and post-epidemic scenario when the infection rate is low ($\beta = 0.2$). In Figs. 4.4(c) and (d), the infection rate is high ($\beta = 0.8$). The plots are for different values of the key sharing probability q which governs the connectivity of the key sharing sensor topology. Fig. 4.5 shows the plots when the nodes are deployed in groups. Comparing

the plots in Figs. 4.4 and 4.5 we observe similar characteristics as observed in the non-recovery case. Here, in the group based plots, the compromise process attains epidemic proportions at higher values of the infectivity duration τ. For instance, comparing Fig. 4.4(d) and Fig. 4.5(d), when $q = 0.3$ and $\beta = 0.8$, the epidemic outbreak for the group deployment starts at around $\tau = 500$, whereas in the uniform deployment scenario, its onset is when τ is around 150. This indicates that the potency of the compromise is reduced by the gaussian distribution deployment model. In other words, the nodes in the group based deployment model would need to stay infective for a larger average duration in order to spread to the entire network than when uniformly deployed. The expected duration of a node's infectivity could be a possible measure of the degree of potency of the viral infection and this comparison is indicative of which deployment scheme is better poised to resist against an outbreak.

In next section we will discuss about an important parameter in epidemic theory whose value indicates whether an infection can potentially result in an epidemic. We derive an expression for this parameter for our sensor network model and investigate its behavior.

4.3 Basic Reproductive Number

In epidemiology, the *Basic Reproductive Number* R_0 is defined as the expected number of people that a single infective individual can infect in a pool of mostly susceptible candidates. Its importance lies in the fact that it characterizes the epidemic growth at the start of an outbreak: the infection will eventually die out when $R_0 < 1$ and when $R_0 > 1$, the disease will spread exponentially and consequently may lead to a large epidemic.

Since we have proposed an epidemic model for the spread of node compromise, it is essential that we verify how our network parameters contribute to the derivation

of the important epidemic parameter R_0. We know that the epidemic threshold for R_0 is 1. Therefore, expressing R_0 in terms of the relevant network parameters like the key sharing probability q, the infection rate β, the infectivity duration τ, etc., would shed more light on what parameter to control in order to prevent an epidemic outbreak. The resultant expression of the basic reproductive number would also be indicative of the correctness of our epidemic model.

Theorem 4.2. *If T denotes the transmissibility and u denotes the probability that the vertex at the end of a randomly chosen edge remains uninfected during an epidemic, then the basic reproductive number R_0 is given by $R_0 = \sum_j j \sum_k \binom{k}{j} T^j (1 - T)^{k-j} \frac{(1-v^k)p(k)}{\sum_k [(1-v^k)p(k)]}$, where $v = 1 - T + Tu$.*

Proof. Let I denote the set of infected nodes in the sensor network. Moreover, let n_i denote the number of nodes that are infected by node i and d_i represent the degree of node i. We are interested in the probability $Pr[n_i = j | i \in I]$. This can be written as

$$Pr[n_i = j | i \in I] = \sum_k Pr[n_i = j | d_i = k, i \in I] Pr[d_i = k | i \in I] \tag{4.30}$$

From Equation (4.30), using Bayes' Rule, we can write

$$Pr[d_i = k | i \in I] = \frac{Pr[i \in I | d_i = k] p(k)}{Pr[i \in I]}. \tag{4.31}$$

where $p(k)$ is the degree distribution of the network. Given that β is the infection probability per unit time and τ is the average recovery time of an infected node, we have from Equation (4.24) the probability with which each link is occupied as

$$T = 1 - e^{-\beta\tau}. \tag{4.32}$$

From Equation (4.26) and (4.27), we have the size of the epidemic as S and u is simply the probability that the vertex at the end of a randomly chosen edge remains uninfected during an epidemic. Thus, the probability that a vertex does not become infected via one of its edges is

$$v = 1 - T + Tu, \tag{4.33}$$

where $1 - T$ is the probability that the edge is unoccupied, and Tu denotes that probability that it is occupied but connects to an uninfected vertex. Consequently, the total probability of being *uninfected* if a vertex has degree k is v^k. This leads to

$$Pr[i \in I | d_i = k] = 1 - v^k. \tag{4.34}$$

Hence, we have

$$Pr[i \in I] = \sum_k (1 - v^k) p(k) \tag{4.35}$$

Substituting this into Equation (4.31) gives

$$Pr[d_i = k | i \in I] = \frac{(1 - v^k) p(k)}{\sum_k (1 - v^k) p(k)}. \tag{4.36}$$

Furthermore, we have

$$Pr[n_i = j | d_i = k, i \in I] = \binom{k}{j} T^j (1 - T)^{k-j} \tag{4.37}$$

and substituting this into Equation (4.30) gives

$$Pr[n_i = j | i \in I] \;=\; \sum_k \binom{k}{j} T^j (1 - T)^{k-j} \frac{(1 - v^k) p(k)}{\sum_k [(1 - v^k) p(k)]}. \tag{4.38}$$

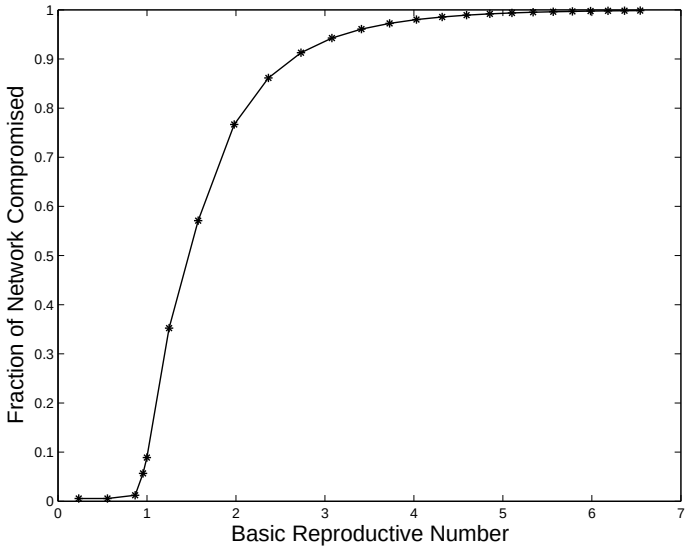


Figure 4.6. Fraction of network infected vs. basic reproductive number R_0 .

Finally, the basic reproductive number R_0 is given by

$$R_0 = \sum_j jPr[n_i = j | i \in I]. \tag{4.39}$$

□

Equation (4.39) gives an expression where R_0 is expressed in terms of the transmissibility T which is dependent on two factors, namely, the infection rate β and the infective duration τ. Fig. 5 illustrates the epidemic size based on the basic reproductive number R_0. As expected, we find that in our random graph based sensor network model, the transition point of R_0 is 1, above which an epidemic outbreak occurs. This result further proves the correctness of the model for capturing the epidemic propagation of a malware infection in a securely communicating sensor network.

4.4 Simulation Study

We employ a discrete-event dynamic system and therefore event-driven simulation to accurately simulate the propagation of the infection spreading process. We

have used JProwler [89], a probabilistic, event-driven wireless network simulator in Java, for our experiments. JProwler, which is a java implementation of the Prowler simulator is capable of simulating the non-deterministic nature of the communication channel and the low-level communication protocol of wireless sensor nodes. A probabilistic radio channel model is used with the received signal strength function defined as

$$P_{rx} = \frac{P_{tx}}{1 + d^{\gamma}} \cdot [1 + \alpha(d)] \cdot [1 + \beta(t)] \tag{4.40}$$

where P_{tx} is the transmit power and d is the distance between two nodes. The parameters α and β are normal random variables and model the probabilistic nature of the radio channel. A packet error rate p_{error} simulates the effect of any unmodeled effects on the transmission probability. In this section, we outline our discrete-event driven simulation model setup for the gradual progress of the spread of node compromise. We then use this model to capture the time dynamics of the spread of the compromise which we have largely omitted in our random graph based epidemic model. In our random graph analytical model, we obtained the static values of the maximum fraction of the network that was compromised. Through our simulation, we aim to capture the dynamics of the infection spread. This way, we not only concern the final stable state results but also investigate the temporal effects of node recovery on the extent of infection spread.

4.4.1 Simulation Setup

In our simulation, we assume the number of sensor nodes in the network to be 10000. The sensor network is produced by distributing the sensors in a 600×600 $unit^2$ area. The communication range of each node is assumed to be 25 units. The mean data rate of the wireless links is set to 40 Kbps with the packet length set to

48 bytes. The mean packet transmission time is calculated accordingly. The MAC layer is a simple CSMA based scheme modeling the Berkeley motes' MAC layer. We have used the default settings of JProwler for the standard deviations of α and β as 0.45 and 0.02, respectively and the packet loss rate is set to 0.1.

For the uniform random deployment, the location of each node is selected from a uniform distribution within the region. For the group based gaussian deployment scenario, the deployment points are arranged in a 10×10 grid in the monitored area and there are 100 nodes in each group. For each group, the location of a node is selected from a two-dimensional gaussian distribution with the deployment point as the mean and standard deviation $\sigma_d = 10$.

We employ the random key pre-distribution scheme described in [28] to establish the pairwise shared keys among sensor nodes. For each pair of neighbors, a small subset of keys are chosen randomly from a large pool such that they share at least one key with probability q.

Our simulation works in two phases. In the first phase, we form the network where each node identifies its set of neighbors and entries are made into a neighbor table. Based on typical communication distances between nodes and their respective locations, we derive the set of neighbors for each node. The degree of the key sharing network is controlled by changing the value of the key sharing probability q between neighbors. It is in the first phase that the key sharing topology for the epidemic propagation is derived based on the value of q between each pair of neighboring nodes and from the manner in which the nodes are deployed.

In the second phase, we simulate actual virus propagation. Initially, at $t = 0$, the number of infected nodes, denoted by $I(0)$ is set to be 1. At any time point t, the population is divided into the group of susceptible nodes, $S(t)$, and the group of infected nodes, $I(t)$. In the situation where we have nodes that are immunized and thus recovered, we denote this set of recovered nodes by $R(t)$. The timeline of these

sub-populations is obtained by observing the population counts after fixed simulation intervals of 1 time unit. The average incubation time at an infected node is assumed to follow a poisson distribution with a considerably low mean of 1 unit. This denotes the time period for a node to evolve from an infected state to an infective state. The low average value essentially dictates that an infected node at simulation time t will be ready to infect its neighbors at time $(t+1)$ with a high probability. Furthermore, when this incubation period is over, we assume that the time it takes for the infected node to infect its susceptible neighbor is negative exponentially distributed with a mean of 1 unit time.

There are two simulation scenarios corresponding to our analysis.

4.4.1.1 No Recovery

First we perform the simulation for the case where nodes once compromised are not recovered. Here, the simulation is based primarily on one event - the *Infection Event* which is an infected packet transmitted from an infected node to its susceptible neighbor. Associated with each Infection event packet are the node ID of the source and the ID of the destination node that has been infected by this source. The seed for the simulation is a single Infection event with a randomly selected node ID from among its neighbors as the infected destination and the source as null. The simulation is started by inserting this event into the event priority queue with the prioritization based on the event times. When an *Infection Event* is popped from the queue, the list of its neighbors are looked up in the neighbor table. From its susceptible neighbors, a node is selected randomly for infection, according to the infection probability β and an Infected packet is transmitted to it. A packet transmission event takes into consideration the physical characteristics of the network like transmission time and packet collision rate. Upon being infected, a node generates a new infection event.

4.4.1.2 With Recovery

In the case where infected nodes are recovered, we define a new *Recovery Event* for our simulation model. Our aim is to keep the average recovery time constant and study the time dynamics of the nodes in different groups based on different topology structure and infection probabilities. We assume the recovery time for an infected node to be negative exponentially distributed with a mean of τ_0 units. The CDF of the recovery time is represented by

$$Pr[t < T] = 1 - e^{-\tau_0 T}. \tag{4.41}$$

When an *Infection Event* is popped from the queue, we obtain the difference of the current simulation time and the time when the event was inserted into the queue. Using this time difference in Equation (4.41), the probability of inserting a *Recovery Event* or an *Infection Event* is calculated. However, when a *Recovery Event* is triggered, no event is further inserted. The corresponding node is marked as recovered and remains immune to further infections.

4.4.2 Simulation Results and Discussion

4.4.2.1 Simulation Results for No Recovery Case

The simulation results for the case without recovery are shown in Figs. 4.7 and 4.8. Fig. 4.7 represents the case for uniform random deployment while Fig. 4.8 shows the case for the group based deployment. The figures illustrate the compromise dynamics under moderate and high infectivity β. We vary the key sharing probability q between neighbors in each plot in order to simulate the variance of the degree distribution of the key sharing topology under the two deployment scenarios. As expected, the uniform random deployment curves are smoother than the group based

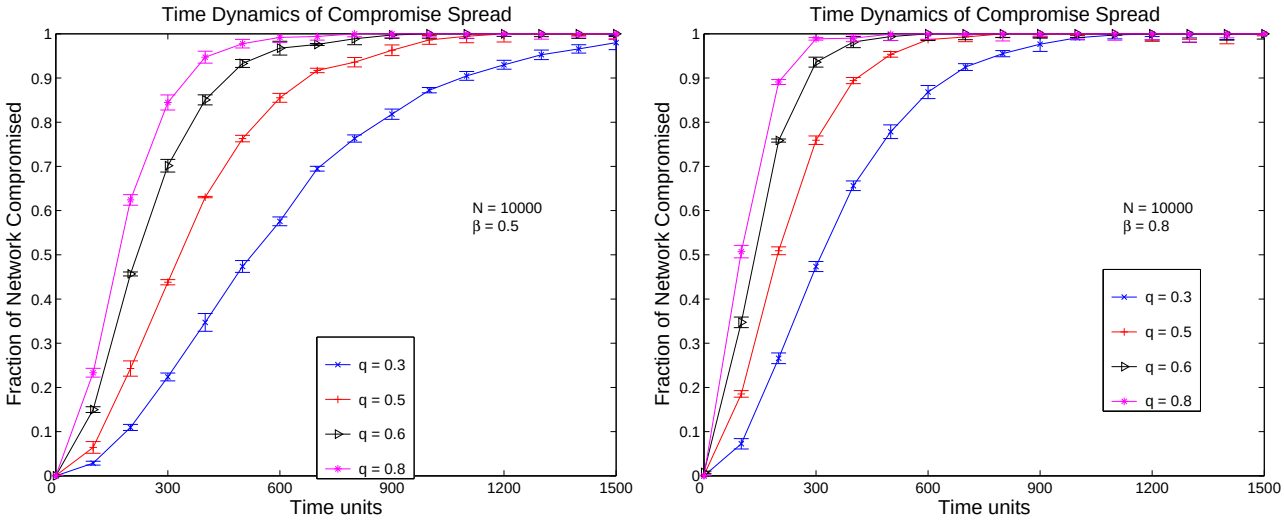


Figure 4.7. Dynamics of the infective population (Uniform Random Deployment Without Node Recovery) : (a) Uniform Random Deployment and Moderate Infectivity, (b) Uniform Random Deployment and High Infectivity.

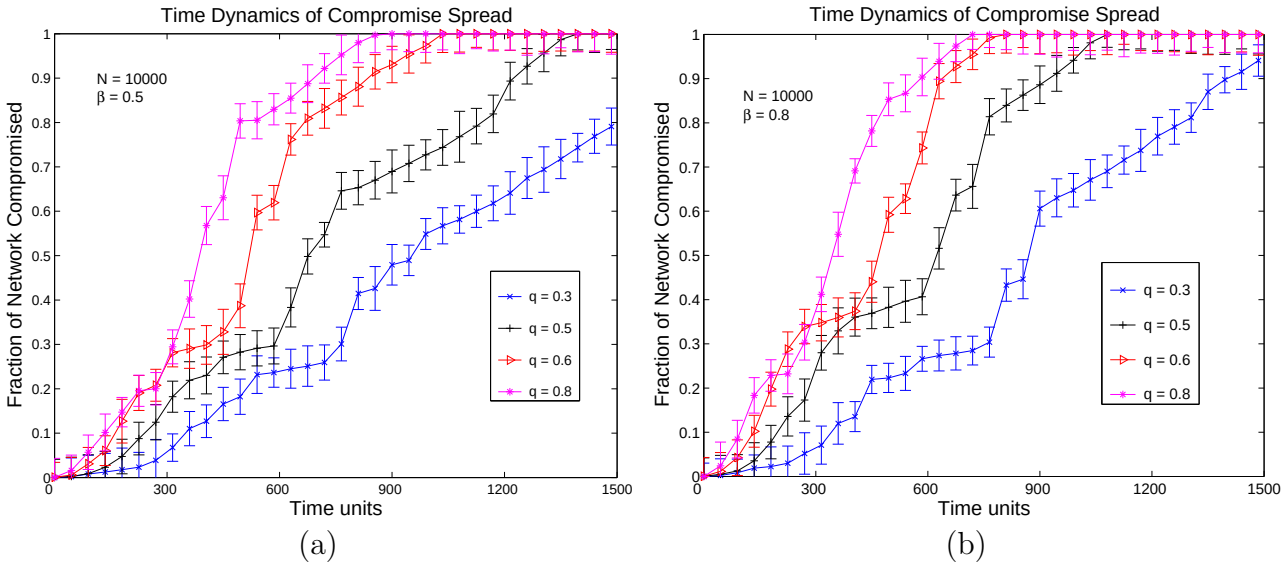
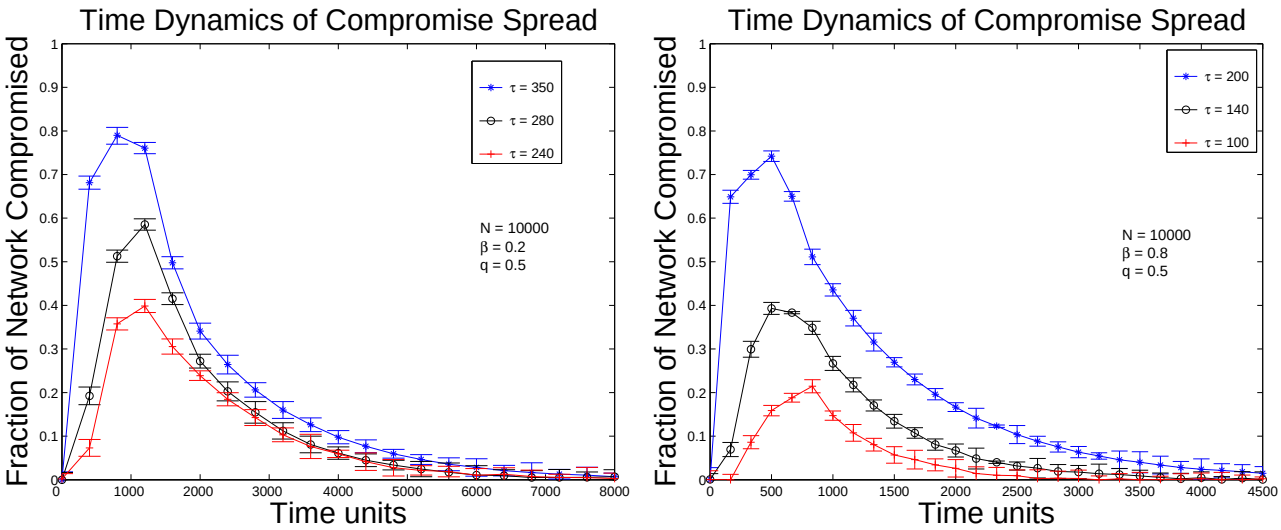


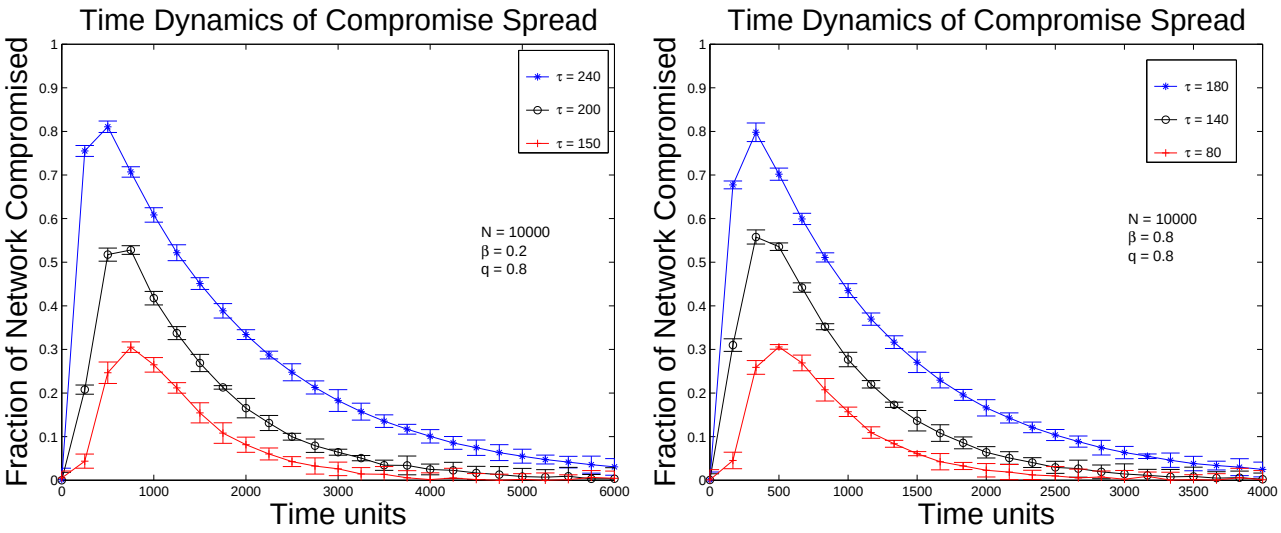
Figure 4.8. Dynamics of the infective population (Group Based Deployment Without Node Recovery) : (a) Group based deployment and moderate infectivity, (b) Group based deployment and high infectivity.



(a) (b)

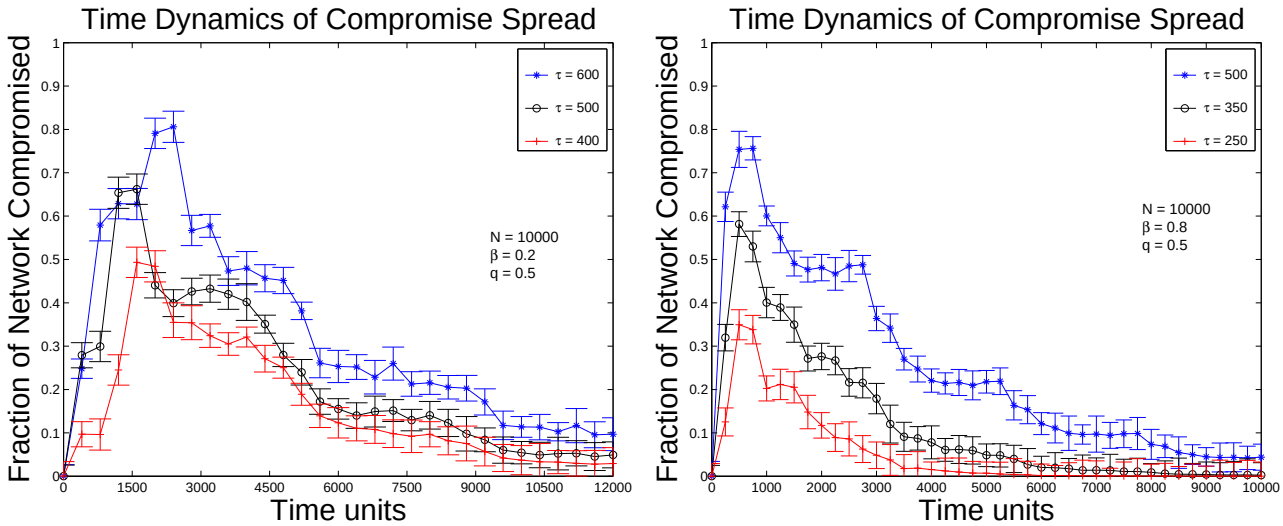
Figure 4.9. Dynamics of the infective population for Uniform Random Deployment (With Node Recovery and $q = 0.5$) : (a) Uniform Random Deployment and Low Infectivity, (b) Uniform Random Deployment and High Infectivity.

deployment. In the latter, the slope is sharply affected by the density of nodes in the region of the propagation spread. This varies regularly and we observe that this variation of node density actually slows down the propagation dynamics. For instance, comparing Fig. 4.8(b) with Fig. 4.7(b), we observe that for $q = 0.5$, the network is compromised around time $t = 1400$ for the group deployment case, whereas for uniform deployment, the network is compromised around time $t = 900$ for the same value of q. A similar observation is made when comparing Figs. 4.7(a) and 4.8(a) for a lower infectivity value of $\beta = 0.5$. The reason for the difference in time taken for the infection to spread to the entire network is attributed to the difference of the expected node degree in the different deployment cases. A group based deployment with each group deployed in a two dimensional gaussian manner results in a lowering of the expected node degree of the network at the physical level.



(a) (b)

Figure 4.10. Dynamics of the infective population for Uniform Random Deployment (With Node Recovery and $q = 0.8$) : (a) Uniform Random Deployment and Low Infectivity, (b) Uniform Random Deployment and High Infectivity.



(a) (b)

Figure 4.11. Dynamics of the infective population for Group based deployment (With Node Recovery and $q = 0.5$) : (a) Group Deployment and Low Infectivity, (b) Group Deployment and High Infectivity.

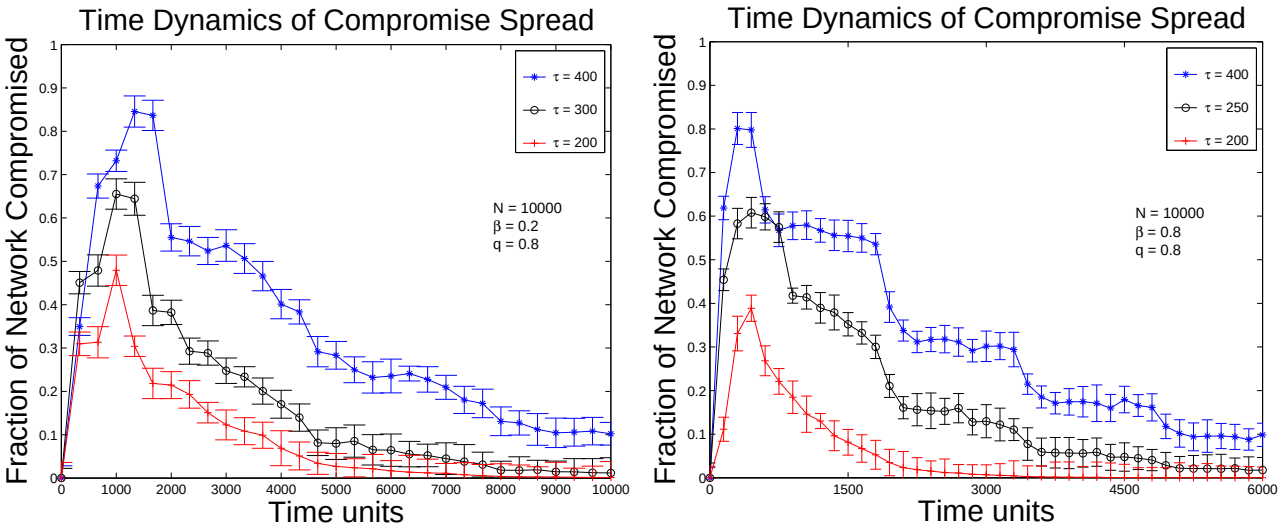


Figure 4.12. Dynamics of the infective population for Group based deployment (With Node Recovery and $q = 0.8$) : (a) Group Deployment and Low Infectivity, (b) Group Deployment and High Infectivity.

4.4.2.2 Simulation Results for Recovery Case

Figs. 4.9, 4.10, 4.11, and 4.12 show the simulation dynamics in the presence of a recovery strategy. Figs. 4.9 and 4.10 depict the epidemic propagation under a uniform random deployment of sensors, while Figs. 4.11 and 4.12 are depictions of a group based deployment with the sensors in each group distributed in a two-dimensional gaussian manner. For both types of deployment, we investigate two scenarios, viz., one in which the infectivity of the compromise process is low and the second where it is quite high. For each of the cases of infectivity, we also observe the dynamics under different key sharing probabilities q. For a comparatively sparser key sharing network ($q = 0.5$), as depicted in Figs. 4.9(a) and (b), we observe that a higher infectivity duration is required to achieve similar levels of epidemic propagation in the network as in Fig. 4.10. Comparing Figs. 4.9 (a) and 4.10 (a), we observe that increasing the node connectivity by increasing q from 0.5 to 0.8, raises the infected population peak count from 40% to 80% for $\tau = 240$. Trivially, we

also observe that the peak count is achieved at a much faster rate with an increased key sharing probability from $q = 0.5$ to $q = 0.8$.

The plots in Figs. 4.11 and 4.12 depict the same dynamics of epidemic propagation, but here the nodes are deployed in groups which are gaussian distributed about the deployment point. In Fig 4.11, each group is composed of 10 nodes, whereas in Fig 4.12, there are 50 nodes in each group. As in the case for no-recovery, we observe that the temporal behavior of the infective population reflects the variability of the node density at different regions of the network. As a result, the slope of the infective curve changes depending on how close to the mean position the propagation front of the compromise process is. In other words, we observe that when the curve is rising, the slope decreases when the compromise process wavefront reaches a sparse section of the network, i.e., an area farther away from any deployment points. On the other hand, when the curve is decreasing, it falls steeply when the wavefront reaches a sparse region because nodes are not infected as fast as they are recovering resulting in a net decrease in infected population. This is the reason we also observe an increase in the total infective percentage at certain phases of the process when the region becomes very dense. In such a situation, the infection process suddenly accelerates resulting in a net increase in the infective population percentage. Figs. 4.11 (a) and (b) depict the process in a network of 10000 nodes under two levels of infectivity β when the key sharing probability is 0.5, while Figs. 4.12 (a) and (b) depict the same for a higher key sharing probability of $q = 0.8$. Apart from the periodic increase and decrease in the propagation process for the gaussian distribution, we also observe an important comparative result between the two types of deployment.

Comparing Figs. 4.9 and 4.10 with Figs. 4.11 and 4.12, we notice that similar peak levels of infectivity are obtained for the group deployment at higher levels of the infectivity duration τ. In other words, the nodes have to remain infective for a longer duration in the group based deployment scenario than when uniformly deployed in

order to infect the same fraction of the population. This is in tandem with what we observed in our previous analytical derivations and simulations where epidemic proportions are reached for larger values of the infectivity duration τ, than in the situation when nodes are uniformly deployed. As mentioned earlier, the reason for which we observe this difference is that the average node degree of the network is reduced when the nodes are deployed in groups. The propagation progresses much more smoothly when the connectivity is uniform in nature than when it varies every now and then. In our simulations, we observed that the compromise process is more adversely affected by regions of low connectivity than helped by regions of higher connectivity. If the process dies out in a region of low density before it could reach a region of higher density, then the nodes in these sparse areas become regions of failure for the epidemic process.

4.4.3 Correlation between Analytical and Simulation Results

Although our analysis and simulation provide separate viewpoints of the epidemic process, they are correlated and a few observations connecting them reinforce the results. As mentioned earlier, in our random graph analysis, we obtained a final picture of the resulting epidemic fallout. However, our simulations captured the temporal dynamics of how the ultimate results were obtained and how the process evolved.

In this subsection, we observe some of the correlative aspects of our analytical and simulation results by focusing on the points of the analytical curves that are represented in our simulation results. For instance, we closely observe the peak values of the curves in the simulation results for the recovery case of both the uniform and group based deployment scenarios. These points represent the maximum fraction of the network compromised before the recovery process caused the network to recover. These points are represented in the analytical plots because the points of the curves

represent the ultimate fraction of the network compromised given the values of the parameters β and q. We find that the values are close to each other. For instance, if we compare Fig. 11(b) with Fig. 5(d), we observe that the peak values of the simulation curves in Fig. 11(b), are very close to the points in the analytical curve for $q = 0.5$ in Fig. 5(d), with corresponding τ values. Similarly, for the analytical curve with $q = 0.8$ in Fig. 5(d), we observe that the peak infective values match closely with the simulation curves in Fig 12(b). Close correlation is also observed for other pairs of simulation and analytical curves.

From our analysis and simulations, we therefore remark that both the analytical and experimental results have significant implication for security scheme design in terms of revoking/immunizing compromised nodes in wireless sensor networks. While the simulation results dictate the speed at which the network must react in order to contain/prevent the effect of network wide epidemic, the analytical plots indicate what values of the key sharing probability should be, in a securely communicating network using private keys, in order to contain an infection spread below the epidemic threshold while still maintain connectivity to promote network-wide communications.

4.5 Summary

In this chapter, we investigate the potential threat for compromise propagation in wireless sensor networks. Based on the principles of epidemic theory, we model the process of compromise spreading from a single compromised node to the whole network. In particular, we focus on the effects of the key factors of the network determining a potential epidemic outbreak where the whole network will be affected. Due to the unique distance and key sharing constrained communication pattern, we resort to a random graph model which is precisely generated according to the parameters of the real sensor network and perform the study on the graph. We also ensure

that the key sharing network generated is connected before performing our epidemic propagation analyses. We, further, introduce the effect of node recovery after compromise and adapt our model to accommodate this effect. Moreover, we perform a comparative study of the effect of two deployment strategies on the outcome of the epidemic propagation. Our results indicate that a uniform random deployment is more vulnerable to epidemic propagation than a group based deployment model and reveal key parameters of the network in defending and containing potential epidemics. In particular, with node recovery, the result provides benchmark time period for the network to recover a node in order to defend against the epidemic spreading and also critical values of the key sharing probability which characterize the transition from a non-epidemic to an epidemic state of the network compromise. Our extensive simulation results validate our analytical results and more importantly, provide insights into the dynamics of the system in terms of temporal evolution of the infection process.

CHAPTER 5

MODELING MALWARE PROPAGATION OVER DISSEMINATION PROTOCOLS

In chapter 3, we modeled the process of data propagation over the different broadcast based data dissemination protocols in sensor networks and analyzed the performances of each in terms of rate of information spread. In this chapter, we highlight a corresponding negative aspect of the multihop data propagation over these broadcast protocols.

Despite their critical usefulness in performing various network-wide tasks in sensor networks, these protocols are also vulnerable if they are used by some malware to to spread across the network. A compromised node could launch an attack on the entire network by using these protocols to transport itself to all the nodes.

In the previous chapter we looked into a general scenario of node compromise spreading across the whole network under some basic mode of secure communication. The adversary uses the retrieved information from a few nodes that are captured and uses that information to spread malicious code my misusing the mutual trust shared between neighbors. We had observed the effects of different kinds of deployment strategies on the outcome of node compromise propagation.

In this chapter, we continue our investigation on the spread of node compromise by focussing on a scenario where a multihop broadcast protocol might be used as a transport vehicle to propagate malicious code across the network [19, 20]. We focus on the dynamics of the propagation process and how each of the different broadcast protocols are vulnerable to the spread of malware over themselves. In other words, we look into the rate of malware propagation and the extent of spread in the network.

The analysis in 3 was a general performance analysis of the rate of information propagation over each protocol. A similar analysis could also throw light on the vulnerability of the protocols in terms of the speed of malware spread over each protocol. In this chapter, we look into the dynamics of a malware spread under the effects of a simultaneous recovery mechanism. Thus, each node can be recovered from a possible infection within a certain mean recovery period. Our objective is to study the dynamics of a malware compromising the network and the how a simultaneous recovery process can prevent that. One of the most useful outcomes of this study would be to identify the maximum fraction of the network thatwas compromised before the network could recover itself.

This chapter is organized as follows : In section 5.1, we explain the system model. In section 5.2, we discuss the attack model. We perform our model analysis in section 5.3. Section 5.4 provides an analytical discussion of the model. We perform our simulation study in section 5.5 and summarize in section 5.6.

5.1 System Model

The system model is the same as used in chapter 3. The population in our model is the total number of nodes, N, in the sensor network which are assumed to be stationary and uniformly randomly distributed with the node density denoted by σ. The number of infected nodes $I(t)$ at time t are those that have been compromised by a malware spreading over the broadcast protocol. We have $S(t)$ denoting the number of susceptible nodes in the network. Moreover, we have $R(t)$ denoting the set of recovered nodes at time t.

The rate of malware infection, β, represents the probabilistic rate at which an infective node communicates with a susceptible one through a broadcast protocol, thus compromising the latter. Here β depends on two factors: (i) probability ρ

representing the infectivity of the malware which is a measure of how contagious it is, and (ii) the rate of communication of the protocol. The degree of susceptibility of a node depends on its average degree η, the rate of communication between nodes, and the probability ρ.

We use ρ as a parameterization of the different infectivity characteristics of different malwares. In other words, we use this parameter to differentiate the threat potentials of different malwares which is independent of the broadcast protocol. This parameter, which is the probability of infection, is used as a weighting factor on the communication rate to generate the infection rate β. The justification is that malwares may use different schemes to use the broadcast protocols for compromising the nodes. Moreover, depending on the scheme, the infection can be potent enough to spread further, i.e., compromise the recipient susceptible node and further go on to infect other susceptible nodes. The infection rate, β, is, thus, obtained by combining the communication rate with ρ. The rate of removal, γ, represents the rate at which nodes recover and lose their infectivity in the network. This recovery procedure, for instance, is effected either by injection of an antivirus to disinfect the virus infected nodes or revoking the compromised nodes.

5.2 Attack Model

In this subsection, we briefly present the possible attacks on the broadcast protocols. A source node uses a broadcast protocol, e.g., Deluge, MNP, etc, to disseminate a piece of information to the rest of the network. Firstly, under the absence of any kind of authentication scheme, the compromise of any node can be a threat to the entire network. In other words, if a compromised node running a malicious software broadcasts a metadata advertising a higher version, other nodes would start to download it. Secondly, in the presence of authentication techniques used by the

source of the broadcast, we observe, that it may not always be the base-station that is disseminating useful information to the entire network. In other words, we assume that a regular sensor node can also be the source of such a broadcast where these protocols are employed. Possible authentication schemes include digital signatures and hash chain based mechanisms [24], [27]. An attacker, who has compromised a source node and retrieved the keys pertaining to the security functions of the protocol, can then implant a piece of malware and use the protocol to transfer it to the rest of the nodes. We characterize a malware [90] as being a piece of malicious software which could include computer viruses, worms, trojan horses, spywares, etc. Since, the malware is now signed by the captured keys at the source, it would pass authentication verification at the recipient nodes. The working mechanism of the broadcast protocol allows an infected node to successfully pass on the malware in its neighborhood and ultimately spread it to the whole network in a circular multihop rippled propagation.

Table 5.1. Malware Propagation Model Parameters

Model Parameter	Description
N	Total number of nodes
η	Average node degree
σ	Node density
R_c	Communication radius of node
S(t)	Susceptible nodes at time t
I(t)	Infective nodes at time t
R(t)	Recovered nodes at time t
β	Malware infection rate
γ	Malware removal rate
ρ	Malware infectivity potential

5.3 Model Analysis

In this section, we construct the formulation for the spread model of a malware in the presence of a simultaneous recovery mechanism. We assume that the network has the capability to recover some of the infected nodes that have been compromised. Once a node gets compromised, there is a finite probability and duration within which it can be recovered and lose its infectivity. This recovery mechanism could be effected by injecting a disinfecting software into the network. Moreover, we assume that the it is only possible to recover an infected node. In other words, it does not incorporate immunization of susceptible nodes prior to infection.

Without any loss of generalization, let τ denote the expected duration that a node stays infected. The expected recovery rate is thus given by $\gamma = \frac{1}{\tau}$. Moreover, from the attacker model's perspective, we also assume that after a node has recovered, it is immune to that particular malware which caused the infection. This may not hold for other classes of malwares which might be currently active in the network. However, it is a fair assumption because we are interested in evaluating the vulnerability of the protocol from the perspective of how fast it takes for a particular malware with a given infectivity to use it to infect the whole network.

Based on the circular strip model as depicted in chapter 3, the infected nodes that lie within a circular strip of thickness R_c are able to interact with a fraction of the susceptible nodes. However, simultaneously, a fraction of the infected nodes is also being recovered. Therefore, with recovery, the mass balance equation takes the form:

$$N(t) = S(t) + I(t) + R(t) \tag{5.1}$$

Similar to Eq. (3.3), the number of infected nodes that lie in the circular strip of thickness R_c is proportional to $\sqrt{I(t) + R(t)} = \sqrt{N - S(t)}$. Moreover, based

on similar analysis for the non-recovery case, each infected node interacts with the susceptible fraction of its neighbors. Therefore, the non-linear ordinary differential equations describing the process can be defined as:

$$\frac{dI}{dt} = \beta c\sqrt{N-S}\frac{S}{N}\eta - \gamma I \tag{5.2}$$

$$\frac{dS}{dt} = -\beta c\sqrt{N-S}\frac{S}{N}\eta \tag{5.3}$$

$$\frac{dR}{dt} = \gamma I \tag{5.4}$$

We derive the exact solutions of these equations to find out the growth of infected and susceptible nodes with time. Then we solve for $S(t)$ and after putting the boundary condition i.e., at $t = 0, S(t) = N - 1$, we obtain:

$$S(t) = N - N\left(\frac{2}{1 + (\frac{\sqrt{N}-1}{\sqrt{N}+1})\, e^{-\frac{\beta c\eta}{\sqrt{N}}t}} - 1\right)^2. \tag{5.5}$$

Substituting this expression in the equation for dI/dt, and denoting the constants $C_1 = \frac{\sqrt{N}-1}{\sqrt{N}+1}$ and $C_2 = -\frac{\beta c\eta}{\sqrt{N}}$, we obtain:

$$\frac{dI}{dt} = \frac{\beta c\eta}{\sqrt{N}}\left[\left\{N - N\left(\frac{2}{1 + C_1\, e^{C_2 t}} - 1\right)^2\right\} \left(\frac{2}{1 + C_1\, e^{C_2 t}} - 1\right)\right] - \gamma I. \tag{5.6}$$

After multiplying both sides by $e^{\gamma t}$, a little simplification leads to the form:

$$\frac{d(Ie^{\gamma t})}{dt} = -C_2 N\left[\left\{1-\left(\frac{2}{1+C_1\,e^{C_2 t}}-1\right)^2\right\} \left(\frac{2}{1+C_1\,e^{C_2 t}}-1\right)\right]e^{\gamma I}. \tag{5.7}$$

We use the Gaussian Hypergeometric Function *Hy2F1* [79] [1], to solve the above equation and obtain a closed form expression for $I(t)$. As with the non-recovery case, we use the same boundary condition, i.e., at $t = 0$, $I(t) = 1$.

$$I(t) = \frac{4C_1C_2Ne^{C_2t}}{(C_2+\gamma)(C_2+C_1C_2e^{C_2t})^2}\big[A\big] + e^{-C_2t} - \frac{4C_1C_2Ne^{-\gamma t}}{(C_2+\gamma)(C_2+C_1C_2)^2}\big[B\big], \tag{5.8}$$

where

$$A = (C_2+\gamma)(-C_2+\gamma+\gamma C_1e^{C_2t}) - (\gamma+\gamma C_1e^{C_2t})^2 \; Hy2F1\left(1, 1+\gamma/C_2, 2+\gamma/C_2, -C_1e^{C_2t}\right)$$

$$B = (C_2+\gamma)(-C_2+\gamma+\gamma C_1) - (\gamma+\gamma C_1)^2 \; Hy2F1\left(1, 1+\gamma/C_2, 2+\gamma/C_2, -C_1\right).$$

Eq. 5.8 gives the closed form expression for the number of infectives at time t in the recovery case. Having proposed the epidemic model for the infection propagation in a sensor network, we now look into how each broadcast protocol fits into the

[1] This function solves the Gaussian Hypergeometric differential equation: $x(1-x)y'' + c-(a+b+1)xy' - aby = 0$

model. In the following section we derive the infection rate β of our model in terms of the communication rate of each protocol in order to effectively characterize the propagation over them.

5.4 Analysis Discussion

As detailed in chapter 3, the infection rate for each of the broadcast protocols, Trickle [53], Deluge [36], Firecracker[52] and MNP [49] are calculated based on the corresponding communication rate of the protocol. This infection rate is then incorporated in the epidemic model to finally produce the formulation for the propagation dynamics of the malware over each protocol.

Apart from the rate of transfer established by the physical characteristics of the network and the working principle of the broadcast protocol, we also need to consider the property of each malware. This property differentiates one malware from another. This is the inherent characteristic of the malware, its type and what mechanism it adopts to spread. We, thus, parametrically capture this aspect of a malware by ρ, the infectivity potential of the malware. This parameter ρ differentiates one malware from another.

Similar to the previous analytical derivation, as shown in chapter 3, the infection rate β is calculated from each protocol's communication paradigm.

Deluge and MNP are both bulk data transfer protocols and thus the rate of transfer of a page is regarded as the communication rate of the protocol. We make the assumption that a page of data is large enough to transfer a malware from one node to its neighbor. This necessarily may not be true but gives us a worst case view of the malware spread rate.

Based on this, we observe that MNP's sender selection algorithm, makes it more secure than Deluge because of the delay of MNP in servicing requests as it waits to

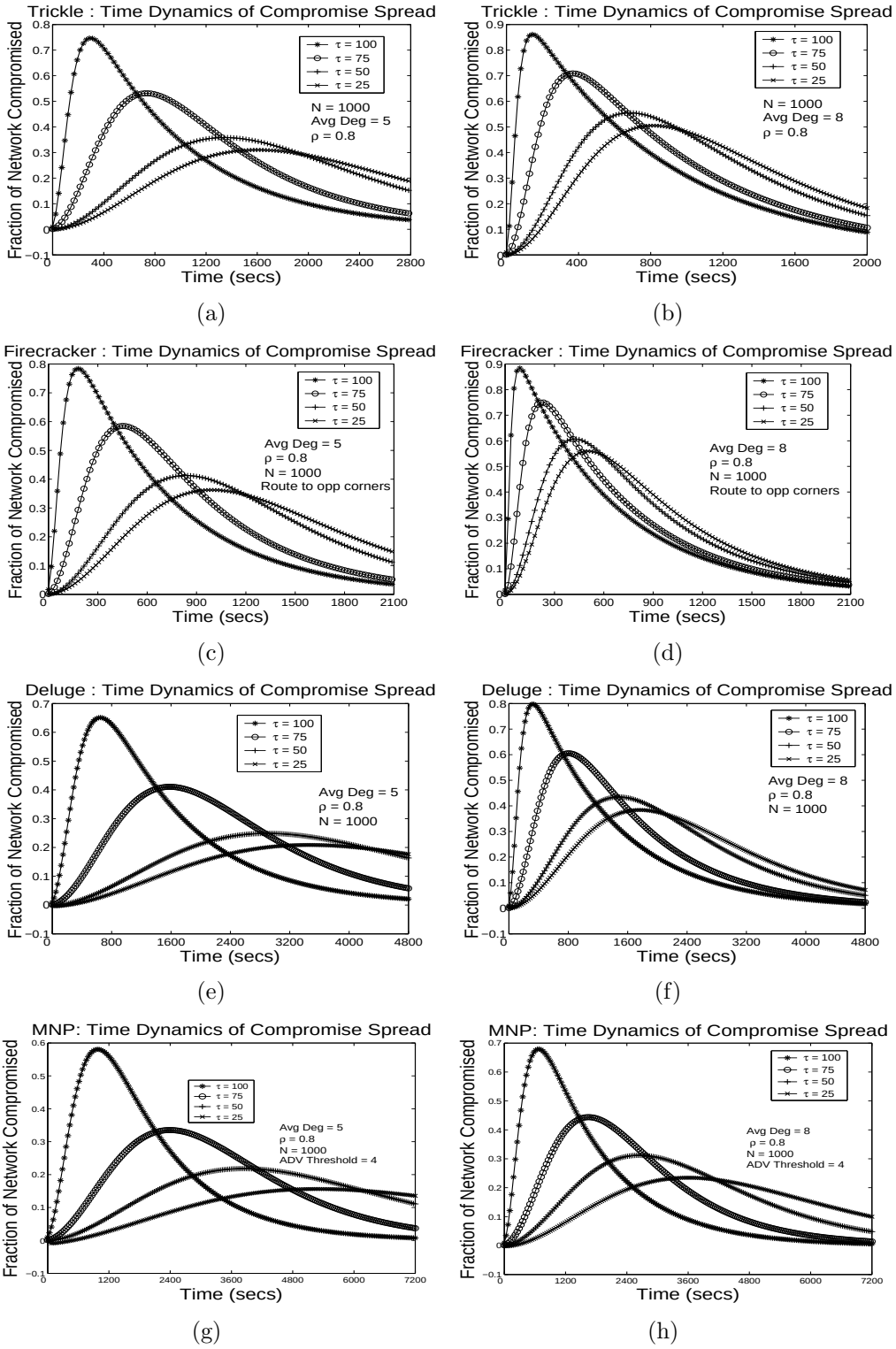


Figure 5.1. Time Dynamics of infected nodes, $I(t)$, for different values of τ (average infectivity duration) : (a) Trickle (avg degree = 5), (b) Trickle (avg degree = 8), (c) Firecracker (avg degree = 5), (d) Firecracker (avg degree = 8), (e) Deluge (avg degree = 5), (f) Deluge (avg degree = 8), (g) MNP (avg degree = 5), (h) MNP (avg degree = 8).

select a particular sender in a neighborhood. From a vulnerability standpoint, this feature slows down infectious malwares which might have spread deeper into the network faster, had the requests been serviced immediately, as in Deluge. This is valuable time for a recovery process to control the spread.

Fig. 5.1 illustrates the infection dynamics in the face of a simultaneous recovery procedure in the network. Without making any assumptions on the recovery process adopted, we have depicted the infection dynamics under different average rates for recovery. As mentioned earlier, recovery could be achieved by injecting a piece of disinfecting software, such as an antivirus, into the network. We observe the infection dynamics based on different values for average infectivity duration depicted by the parameter τ. We also assume the malware to have high infectivity of $\rho = 0.8$. From the sub-figures in Fig. 5.1, we observe that the fraction of the network that gets maximally infected is significantly lowered with a simultaneous recovery procedure. This difference is even more significant in the case of a lower value of τ which further weakens the potency of the infection. For instance, comparing figs. 5.1(f) and 3.4(f), the peak of the infective curve, with average $\tau = 100$, is lowered significantly and it is also achieved at a time close to 200 seconds in the non-recovered case, while it reaches a value close to 450 seconds with recovery. Similarly in MNP, comparison of figs. 5.1(g) and 3.4(g) shows that a simultaneous recovery procedure not only lowers the peak of the infection curve but also slows the time it is reached, considerably. For the curves with lower infectivity duration (e.g., $\tau = 25$), the difference in the peak fraction infected is even larger. Thus, the introduction of the recovery process slows down the infection considerably and in the case of Deluge and MNP, it is even more conspicuous because the general speed of the protocol is slower. As an aside, the total recovery time for Deluge takes almost twice the time taken by Firecracker.

5.5 Simulation Study

The simulation setup is similar to the one chosen for chapter 3. The time dynamics of the malware spread is captured with varying degrees of infectivity on the whole network. We have used JProwler [89], a probabilistic, event-driven wireless network simulator in Java, for our experiments.

We assume $N = 1000$ sensor nodes with uniform random deployment in the network. We have used the default radio model in JProwler where all communication links in the network are symmetric and deterministic. Packets are lost only when there is a collision. Likewise, the maximum data rate of wireless links is set to be 32 Kbps. The maximum length of a packet is also fixed at 40 bytes. The MAC protocol is based on a simple CSMA scheme like BMAC [66]. The metrics for evaluating the proposed framework is the time it takes for a malware to infect a given fraction of the network, spreading over each broadcast protocol. Each reported result is averaged over twenty simulation runs.

Similar to the earlier setup in chapter 3, our simulation works in two phases. In the first phase, we form the network where each node constructs the neighbor table. The entry for each node in the neighborhood table can indicate whether a node is susceptible, infected or recovered. The average node degree of the network is set to typical values of 5 and 8.

In the second phase, we simulate actual virus propagation over each broadcast protocol. Initially, at $t = 0$, the number of infected nodes, denoted by $I(0)$ is set to be 1. The time period T_p, of Trickle, has been assumed to be the unit and is equal to 1 second in our simulation.

We simulate under different network connectivities and different values of the malware infectivity, ρ, and subsequently, study the time dynamics of the infected population. For each susceptible neighbor of an infective node to which a data packet

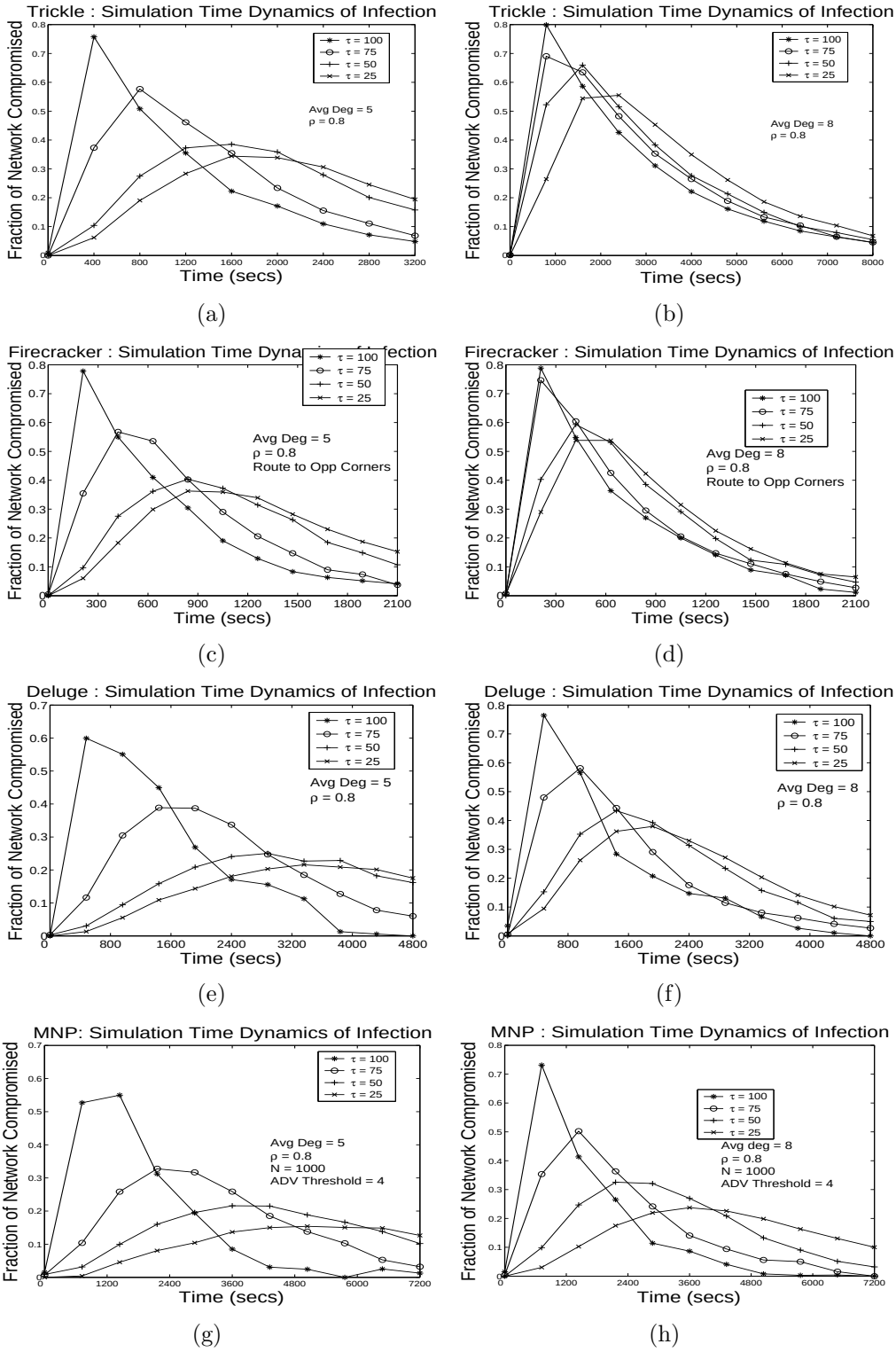


Figure 5.2. Time Dynamics of infected nodes, $I(t)$, for different values of τ (average infectivity duration) : (a) Trickle (avg degree = 5), (b) Trickle (avg degree = 8), (c) Firecracker (avg degree = 5), (d) Firecracker (avg degree = 8), (e) Deluge (avg degree = 5), (f) Deluge (avg degree = 8), (g) MNP (avg degree = 5), (h) MNP (avg degree = 8).

is to be transmitted, malware transmission is done based on the probability ρ, independently for each node.

In the case where there is no recovery, a node, once infected, stays infected for the rest of the simulation time. The simulation results for such a case would be similar to the results as shown in chapter 3 for data propagation.

However, with a simultaneous recovery mechanism, the simulation results are depicted in Fig. 5.2. With predefined infection rates derived from the broadcast protocols, we have simulated a simultaneous recovery process given that an infection spread is active. We observe that the nature of the curves closely match our analytical model.

However, similar to the simulation performed in chapter 3, we also observe some discrepancies between our simulation and analytical results as is evident from figs. 3.4 and 3.5. This is similarly attributed to the fact that the differential equation based approach approximates the process to be continuous in time which is not the case in our simulation. Moreover, the model does not incorporate border or edge effects and the infection is assumed to propagate from the center outwards. With a considerable increase in the density of the network, our simulation results would deviate significantly from the analytical results. This is attributed to the error in the packet loss probability that creeps in such scenarios. Our model would then have to be tuned accordingly so that the packet loss probability can effectively capture the impact of high density.

5.6 Summary

Broadcast protocols in sensor networks are vulnerable as potential carriers of malwares/viruses that spread over air interfaces. In this chapter, we evaluated the vulnerability of these protocols in sensor networks wherein they could be potential

carriers of malwares/viruses that spread over air interfaces. We provided a common mathematical model to analyze the process of malware propagation over different multihop broadcast protocols. Although approximately, our model successfully captures the ripple based propagation behavior of the wavefront of a broadcast protocol. Not only is the model capable of assessing the performance of each protocol in the face of a virus outbreak, but it also helps in comparing their vulnerabilities against each other. Its generic and flexible nature allows us to conveniently fit parameters of different broadcast protocols and analyze their susceptibilities. Despite the similarities in operation between some of the protocols discussed, the epidemic model successfully highlights their differences from a propagation standpoint. The model can also be extended to other complex broadcast protocols by successfully computing the infection rate β for that protocol.

CHAPTER 6

REPROGRAMMING PROTOCOL DESIGN FOR MOBILE SENSOR NETWORKS

In the previous chapters, we investigated the data propagation performance of broadcast protocols and also studied the vulnerability aspects and how a minor security breach could spread itself into the whole network, potentially exploiting the working principle of the broadcast protocols. Our entire analysis was performed on static sensor networks which are either deployed uniformly randomly onto a terrain or deployed according to another special strategy.

We now shift our focus and consider sensor networks comprised of mobile nodes. Several sensor network applications require the nodes to be mobile. Given such a mobile scenario, we perform our analysis and characterization of a broadcast based dissemination protocol.

In this chapter, we study the design of a broadcast based data dissemination protocol specifically suited to mobile sensor networks. Although the mobility factor brings considerable uncertainty about a node's location it also provides several advantages. For instance, it increases the extent of coverage of the whole area. It also increases the reliability of coverage of a given point in the area and more importantly, a lesser number of sensors can achieve the desired sensing performance. Thus, compared to static sensors, a lesser number of mobile sensors could be sufficient for a particular application. Consequently, reprogramming protocols for these mobile sensors are necessary keeping such applications in mind.

However, current reprogramming protocols for sensor networks [36], [53], [49] are unsuitable when applied in a mobile scenario. The random and continuous mobility

of nodes, renders the feature of acquiring pages in-order of existing reprogramming protocols, inefficient.

In this chapter, we present *ReMo*, a reprogramming protocol specifically designed for mobile sensor networks.

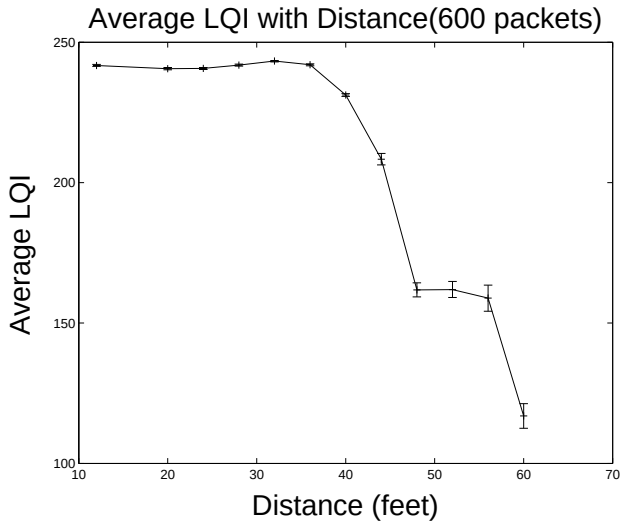
The chapter is organized as follows : In section 6.1, we discuss about link quality and signal strength metrics. In section 6.2, we depict the behavior of the Deluge protocol in a mobile environment. We present ReMo in section 6.3. In section 6.4, we perfrom our simulation based performance evaluation of ReMo and summarize the chapter in section 6.5.

6.1 Link Quality and Relative Distance Estimate

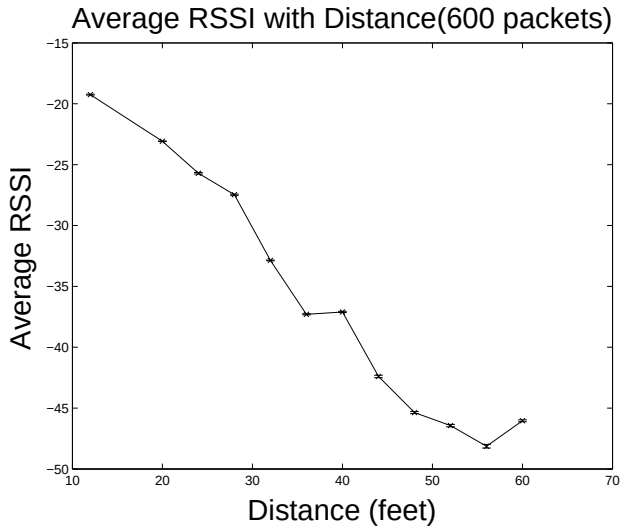
Since the mobile nodes are not assumed to know their own location, it is important to determine not only the relative distance between two neighbors but also the link quality between them in terms of the packet reception rate. These two parameters would help a node to select the best neighbor for code download. We have performed some outdoor experiments to characterize the link between two sensor nodes with 802.15.4 radios at varying distances. Our goal was to find out how the RSSI and LQI values of a radio packet varied with distance and correlated with the packet reception rate over the corresponding link.

For our experiments on link quality measurements, we used the *Sun Small Programmable Object Technology* (SunSPOT) nodes [86]. A SunSPOT has a 180 Mhz 32 bit ARM920T processor with 512K RAM and 4M Flash. The radio chip is the CC2420 [87].

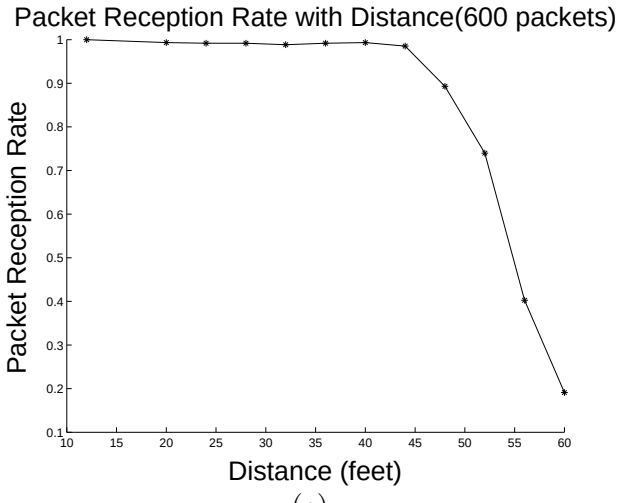
As part of the ReMo protocol, each node would continuously snoop on all radio packets transmitted in the neighborhood in order to construct important statistics like relative distance and link quality with its neighbors. However, snooping is not



(a)



(b)



(c)

Figure 6.1. Outdoor Measurements of LQI, RSSI and Packet Reception Rate with Distance : (a) LQI with Distance, (b) RSSI with Distance, (c) Packet Reception Rate with Distance.

cheap since nodes have to listen for packets that are not necessarily addressed to itself. Several low-power listening techniques exist[71] that would allow nodes to snoop at a much lower cost. However, much of the traffic in the case of ReMo is broadcast and thus nodes would not need to do any extra snooping in order to construct the link characteristics with their neighbors.

As shown in Fig. 6.1, we measured the average RSSI and LQI values of 600 packets for a distance of 12 feet to 60 feet at steps of 4 ft in an open area. The nodes were kept at a height of 3 ft above the ground. Our observations indicate that RSSI is a good indicator of relative distance between two nodes but it is not correlated with the packet reception rate as accurately as the average LQI values. Thus, ReMo uses the average LQI values as an indicator of link quality and the RSSI measurements as indicators of relative distance.

6.2 Deluge in a Mobile Sensor Network

In order to demonstrate the requirement of a suitable reprogramming protocol for mobile sensor networks, we evaluated the performance of Deluge [36], a popular reprogramming protocol, in a network with mobile nodes. Most other existing protocols [49, 52, 53] follow a similar paradigm for code propagation. We have performed simulation experiments under varying degrees of node mobility. Our primary goal was to observe the completion time of the code update for the entire network under varying average speeds for the mobile nodes. Our results indicate that Deluge shows a significant increase in network programming time as the average speeds of the mobile nodes are increased. One of the primary reasons behind this fall in performance is that Deluge distributes pages strictly in order. This strict page-ordered download makes sense in a static scenario where the new code propagates from a specific source in a wave across the network. Thus, there is a very low probability of pages arriving

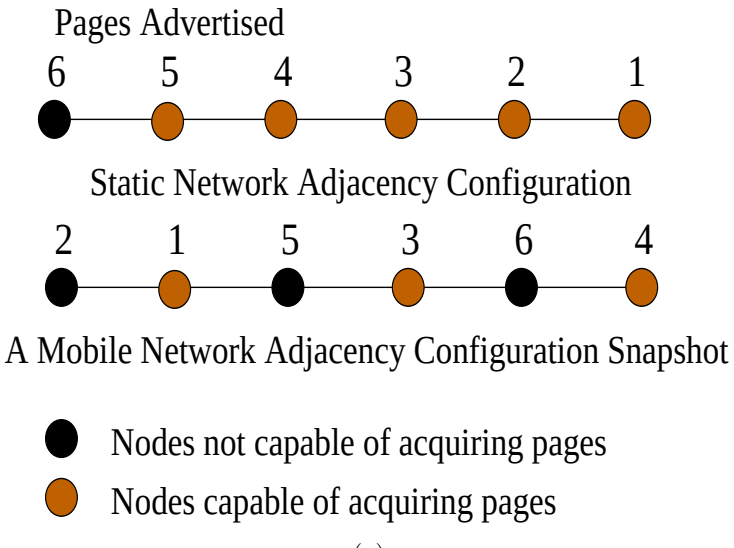
at a given node out of order. However, in a mobile scenario, this constraint no longer holds and a node might have neighbors potentially capable of providing arbitrary pages for download.

Moreover, since Deluge has a simplistic rate control mechanism for its meta-data advertisements, it is unsuitable to the dynamic changes in neighborhood density brought about by the mobility of the nodes. Furthermore, it is very susceptible to the hidden terminal problem when the neighborhood density increases.

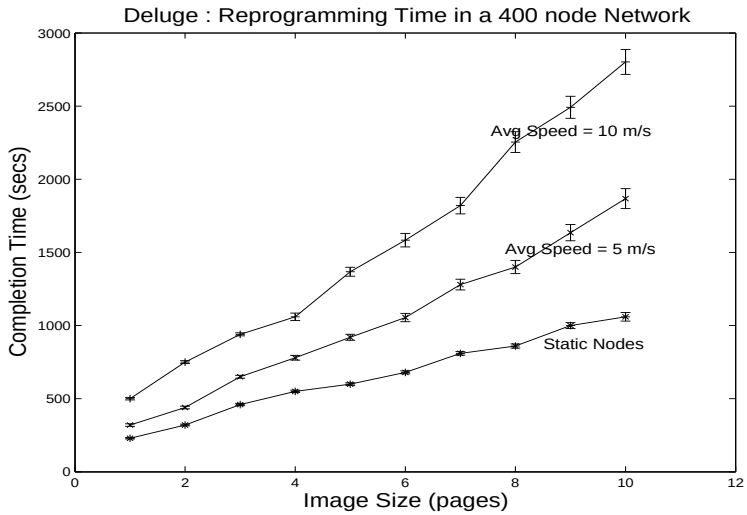
Fig 6.2 (a) is a snapshot of a network running Deluge with adjacent nodes connected by an edge. The first set of nodes shows the normal operation of Deluge in a static network where the code update propagates in a particular direction. Each node is marked with the page number it is advertising. Thus all the nodes having a page number of 5 or lower can acquire missing pages from their neighbor nearer to the source. The next array of nodes shows a mobile scenario where nodes have moved into a new adjacency configuration. Thus, the number of nodes potentially able to download new pages are less. This example proves that the regular paradigm of code propagation in an increasing order of pages in unsuitable when the nodes are mobile. Fig. 6.2 (b) shows the effect of mobility on the completion time taken by Deluge with increasing image size in a 400 node network in a square of area $4000m^2$. Each page is of size 1 KB.

6.3 The *ReMo* Protocol

Mobility among nodes poses new challenges to the efficient design of a reprogramming protocol for sensor networks. Sensors, with their stringent resource constraints have more responsibility in not only tackling the uncertainty due to mobility, but also minimizing the resource utilization in overcoming that overhead and propagate the code throughout the network as fast as possible. In this section, we first



(a)



(b)

Figure 6.2. Deluge in a Mobile Sensor Network : (a) Downloaded/Required Pages of mobile nodes, (b) Completion time of Deluge in a mobile sensor network.

introduce a few assumptions and design directions of ReMo before describing the working principle of the protocol.

6.3.1 Node Mobility Model

The most commonly used mobility model in ad hoc networks is the *random waypoint* (RWP) *model*[10]. In this model, a node in a movement period i, randomly

chooses a waypoint P_i, moves towards it at a velocity V_i chosen uniformly randomly, and pauses at the waypoint P_i for a random period T_i. This process is repeated for each subsequent movement period.

In our mobile sensor network, we assume that nodes move according to the RWP model. Moreover, nodes are *not* required to be aware of their locations. We do not make any assumptions on network connectivity being maintained at all times but assume that the nodes move around within the confines of the region.

6.3.2 Data Representation

Similar to the data representation of Deluge [36], ReMo divides the code image into fixed sized packets of size S_{pkt}. The protocol uses a basic unit of transfer called a *page* which is of size $N.S_{pkt}$ where N is a fixed number of packets. Breaking the code image into pages enables pipelining the transfer of the file over multiple hops across the network. A *version number* is used to distinguish between different code updates and must be monotonically increasing to maintain an order for all updates. A node needs to compare version numbers to decide on whether it requires an update.

6.3.3 Page Download Potential (PDP)

We define the page download potential (PDP) ω_j^i of node i w.r.t node j as a measure of the potential that j has in providing pages to i. Let the code image C_{img} consist of κ pages, such that $C_{img} = \{P_1, P_2, \ldots, P_\kappa\}$ where P_l is the l^{th} page. Let S_i and S_j denote the set of pages of node i and node j, respectively. Then the page download potential ω_j^i of i w.r.t j is denoted by

$$\omega_j^i = \frac{|S_j - S_i|}{|C_{img}|} \tag{6.1}$$

where $S_j - S_i$ is the set difference between the page sets of node j with node i. Thus, for each node i with σ neighbors, the page download potential vector Ω is denoted by

$$\Omega_i = \left\{\omega_1^i, \omega_2^i, \ldots, \omega_\sigma^i\right\} \tag{6.2}$$

6.3.4 Neighbor Link Profile (NLP)

Each node i snoops in its neighborhood for packets transmitted by other nodes in order to build a Neighbor Link Profile (NLP) vector Φ_i. ϕ_j^i, which is an element of Φ_i, represents an entry of the NLP of node i w.r.t. node j. It is a 2-tuple $< lq_j^i, dm_j^i >$ where lq_j^i is a normalized representation $\in [0, 1]$ of the link quality with node j, and dm_j^i is a direction of motion indicator. lq_j^i is calculated based on the average of the LQI values of packets received from node j. The average LQI values, as depicted in Fig 6.1, have been used for estimating the link quality with a given neighbor as they have a better correlation with the packet reception rate of a link. The link quality estimate is updated based on a window mean exponentially weighted moving average (WMEWMA) of the LQI values in each slot of a neighbor node, the window being the duration τ of each slot. Thus, the link quality update of node j is given by

$$lq_j^i(t+1) = \gamma lq_j^i(t) + (1-\gamma)\frac{lq_j^{avg}(t)}{lq_{max}} \tag{6.3}$$

where $lq_j^{avg}(t)$ is the average of the LQI values of the packets received from node j in the current slot. The weight factor γ decides the contribution of the previous estimate of the link quality.

On the other hand, dm_j^i is calculated based on RSSI values of multiple packets from the same node. The RSSI values of the 802.15.4 packets from the $CC2420$ [87] radio chip have been confirmed to have shown an agile linear correlation with the distance as depicted in Fig 6.1. Accordingly, consecutive values have been used

to approximately indicate whether a neighboring node is approaching or departing. Thus, dm_j^i takes values -1(departing), 0(uncertain), and $+1$(approaching). However, the measured RSSI values need to be sufficiently spaced in time in order to reflect a change in position of the mobile node. Accordingly, RSSI values are taken from packets that are at least spaced by a time duration of $\frac{\Delta}{\varphi}$. where Δ is a constant distance (say 8m) and φ is the average speed of a node.

If no packet is heard from an existing neighbor in the NLP in the last time slot, it is marked as *stale*. Moreover, a *stale* neighbor is again made *fresh* when a packet is received from it. After a maximum number of W slots with a neighbor remaining *stale*, it is evicted from both the NLP and the PDP lists.

6.3.5 Probability of Metadata Broadcast

One of the central features on which ReMo is based is an adaptive metadata advertisement scheme. This is a probabilistic technique for broadcasting metadata in the neighborhood considering local node density and the presence of new metadata. Time is divided into slots of fixed duration τ, and nodes essentially broadcast their metadata at every slot t based on their current advertisement probability θ_t.

In this section, we formulate θ_t, which is the probability of metadata broadcast by a node at each time slot t. Each node dynamically updates θ_t at each time slot based on gathered observation at the previous time slot. Not only is the computation of this transmission probability based on the presence of new code information in the neighborhood, but also on the relative density of the current neighborhood. Thus, each node tries to proactively ascertain if there is any neighbor with new code and also control the level of gossip in order to avoid collisions and packet loss.

Let N_t^d and N_t^s, respectively, denote the number of *different* and *similar* metadata advertisements heard by a node during slot t. Morever, let A_t denote the sum of all advertisement messages heard by the node in slot t. Thus, $A_t = N_t^d + N_t^s$.

Algorithm 1 depicts the update of the broadcast probability θ at each time slot. The notion is to allow a node to increase its broadcast probability whenever it senses that there is new code in the neighborhood. This increase is inversely proportional to the number of nodes advertising new code. In other words, nodes increase their probability by a smaller amount when the number of neighbors advertising new code is high and vice versa. Moreover, in order to avoid packet collisions, nodes decrease their probability of transmission if N_t^d and θ_t cross a threshold value of N_{Th} and θ_{high}, respectively.

More importantly, nodes keep track of the number of neighbors that are advertising the same metadata, N_t^s. They, accordingly, decrease their transmission probability in order to minimize redundantly broadcast metadata in the neighborhood. δ denotes a small step used to increase or decrease the probabilities.

Thus, for each slot t, nodes gather these information about their neighbors and update their advertisement probability for the next slot $t + 1$.

For the initial transmission probability θ_0, each node is assigned a value of $\theta_0 = \frac{1}{\eta}$ where $\eta = \frac{\pi R^2 N}{D}$ denotes the average number of neighbors that a node has within its transmission radius R. D denotes the area in which the nodes are deployed and N is the total number of nodes..

We note that a node reacts by decreasing or increasing θ as described above, based on corresponding advertisement counts received in the last time slot. When a node does not receive any advertisements in a slot, it assumes that it is either alone or in a very sparse location. In this case, it increases its probability of advertisement transmission in order to reach out to any other node in the vicinity. However, after trying for a threshold max_{NoADV} number of slots with no received advertisements, the node sleeps for a short duration and then wakes up to broadcast at the initial probability of θ_0.

Algorithm 1 TransmissionProb θ_{t+1} for $(t+1)^{th}$ slot

Input: $\theta_t, N_t^d, N_t^s, \delta, slots_{NoADV}, max_{NoADV}$, $maxNbrs$

Output: θ_{t+1}

Initialize $slots_{NoADV} = 0$;
At the Expiry of the $(t+1)^{th}$ **Timer**
Set $N_{t+1}^d = 0$;
Set $, N_{t+1}^s = 0$;
if $A_t > 0$ **then**
  if $(N_t^d > N_{Th}$ **and** $\theta_t > \theta_{high})$ **then**
    $\theta_{t+1} = \frac{N_t^d}{A_t}\theta_t\left(1 - \frac{N_t^d}{A_t}\delta\right) + \frac{N_t^s}{A_t}\theta_t\left(1-\delta\right)$;
  else
    $\theta_{t+1} = \frac{N_t^d}{A_t}\theta_t\left(1 + \frac{\delta}{N_t^d}\right) + \frac{N_t^s}{A_t}\theta_t\left(1-\delta\right)$;
  end if
else
  $slots_{NoADV} = slots_{NoADV} + 1$;
  if $(slots_{NoADV} < max_{NoADV})$ **then**
    $\theta_{t+1} = \theta_t\left(1+\delta\right)$;
  else
    Sleep for a short duration;
    Set wakeup broadcast probability $\theta_0 = \frac{D}{\pi R^2 N}$;
  end if
end if
if $(\theta_{t+1} > 1)$ **then**
  $\theta_{t+1} = 1$;
end if
if $(\theta_{t+1} < 0)$ **then**
  $\theta_{t+1} = 0$;
end if

As shown in Algorithm 1, we note that, in the situation where a node receives advertisement packets in the last slot, its probability update is composed of a weighted sum of two components. The first component is based on the contribution of advertisements that contain new information, N_t^d, whereas the second one is for advertisements with same metadata, N_t^s. The weights of these two components are based on the normalized counts of the corresponding types of advertisement messages. Moreover, we note that decrease of θ is done more aggressively for the component related to advertisements with same metadata. The obvious reason is that these advertisements pose as redundant transmissions in the neighborhood and could only cause packet collisions without providing extra information. θ is only increased when

the node hears new metadata and the number of neighbors are below a required threshold. However, as mentioned before, θ is increased cautiously by setting the increase factor inversely proportional to the count of new metadata advertisements so that it does not result in a gossip storm and subsequent packet collisions.

6.3.6 Protocol Description

In ReMo, the goal of a node is to periodically keep its neighbors updated on the version of its code and its location information. Whenever a new code is propagating through the network, a mobile node needs to optimally choose a suitable neighbor to download pages from, given the fact that the neighborhood is very dynamic as the nodes are constantly mobile.

Broadly, each node lies in either of three major states, viz., *Advertise (ADV)*, *Receive (RX)*, and *Transmit(TX)*. Fig 6.3.6 shows the detailed state transition diagram of the protocol.

In the ADV state, a node performs important functions like periodic advertisement of code metadata, neighborhood assessment, and optimal decision making for different actions like choosing an appropriate neighbor to download code from or modifying its advertisement transmission rate based on dynamic information about its current neighbors. In this state, a node broadcasts an advertisement message M_{data} containing some meta information in each slot t of duration τ with a given probability θ_t. It selects a random time $\in [\frac{\tau}{2}, \tau]$ for transmitting M_{data} to account for the *short listen* problem [53]. M_{data} is primarily comprised of two components : 1) Version Number, and 2) Downloadable data information which consists of a bit vector $< p_0, p_1, \ldots, p_{k-1} >$ of the k pages of the object image. We also note that the duration τ of each slot has an important role to play in the efficient working of the protocol. The value of τ is generally based on the average velocity of the nodes in the network. For instance, in a network where nodes move at very high speeds,

τ is smaller because the neighborhood states change more frequently. Whereas, in networks with relatively slower movement speeds, τ is of a longer duration. At each timer expiry, a node updates its count variables N_t^d and N_t^s and recalculates θ_{t+1} for the next slot as described in Algorithm 1.

The *Request* messages are of two types, 1) A *Half Request (HReq)* message, and 2) A *Full Request (FReq)* message. Since nodes do not know their location, and estimate their relative distance and link qualities with their neighbors from received packets, ReMo uses these two types of *Request* messages. When a node is sure that the destination neighbor is close enough for a reliable page download, it sends a *FReq* message, whereas it sends a *HReq* message when it is unsure of the reliability. In the latter case, it is upto the neighbor to respond with the requested page or ignore the *HReq* message. Each *Request* message contains a bit vector indicating the required set of pages. Since both the *Advertisement* and the *Request* packets from a neighboring node contain the page information, the PDP is updated upon hearing any of these packets from the neighbor.

The choice of a neighbor to transmit a *Request* message to, is based on which neighbor has a high page download potential as well as a sufficiently good link quality. Thus, a node i computes $p_j^i = \omega_j^i \cdot lq_j^i$ and selects a neighbor for sending a *Request* message after transiting to the RX state. However, after selection of j, node i sends a *FReq* message to j if $lq_j^i > lq_{thresh}$ and $dm_j^i \neq -1$. Otherwise, a *HReq* message is transmitted. The *Request* message is also broadcast in the neighborhood with the address of node j incorporated as one of the fields of the packet.

If a node in the ADV state finds that it has a non-empty PDP and NLP list and it needs code to download, it would transit to the RX state at the next timer expiry and send out an appropriate *Request* message targeted at the chosen neighbor and wait for the requested data to be downloaded. For each *Request* message, a node generates a random sequence number and includes it as a field in the message.

However, if it also receives a *Request* message in this interim before the timer expires, it would first service the arrived request provided the sequence number in the incoming request message is higher than its own generated sequence number.

A node in the RX state would transmit a maximum of R_{max} unserviced requests before transiting back to the ADV state.

When a node receives a *FReq* message in the ADV state, it transits to the TX state and transmits all the packets of the first page of the requested page vector. The recipient node sends a $Data_{ACK}$ message for the page sent. On receiving this acknowledgment, the sender transmits the data packets of the next page in the requested vector provided the link quality of the destination node (as per the last measured estimates) and the *RSSI* of the received $Data_{ACK}$ packet is above the required threshold. However, if no acknowledgment arrives or, the sender transits to the ADV state and starts broadcasting its metadata with probability θ_0. A recipient node transits to the *REDEEM* state to download missing packets of a page.

Upon receiving a *HReq* message, a sender node migrates to an intermediate *CHECK* state to decide whether the recipient node is suitable for a page transfer by evaluating the link quality and the *RSSI* of the received *HReq* packet. If the link quality is satisfactory and the node is approaching, then the page is transmitted.

The protocol messages with their essential fields are defined as follows:

- **Advertise Message**: (a) ImageName (b) Version Number (c) Image Size (d) Image Page Bit vector.
- **HReq Message**: (a) Requested Version (b) ImageName (c) Sequence Number (d) Destination Address (e) RSSI of Recvd Adv (f) Requested Vector of Pages.
- **FReq Message**: (a) Requested Version (b) ImageName (c) Sequence Number (d) Destination Address (e) Requested Vector of Pages.
- **Data Message**: (a) ImageName (b) Version Number (c) Page Number (d) Packet Index (e) Data Size (f) Data.

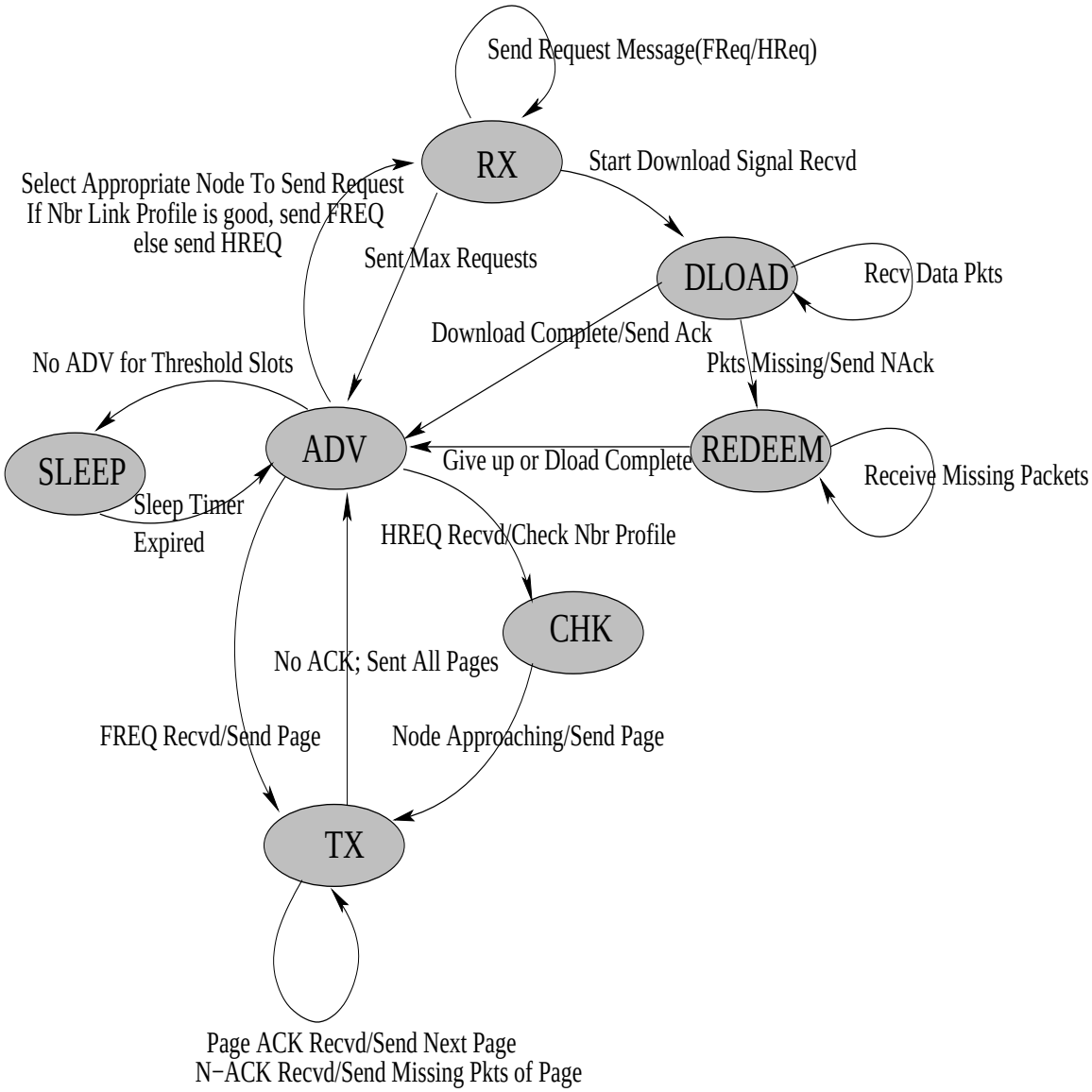


Figure 6.3. ReMo State Transition Diagram.

6.4 Performance Evaluation

We have evaluated the performance of ReMo in comparison with Deluge and MNP using the packet level network simulator GloMoSim [83]. All the three protocols have been implemented at the application layer of GloMoSim. The CSMA MAC

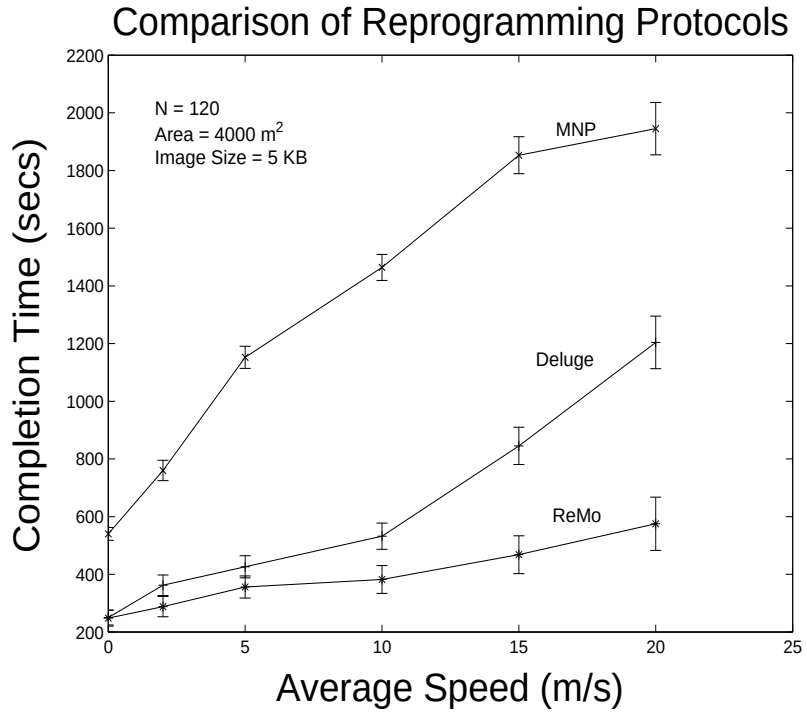
protocol model was used with a communication range of $30m$. The terrain is assumed to be a rectangular area of 4000 sq. meters. We have used the Random Waypoint mobility model [10] to evaluate the performance of the protocols with average speeds ranging from 2 to 20 m/s with a maximum pause time of 100 ms. Nodes are initially uniformly randomly deployed in the terrain. Network reprogramming time and the total number of packets transmitted to achieve that were the primary metrics of measurement for evaluating the protocols. Each simulation result was taken over 30 runs with a 95% confidence interval. Table 6.1 shows the values of the different protocol parameters for our simulation.

Table 6.1. ReMo Parameters and Value Settings

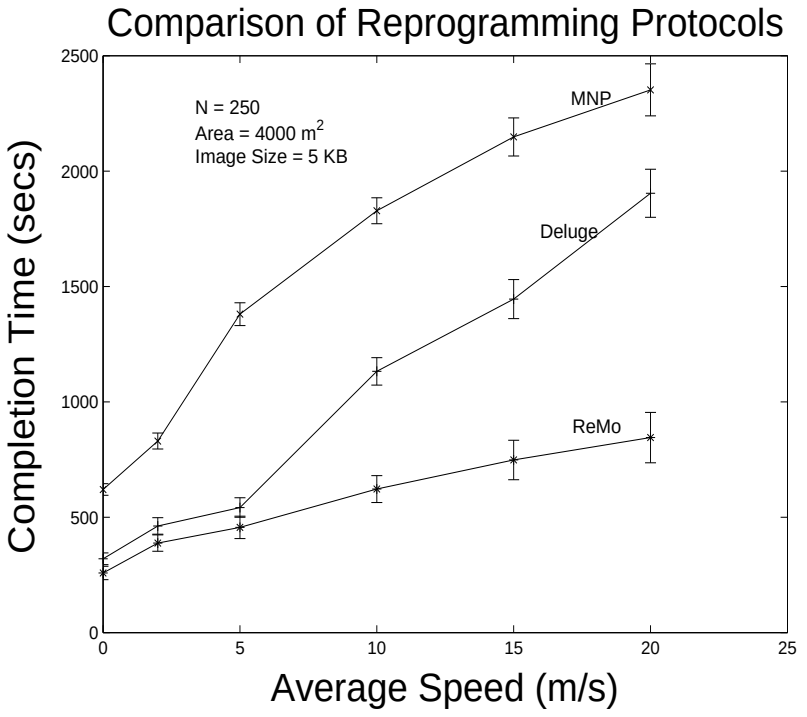
Model Parameter	Value
τ	$\frac{10}{AvgNodeSpeed(m/s)}$ sec
γ	0.4
W	6
θ_{high}	0.9
N_{Th}	η
δ	0.1
R_{max}	3

6.4.1 Reprogramming Completion Time

In this section, we focus on the completion time for transferring an image of size 5 pages with each page of size 1 KB comprised of 16 packets of size 64 bytes each. The initially uniformly randomly deployed nodes move with average speeds of 2, 5, 10, 15 and 20 meters per sec $\pm$ 10%. We also vary the number of nodes moving in the fixed area of the terrain. This lets us study the effects of increasing average

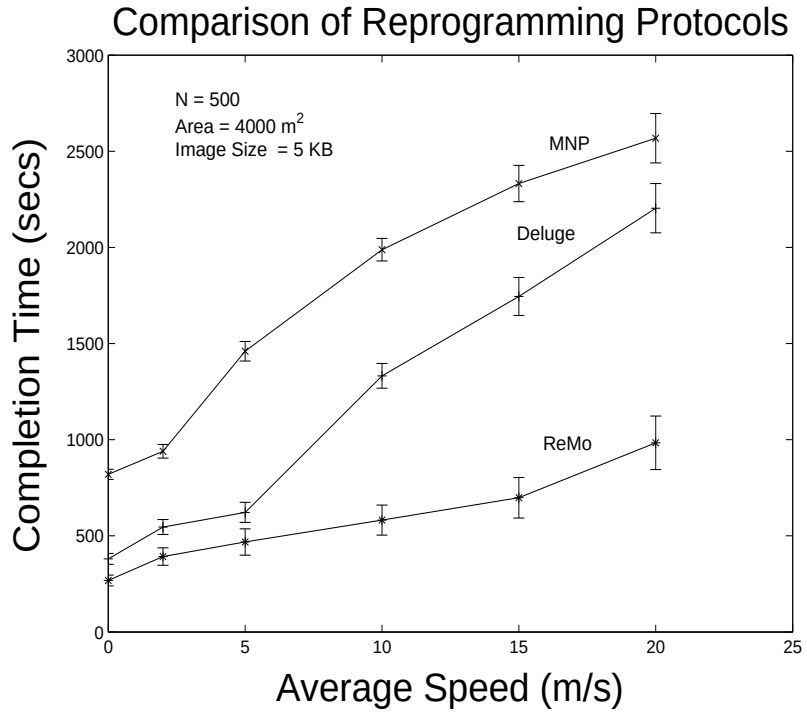


(a)

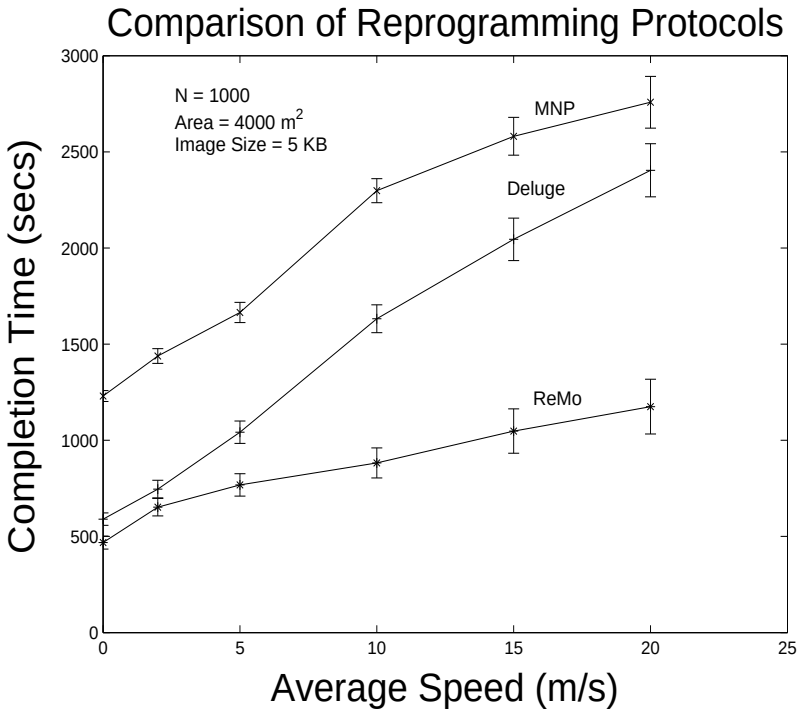


(b)

Figure 6.4. Completion Time of Deluge, MNP and ReMo in a Mobile Sensor Network : (a) 120 and (b) 250 nodes).



(a)



(b)

Figure 6.5. Completion Time of Deluge, MNP and ReMo in a Mobile Sensor Network : (a) 500 and (b) 1000 nodes).

node density on the performance of the protocol. Comparative results are shown in Fig. 6.4 and Fig. 6.4 for 120, 250 and 500 nodes.

Fig. 6.4 (a) depicts the effects of a low average node density on the reprogramming completion time. Without mobility, and at this low node density, we observe that both ReMo and Deluge take almost the same time for propagating 5 pages through the network. However, as the mobility of the nodes increases, ReMo outperforms Deluge. One of the primary reasons is Deluge's constraint of downloading pages in order. Moreover, ReMo adapts better to the dynamic changes in local node density in curbing redundant transmissions of advertisement messages thus promoting faster transfer of data.

MNP gets the most adversely affected by node mobility as the sender selection algorithm fails to optimally select a node in a neighborhood. As per the MNP protocol, nodes wait to gather multiple request packets while having the sender selection algorithm choose a particular sender and other competing advertising nodes go to sleep. However, this delay in serving the requests proves useless because when a sender is finally chosen and scheduled to transmit, it is in a different neighborhood. Thus, certain nodes go to sleep when they do not have to, and some nodes end up transmitting in a neighborhood where they are not required. Subsequently, nodes end up sending more messages for a longer duration until the whole network is finally updated. However, we also notice that the slope of the MNP curve decreases at higher mobility. The possible reason is that the effect of mobility on the sender selection mechanism becomes less dependent on the speed of the nodes after a certain value.

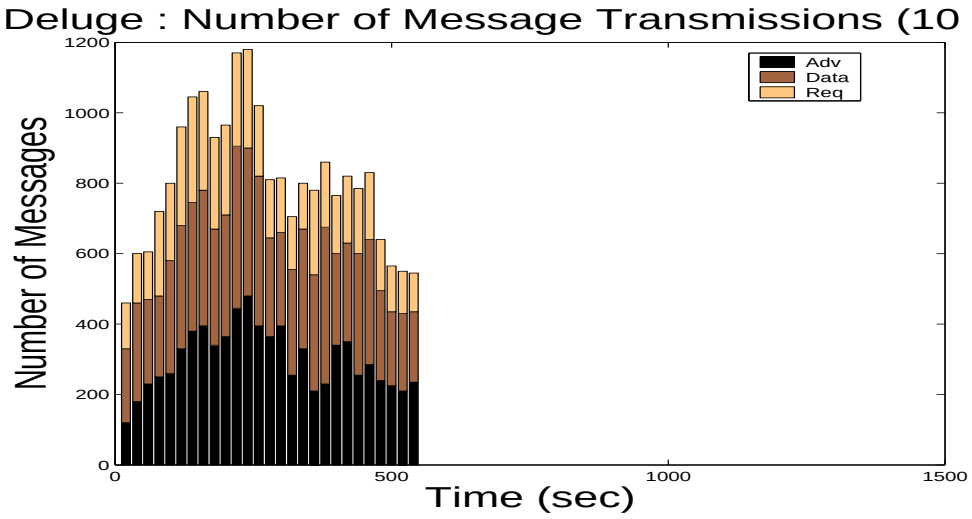
Fig. 6.4 (b) shows the effects of an increased level of average node density. We observe that even at zero mobility, ReMo performs slightly better than Deluge. The reason is that the hidden terminal problem becomes conspicuous and ReMo's probabilistic advertisement mechanism copes with it better than Deluge. However, we observe that a slight increase in node mobility to 5 m/s helps Deluge cope with the

effects of the hidden terminal problem brought about by the increased average node density. Thus, its completion time does not increase significantly and the two curves stay close to each other. However, a further increase of node mobility to 10 m/s and higher causes Deluge to degrade in performance and the difference in completion time with ReMo becomes significant. In Fig. 6.4 (a) and (b), the average node density is increased further and we observe that ReMo continues to show significant improvements over Deluge and MNP. The flexible order for page download and the smoothened probabilistic advertisement mechanism based on neighborhood density helps ReMo overcome the effects of mobility and node density fluctuations better than Deluge and MNP.

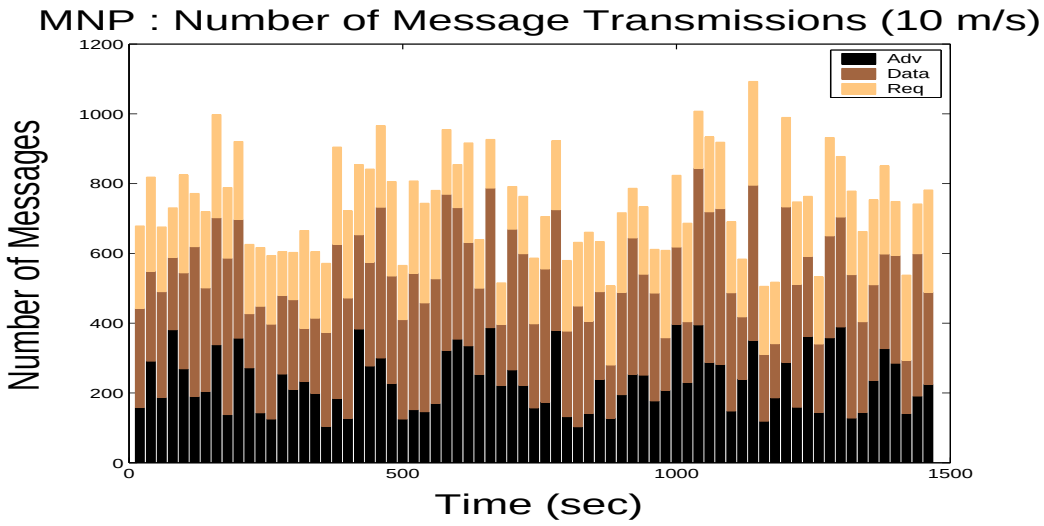
6.4.2 Number of Message Transmissions

We look into the number of messages transmitted by each protocol in fixed sized time windows of 20 seconds for the entire duration of the code update. Fig 6.6 depicts the message transmission distribution for each protocol. There are 120 nodes in a $4000m^2$ area and the nodes move with an average speed of 10 m/s $\pm$ 10%. In Fig 6.6 (a), the overall transmitted messages for Deluge are shown. Comparing with Fig 6.6 (b), we see that although the average number of messages sent in each 20 second slot was less in MNP, the entire duration was much longer. However, in Fig 6.6 (c), the number of messages transmitted by ReMo shows that both the average rate of message transmission and the entire duration of update is less than both MNP and Deluge.

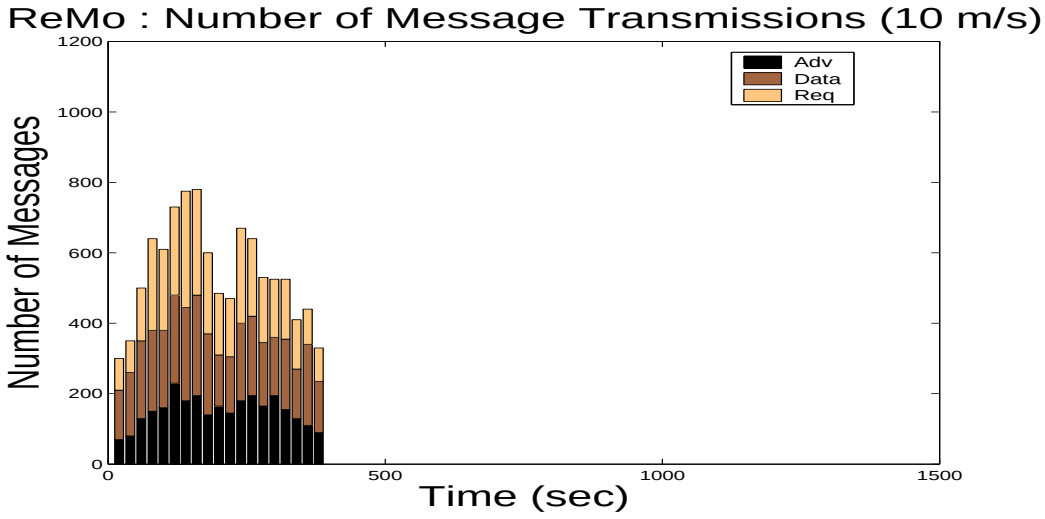
In Fig. 6.7, we look at the number of message transmissions in a denser network of 500 nodes moving at the same average speed of 10 m/s $\pm$ 10%. We observe that Deluge is hit worst by the increase of node density and the number of messages transmitted in each 20 second window is significantly more than the other two protocols. MNP has more latency for the entire completion of code transfer but transmits less



(a)

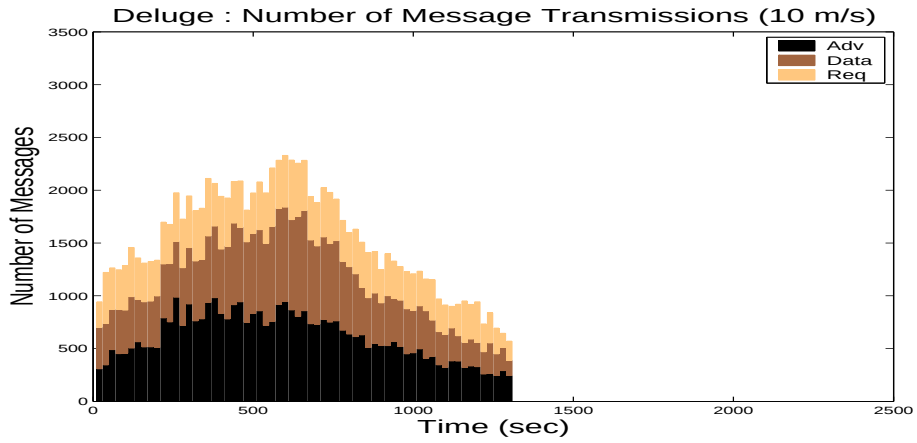


(b)

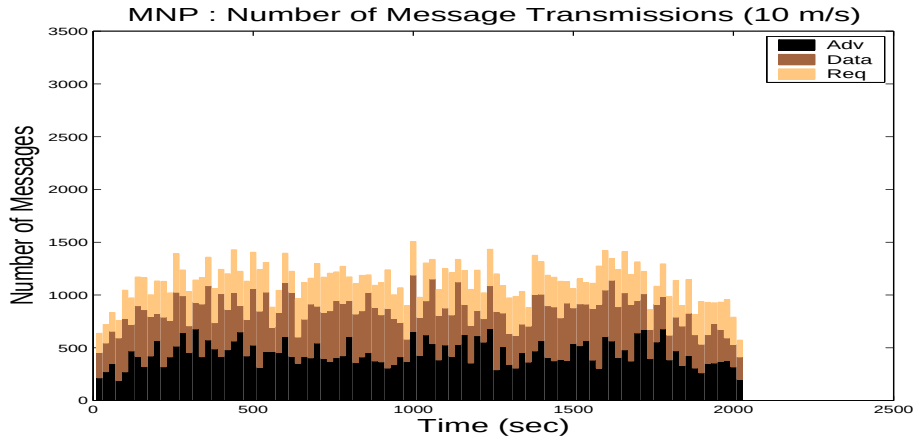


(c)

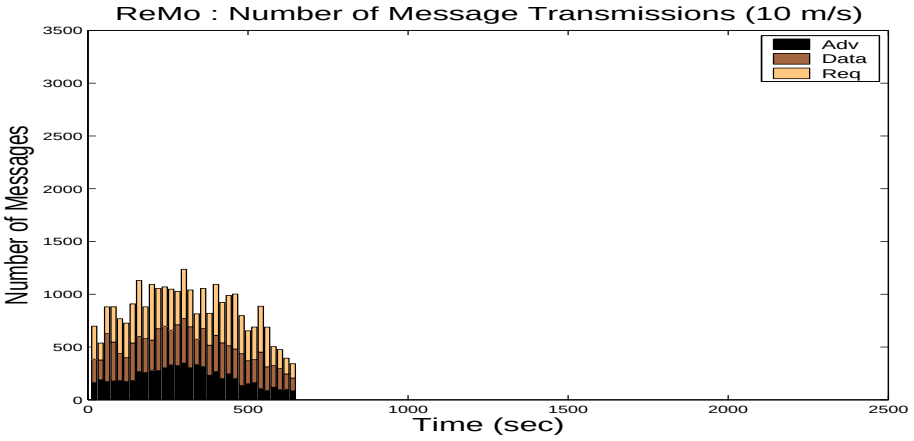
Figure 6.6. Number of Transmitted Messages (120 Nodes) : (a) Deluge, (b) MNP, (c) ReMo.



(a)



(b)



(c)

Figure 6.7. Number of Transmitted Messages (500 Nodes) : (a) Deluge, (b) MNP, (c) ReMo.

number of messages than Deluge in each 20 second window. ReMo completes the transfer in lesser time and also transmits almost half the total number of messages as Deluge does in each time window.

The lower average message transmission rate also indicates that ReMo is more energy efficient than Deluge or MNP in a mobile environment.

6.5 Summary

In this paper, we presented *ReMo*, a reprogramming protocol specifically suited for mobile sensor networks. We showed how the existing reprogramming paradigm for static networks fails to adapt to a mobile scenario. The protocol takes advantage of the mobility of the nodes by having them download pages out-of-order, thus expediting the download process. The protocol also smoothly adapts its periodic metadata advertisements to cope with the constantly varying node density of the mobile environment and suppress redundant transmissions as much as possible, thereby saving valuable energy. Finally, ReMo selects neighbors for code exchange based on better link quality and thus improves on the throughput of data transfer. Our comparative results indicate significant improvements in completion time of reprogramming the whole network and the number of messages transmitted over existing reprogramming protocols like Deluge and MNP. The results are more pronounced as we increase the average node speeds as well as the network size.

In the next chapter, we present our implementation of the ReMo protocol on a testbed of SunSPOT devices.

CHAPTER 7

TESTBED IMPLEMENTATION OF REMO

In this chapter, we discuss the implementaiton of the ReMo [22] protocol which we have implemented on a testbed of SunSPOTs (*Sun's Small Programmable Object Technology*) [86]. This chapter is organized as follows : In section 7.1, we provide a brief overview of the SunSPOT. In section 7.2, we discuss our implementation architecture. We provide details about our experiments in section 7.3 and summarize the chapter in section 7.4.

7.1 Sun Small Programmable Object Technology (SunSPOT)

The SunSPOT is the new sensor device designed at Sun Microsystems Research Laboratories. The unique feature of the SunSPOT is that it runs on Java and there is no operating system. It is comprised of

- *Sun SPOT Processor Board* : The processor board consists of a 180 Mhz 32 bit ARM9OT core. It has 512 KB of RAM and 4 MB of flash memory. It also runs on the CC2420 [87] radio chip having a 2.4 Ghz 802.15.4 radio and has an integrated antenna. It uses a USB port for programming and provisioning and runs on a 3.7V rechargeable 720 mAh lithium-ion battery. It also has a double sided connector for stackable boards.
- *General Purpose Sensor Board* : The SunSPOT comes with a general purpose sensor board which has a 2G/6G 3-axis accelerometer, a temperature sensor, a light sensor, 8 tri-color LEDs and two momentary switches that can be programmed. It also has 6 analog inputs, 5 general purpose I/O pins and 4 high current output pins.

- *Squawk Java Virtual Machine* : The device uses a fully capable J2ME CLDC 1.1 Java Virtual Machine [72] with operating system functionalities. The VM executes directly out of flash memory. Moreover, the device drivers are also written in Java.

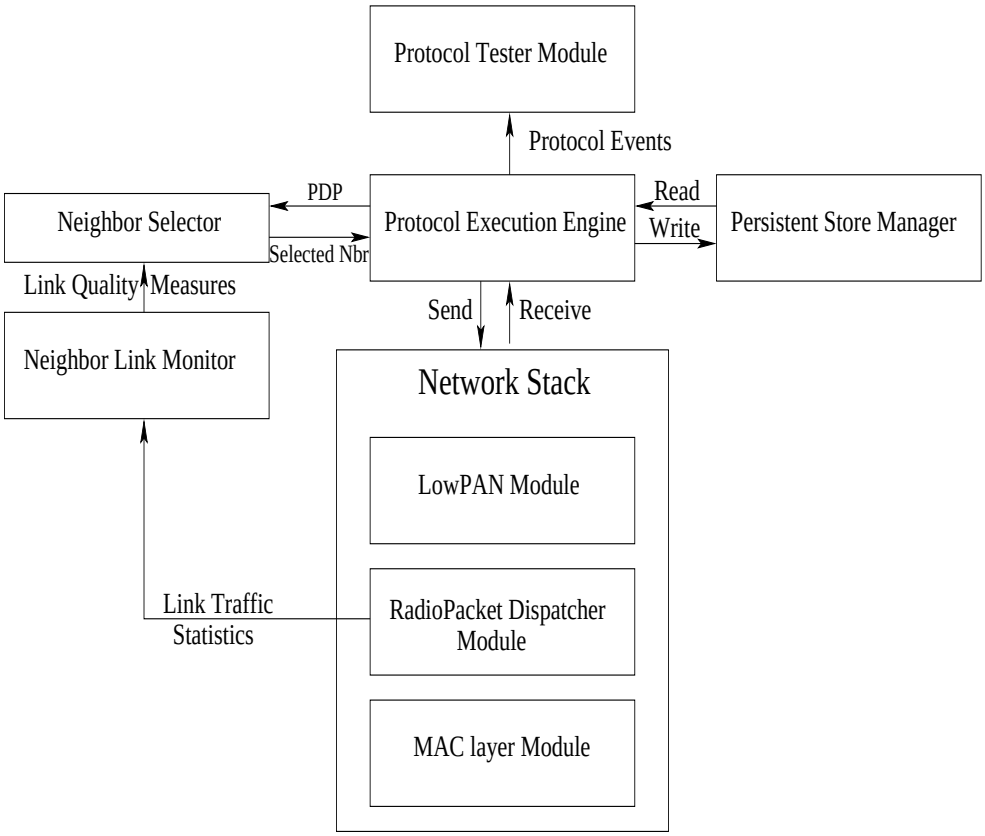


Figure 7.1. ReMo : Protocol Architecture.

Further details are available at SunSPOTWorld [86].

7.2 Implementation Architecture

We have implemented a prototype of ReMo and its architecture as is depicted in figure 7.1. Central to the architecture is the *Protocol Execution Engine* where the protocol finite state machine executes. The architecture is implemented at the application level and runs atop the network stack of the SunSPOT. A separate module

called the *Neighbor Link Monitor* registers with the low-level *Radio Packet Dispatcher* module of the network stack to be notified of packets transmitted and received. It uses these notification events to build important statistics about the link quality with each neighbor. A *Neighbor Selector* module combines the information about the link quality information from the link monitor and the page download potential from the protocol engine to select the best neighbor to download code from. A separate module called the *Persistent Store Manager* controls and manages the operations of reading and writing each page from the flash memory.

7.2.1 The Protocol Execution Engine

The Protocol Execution Engine is the core of the architecture and handles the periodic broadcast paradigm of metadata and the transmission/reception of the data and control packets. The entire state machine is implemented in this module. Moreover, this module maintains a runtime description of the image currently running on the SPOT and also a persistent copy of the metadata that is being broadcast periodically. A timer is also implemented in this module to help with the periodic broadcasts of the metadata. The *TransmissionProb* algorithm for smoothly monitoring the neighborhood density for modifying the broadcast probability is implemented in this module as well.

ReMo initializes itself by populating fields to identify the proper locations of the image in flash and constructing a metadata object for periodic broadcast. Then it initializes the state machine engine and invokes the threads corresponding to the transmitter, the receiver, and the timed task for broadcasting the metadata object. It then initializes the Neighbor Link Monitor. A snippet of code depicting the initialization of the protocol engine is depicted in Fig. 7.2.

```
void initEngine() throws IOException {
        /* Instantiate the states */
        advState =  new ADVState();
        reqState = new REQUESTState();
        transmitState = new TRANSMITState();
        downloadState = new DOWNLOADState();
        nodeDes = NodeDescription.getInstance();
        /* Initialize the state machine to the ADV State */
        fsmState = advState;
        nodeDes.setNodeFSMState(fsmState);
        /* Setup Relevant Ports and Datagrams */
         ...
        /* Timed Task to schedule regular broadcast
           transmission of metadata */
        tauExpTask = new TauExpiredTask();
        startCommunicationThreads();
        /* Start the link monitor task */
        NodeLinkMonitor.getInstance().initialize();
}
```

Figure 7.2. Protocol Engine initialization code snippet.

7.2.2 Neighbor Link Monitor

The Neighbor Link Monitor initializes the task for the periodic listening of the neighborhood and constructing relevant link quality statistics. As neighbors are discovered they are stored in a table and their normalized link quality metric dynamically updated. A neighbor is marked as *stale* in the table and subsequently flushed if it is detected as inactive for a threshold time window. This module is implemented at the level of the LowPAN layer of the network stack, receiving notification from the *RadioPacketDispatcher* layer on any packet received. In other words, the statistics is built based on the reception of any packet by the node and is independent of the messages of the protocol itself.

The Neighbor Selector Module is very closely associated with the Neighbor Link Monitor and it uses the link statistics from the latter and using the page download

potential weights calculated with each neighbor, returns the best neighbor to send a request packet to.

7.2.3 Persistent Store Manager

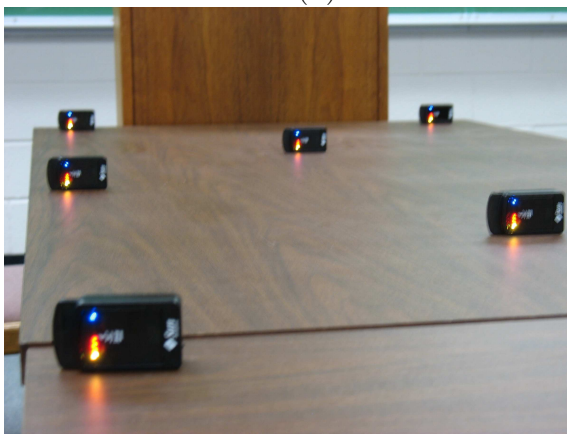
The persistent store manager maintains persistent storage of the currently received code image, the metadata that is being currently used for periodic broadcast and the dynamic bit vector used to signify the current status of the code pages that have been downloaded. We have used the *Record Management Store* implementation on the SunSPOT for the flash storage of the code pages. The persistent store manager is closely coupled with the execution engine and performs the relevant storage and retrieval of page and metadata information at different states of execution of the protocol. For instance, when it hears a new metadata in the *Advertisement* state it stores the new metadata in the flash and transits to the *Request* state to request data packets from potential senders. Moreover, methods of the store manager are invoked in the *Transmit* and *Download* states, for reading and writing pages to the flash, respectively.

7.2.4 Protocol Tester Module

This module is a test application that we wrote to test the working of our protocol. This test module uses the *Manifest* file with each J2ME midlet application to set some manifest properties which could be used to set up the functioning and testing of ReMo. For instance, we had an image version number, an image identification string, etc. set as configurable parameters in the manifest file. The image version number is a *red-green-blue* identifier string giving a color combination which is representative of the code version executing on the current SunSPOT. We have used the first six from the array of eight LEDs on the SunSPOT to reflect the image version of the code executing on it. More specifically, we have used these LEDs as a progress bar to

(a)

(b)



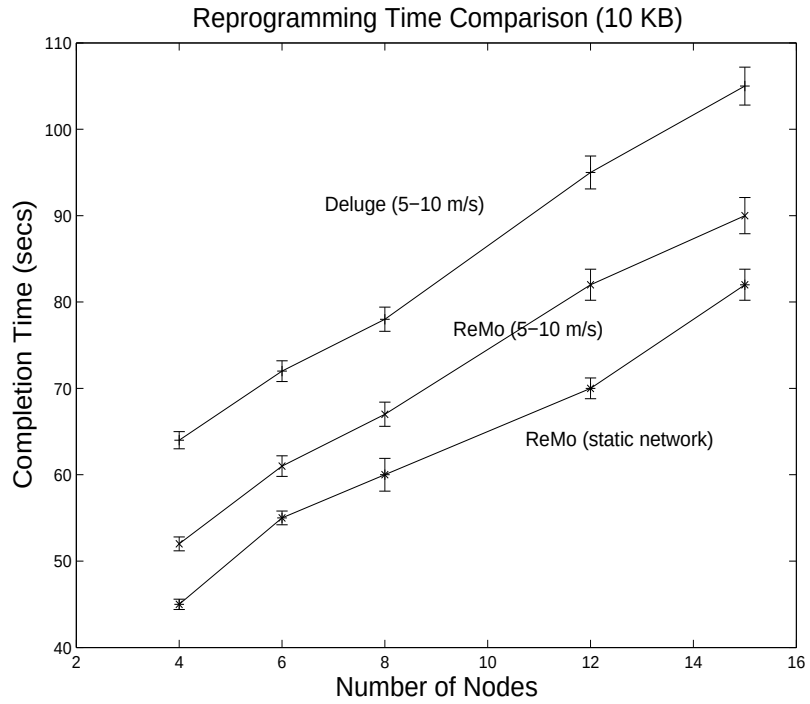
(c)

Figure 7.3. Snapshots of new image propagation from source : (a) Propagation Initiation, (b) Propagation Progress, (c) Further Progress.

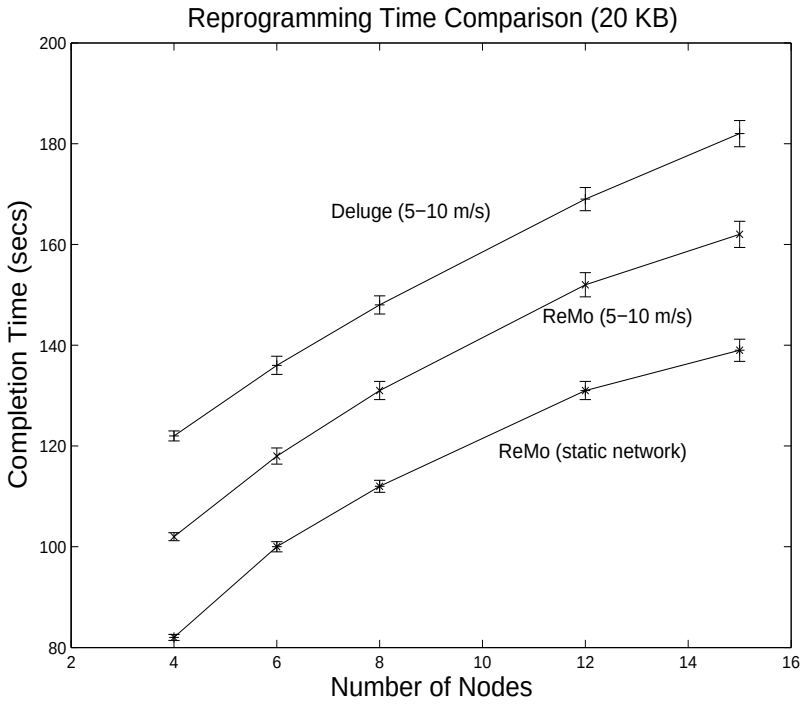
```
public boolean startApp() {
    ...
    String imageName = Utils.getManifestProperty(
                        "MIDlet-Name", "NULL");
    String imageVersion = Utils.getManifestProperty(
                        "ImageVersion", "0");
    String imageId = Utils.getManifestProperty(
                        "ImageID", "255-255-255");
    ...
    /* Start ReMo */
    ReMo.getInstance().startReMo(imageName, imageVersion,
                                 imageId, imageSize);
    /* Register for Events from ReMo */
    ReMo.getInstance().registerForReMoEvents(
                        new ReMoEventListener(imageId));
    ...
}
```

Figure 7.4. Testing module code snippet.

show the code download process at a SunSPOT. This module implements an event listener and upon notification of appropriate events lights up the corresponding LED to show the protocol operation. In Fig. 7.3, we observe two snapshots of ReMo executing in a small experimental testbed. In Fig. 7.3 (a), if we observe closely, we find that the array of LEDs in the nearest node is *red*, while the rest is *yellow*. This is the beginning of the reprogramming phase. The red version is a more recent one and is about to replace the yellow version in the rest of the SunSPOTs. in Fig. 7.3 (b), we observe the reprogramming in progress, where part of the red code has been downloaded in the nodes. However, if we observe closely, the first LED is still yellow signifying that the entire red code has not been downloaded and the previous code was yellow. When the complete red code version is downloaded, the entire array would turn red signifying the completion of the process. The code snippet from this module is depicted in Fig. 7.4.

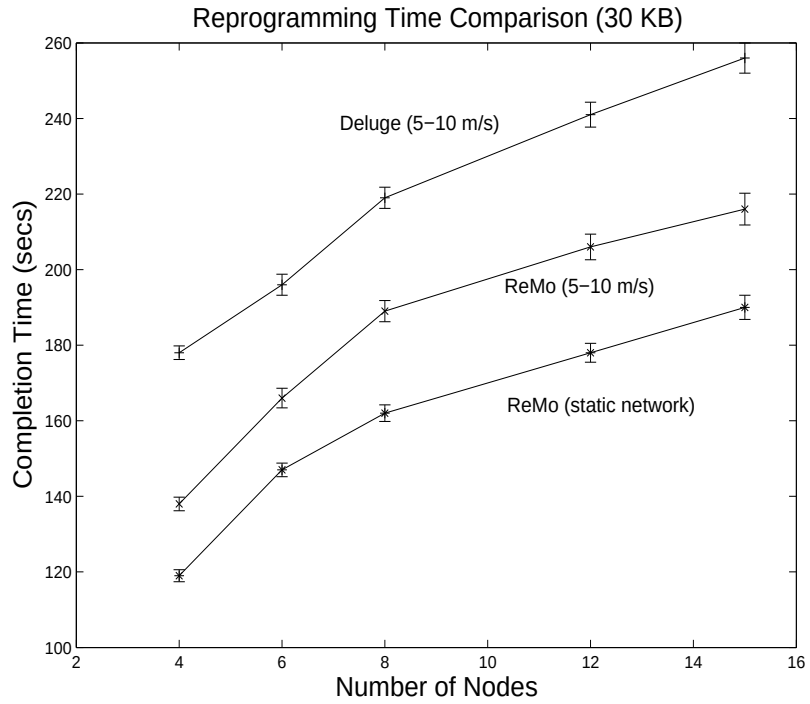


(a)

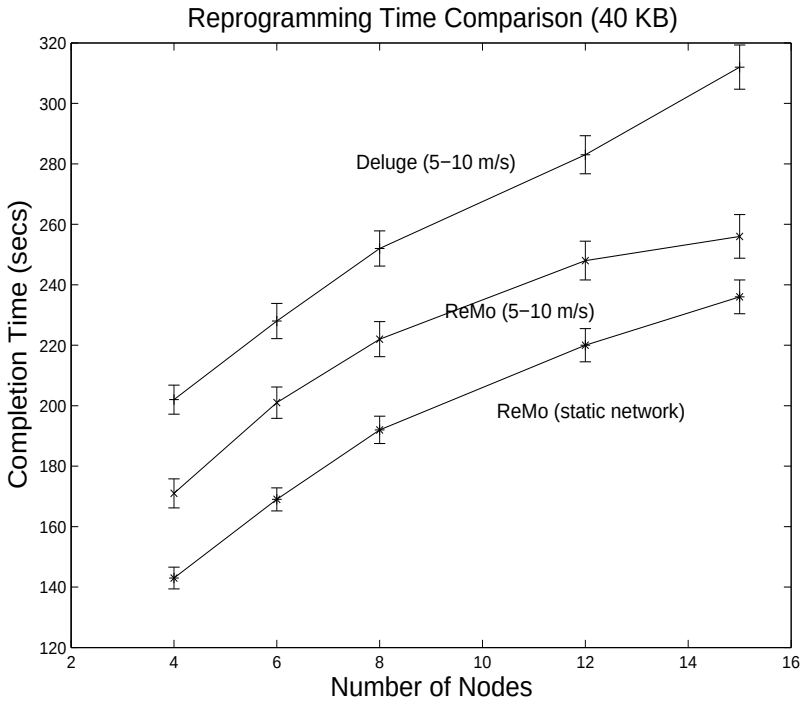


(b)

Figure 7.5. Transfer Completion Times for different data sizes : (a) 10 KB Transfer, (b) 20 KB Transfer.



(a)



(b)

Figure 7.6. Transfer Completion Times for different data sizes : (a) 30 KB Transfer, (b) 40 KB Transfer.

7.3 Experimental Results

In this section, we discuss the results of the testbed experiments we have performed for evaluating the efficacy of ReMo. We evaluate our implementation by transferring different code sizes over ReMo and Deluge in a network of at most 15 SunSPOTs. We manually move the SunSPOTs around randomly during the code transfer duration at an average speed of $5 - 10m/s$. The transmit power of each SunSPOT has been reduced to $-29dB$ so that the transmission range for each of them is approximately between 1 to 1.5 feet, thus helping us in performing the experiment in a limited area and forcing multihop propagation. We have considered executable image sizes of the order of typical sample demo applications that are shipped with the SunSPOT software development kit. For instance, the *Airtext* demo application's binary suite file that gets deployed onto a SunSPOT is around 9 KB. Similarly, the *BounceDemo* application is around 33 KB. In Figs. 7.5 and 7.6, we depict the plots for the completion time of ReMo in comparison with Deluge for different image sizes. We observe in Fig. 7.5 (a), that at smaller image sizes and also with considerably small size of the network, the curves depict a linear nature of code transfer time. However, mobility still affects the transfer time , as we see that in comparison to a static scenario, ReMo and Deluge take longer time. As expected, ReMo outperforms Deluge because it chooses better links and downloads pages as they arrive without explicitly maintaining the order. We observe the effects of pipelining the transfer in Fig. 7.5 (b), where the slope of the curves dip slightly when the network is around 15 SunSPOTs. This is also an expected result and we further observe that Deluge takes performs worse than ReMo when the nodes are mobile with an average speed of $5 - 10m/s$.

However, as we increase the image size further to around 40 KB, as depicted in Fig 7.6 (b), we notice that the performance of Deluge worsens further in the face of

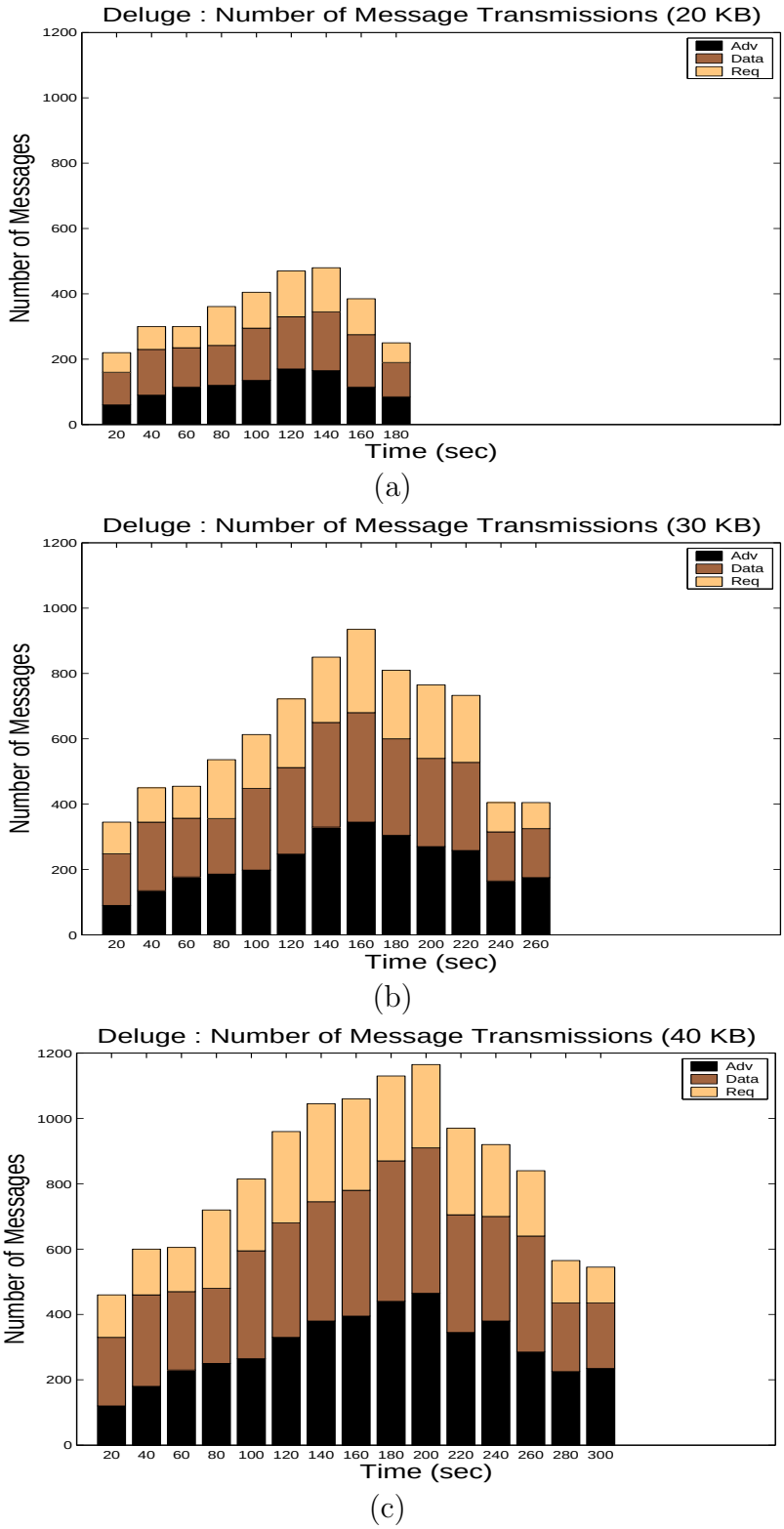


Figure 7.7. Number of Messages Transmitted under Deluge (5-10 m/s) : (a) 20 KB Transfer, (b) 30 KB Transfer, (c) 40 KB Transfer.

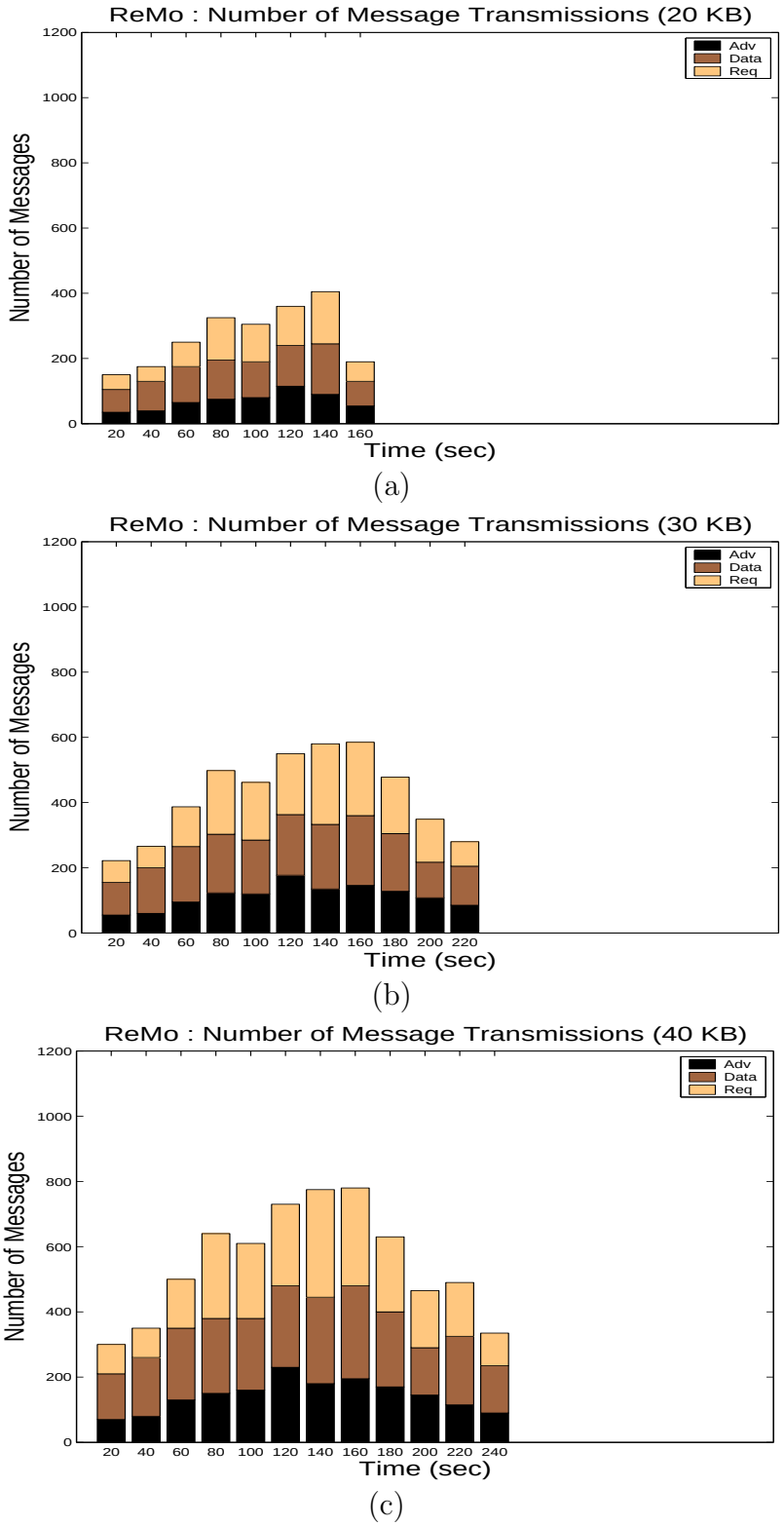


Figure 7.8. Number of Messages Transmitted under ReMo (5-10 m/s) : (a) 20 KB Transfer, (b) 30 KB Transfer, (c) 40 KB Transfer.

mobility. Constrained under the restriction of maintaining page order and also not choosing neighbors based on better link qualities, Deluge suffers due to more packet losses and longer wait times due to the order restriction. ReMo, on the other hand, adjusts to the mobility and thus takes full advantage of the pipelined page transfer in the network, allowing nodes to download pages as and when they are available from the neighbors.

In Figs. 7.7 and 7.8, we depict the number of message transfers done by both Deluge and ReMo in a mobile environment where nodes are moving roughly at an average speed of $5 - 10$ m/s. The results were based on a network size of 15 nodes. We observe that even in a comparatively small network size and moderate speed, ReMo still fares significantly better than Deluge in terms of the number of messages transmitted. As depicted by our earlier simulation study, we expect this difference in performance to get magnified as the network and data object size increases. The reasons, as stated earlier, are primarily due to the ordered page download of Deluge,the choice of neighbors made by ReMo based on link quality and the better management of metadata broadcasts based on neighborhood density. ReMo thus saves significantly in the number of messages transmitted making it more energy efficient in a mobile environment. In particular, the total number of messages transmitted in ReMo is 32% less than Deluge for 20KB transfer which increases to around 46% less messages for transferring 40KB.

7.4 Summary

In this chapter, we have discussed the implementation of ReMo, a reprogramming protocol for mobile sensor networks, on a testbed of SunSPOTs, which is a new class of wireless sensor devices made by Sun Research Laboratories. We have tested our protocol on a maximum network size of 15 SunSPOTs, emulating a multihop net-

work by reducing the transmit power of the nodes sufficiently to generate a network diameter of 3-4 hops. Our implementation was in the Java language executed on the Squawk Java Virtual Machine running on the SunSPOT. The protocol was implemented at the application layer with callback hooks established with the lower layers of network stack for obtaining link quality information with surrounding neighbors. We have compared ReMo with an implementation of Deluge under different mobility scenarios and network size. Although small in comparison to the network sizes used in our simulation, our results are in tandem with the effects we perceive in our simulation, i.e., ReMo showing pronounced performance benefits as the average node speeds, the network size and the transferred code size increases.

CHAPTER 8

CONCLUSIONS

In this chapter we summarize our work in this dissertation. Having asserted the importance of the broadcast based communication paradigm in wireless sensor networks, we have focused on the construction of mathematical models for performance and security analyses of data dissemination in sensor networks. Our performance model provides a convenient tool for analysis of different data dissemination protocols in terms of their data propagation rate and network reachability. The model has the flexibility to allow different protocols to be plugged into it by expressing one if its key parameters, namely the infection rate, in terms of the communication rate of the particular protocol. On the other hand, our security models expose the comparative vulnerabilities of these protocols to piggybacked malware. As much as these protocols are useful in transporting bulk data throughout the network, they are equally susceptible to compromise when an adversary could use them for propagating harmful code. Although some of the existing works deal with authenticating the source, there are shortcomings when used under different application scenarios. In our study, we try to use our model to make this assertion, and capture the comparative vulnerabilities of the protocols in spreading the malware. Specifically, we look into the aspect of how the malware would behave in the face of a simultaneously working recovery procedure.

Our work also throws significant light on how different sensor node deployment strategies would react to the propagation of node compromise within a network and the dynamics of how a simultaneous recovery procedure can contain such a malicious process under the different deployment strategies. In particular, we have assumed the

network to be already securely communicating by way of shared secret keys which are randomly pre-distributed at deployment time. Our assertion is that in such a scenario, owing to the probabilities of overlap of key rings between neighbors, a malware which is residing in a few compromised nodes, can exploit this aspect and gradually infect more nodes and eventually the entire network. In our study, we have identified the key network parametric points that define an epidemic breakout of the compromise under separate network deployment types.

Finally, we have appreciated and presented the performance drawbacks faced by existing data dissemination protocols when applied to a mobile scenario. Several characteristics of the previous protocols rendered them inefficient in a mobile environment. Subsequently, we have proposed a novel protocol, ReMo, suitable for reprogramming a mobile sensor network. The features of ReMo that make it perform better than existing protocols are the relaxation of strict ordered download of pages, the incorporation of measured link qualities in the selection of neighbors for data exchange, and the smoothly varying broadcast rate based on the highly dynamic node density in the mobile environment. We have implemented ReMo in a testbed of sensor devices called SunSPOTs.

In essence, in this dissertation, we have focused on a comprehensive study of broadcast based data propation over dissemination protocols in sensor networks and looked at the subject not only from the angle of performance measures and security aspects in a formal manner, but also presented the shortcomings of existing dissemination protocols when used in a mobile scenario and went on to propose a new protocol for a mobile network.

8.1 Future Work

As part of our future work, we are extending our model and formal analysis on data propagation and vulnerability study on the dissemination protocols to encompass network mobility. The primary modification required by the model is the fact that, in a mobile scenario, the restricted interactions between the infective and susceptible nodes would be relaxed and nodes would be able to homogeneously mix. This actually would simplify the model to a certain extent. Apart from that, we would still be able to plugin the individual infection rates of each protocol to bring out their behavior.

Since security of these protocols is a critical issue owing to the very nature of their operation, we are extending our design and trying to incorporate security features into ReMo. The security feature would provide authentication of the source of the data disseminated and preserve the integrity of the same. Existing hash chain based mechanisms would fail in the case of ReMo owing to its unordered dissemination approach. As a result, we require efficient mechanims which would not assume any particular page order of download and be lightweight enough to be supported by sensor devices.

REFERENCES

[1] M. Akdere, C. C. Bilgin, O. Gerdaneri, I. Korpeoglu, O. Ulusoy, U. Cetintemel, "A comparison of epidemic algorithms in wireless sensor networks", In *Computer Communication*, 2006.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks", In *IEEE Communications Magazine*, vol. 40, no. 8, 2002.

[3] R. M. Anderson and R. M. May, "Infectious Diseases of Human: Dynamics and Control", *Oxford Univ. Press*, Oxford, 1991.

[4] N. T. J. Bailey, "The Mathematical Theory of Infectious Diseases and its Applications". Hafner Press, New York, 1975.

[5] M. Barthelemy, A. Barrat, R. Pastor-Satorras, A. Vespignani. "Velocity and Hierarchical Spread of Epidemic Outbreaks in Scale-Free Networks". In *Phys Rev Lett*, Vol 92, 178701, 2004.

[6] C. Bettstetter, "On the minimum node degree and connectivity of a wireless multihop network", In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking and computing*,(MobiHoc) 2002.

[7] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. "Analysis and optimization of randomized gossip algorithms", In Proceedings of the 43rd Conference on Decision and Control (CDC 2004), 2004.

[8] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. "Gossip algorithms : Design, analysis and applications", In Proceedings of the 24th Conference of the IEEE Communications Society (INFOCOM 2005), 2005.

[9] D. Braginsky, and D. Estrin, "Rumor routing algorthim for sensor networks", In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, (WSNA) 2002.

[10] T. Camp, J. Boleng and V. Davies, "A Survey of Mobility Models for Ad Hoc Networks Research", *Wireless Communications and Mobile Computing*. Volume 2, Number 5. 2002

[11] A. Chadha, Y. Liu. and S. Das, "Group key distribution via local collaboration in wireless sensor networks," In *Proceedings of the IEEE International Conference on Sensor and Ad Hoc Communications and Networks* (SECON) 2005.

[12] H. Chan; V.D. Gligor, A. Perrig, G. Muralidharan, "On the Distribution and Revocation of Cryptographic Keys in Sensor Networks", In *IEEE Transactions on Dependable and Secure Computing*, Vol 2, Issue 3, 2005.

[13] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks", *in Proc. of the IEEE Symposium on Research in Security and Privacy* (SP) 2003.

[14] J. Y. Chen and D. X. G. Pandurangan. "Robust aggregates computation in wireless sensor networks: Distributed randomized algorithms and analysis". In Fourth International Symposium on Information Processing in Sensor Networks (IPSN), 2005.

[15] C. Chong and S. Kumar, "Sensor networks: Evolution, opportunities, and challenges", In *Proceedings of the IEEE*, vol. 91, no. 8, 2003.

[16] M. Chu, H. Haussecker, and F. Zhao, "Scalable Information-Driven Sensor Querying and Routing for ad hoc Heterogeneous Sensor Networks," In *The International Journal of High Performance Computing Applications*, Vol. 16, No. 3, August 2002.

[17] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs", *Discrete Math*, Vol. 86, 1-3 (Jan. 1991).

[18] P. De, Y. Liu, and S. K. Das, "Modeling Node Compromise Spread in Sensor Networks using Epidemic Theory", In *IEEE World of Wireless, Mobile and Multimedia Networks*, (WoWMoM) 2006.

[19] P. De, Y. Liu, and S. K. Das, "An Epidemic Theoretic Framework for Evaluating Broadcast Protocols in Wireless Sensor Networks", In *The Fourth IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, (MASS) 2007.

[20] P. De, Y. Liu, and S. K. Das, "A Framework for Vulnerability Analysis of Broadcast Protocols in Sensor Networks", accepted for publication in *IEEE Transactions on Mobile Computing* (TMC), 2008.

[21] P. De, Y. Liu, and S. K. Das, "ReMo : An Energy efficient Reprogramming Protocol for Mobile Sensor Networks", To appear in *The Sixth IEEE International Conference on Pervasive Computing and Communications* (PerCom) 2008.

[22] P. De, Y. Liu, and S. K. Das, "A Prototype for an Energy Efficient Protocol for Reprogramming Mobile Sensor Networks", demonstration and poster presentation at *The 6th IEEE International Conference on Pervasive Computing and Communications* (PerCom) 2008.

[23] I. De, S. Pool and M. Kochen, "Contacts and influence", Social Networks, 1 (1978), pp. 148.

[24] J. Deng, R. Han, and S. Mishra, "Secure code distribution in dynamically programmable wireless sensor networks", In *The Fifth International Conference on Information Processing in Sensor Networks* (IPSN) 2006.

[25] A. G. Dimakis, A. D. Sarwate, and M. J. WainWright, "Geographic Gossip: Efficient Aggregation for Sensor Networks", In Fifth International Symposium on Information Processing in Sensor Networks (IPSN) 2006.

[26] W Du, J Deng, YS Han, S Chen, PK Varshney, " A key management scheme for wireless sensor networks using deployment knowledge", In *Proceedings of IEEE Conference on Computer Communications* (INFOCOM) 2004.

[27] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler. "Securing the network programming system", In *The Fifth International Conference on Information Processing in Sensor Networks* (IPSN) 2006.

[28] L Eschenauer and V. D. Gligor. "A key-management scheme for distributed sensor networks", In *Proceedings of the 9th Computer Communication Security* (CCS) 2002.

[29] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks" Intel Research, Berkeley Technical Report IRB-TR-02-003, March, 2002.

[30] A. Goel, S. Rai, and B. Krishnamachari, "Sharp thresholds For monotone properties in random geometric graphs", In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing* 2004.

[31] P. Grassberger, "On the critical behavior of the general epidemic process and dynamic percolation", In *Math. Biosc.* Vol 63, 157 1983.

[32] C. Griffin, R. Brooks, "A note on the spread of worms in scale-free networks", In IEEE Transactions on Systems, Man and Cybernetics, Vol 36, Issue 1, Feb, 2006.

[33] Z. J. Haas, J. Y. Halpern, and L. Li, "Gossip-Based Ad Hoc Routing", In INFOCOM 2002.

[34] C. Hartung, J. Balasalle, and R. Han, "Node Compromise in Sensor Networks: The Need for Secure Systems", *Technical Report CU-CS-990-05* 2005.

[35] H. W. Hethcote, "Mathematics of infectious diseases". In *Society for Industrial and Applied mathematics (SIAM) Review* Vol 42, No. 4, pp 599-653, 2000.

[36] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale", In *The Second ACM Conference on Embedded Networked Sensor Systems*, (SenSys) 2004.

[37] C. Intanagonwiwat, R. Govindan, and D. Estrin. "Directed diffusion: a scalable and robust communication paradigm for sensor networks". In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking* (MOBICOM), 2000.

[38] J. Kahn, R. Katz, and K. Pister. Next century challenges: mobile networking for smart dust. In International Conference on Mobile Computing and Networking (MOBICOM 99), pages 271278, Aug. 1999.

[39] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking, "Randomized rumor spreading", In Proc. IEEE Conference of Foundations of Computer Science, (FOCS), 2000.

[40] B. Karp and H. T. Kung. "GPSR: greedy perimeter stateless routing for wireless networks". In *International Conference on Mobile Computing and Networking* (MobiCom 2000), pages 243254, Boston, MA, USA, 2000.

[41] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information", In Proc. IEEE Conference of Foundations of Computer Science, (FOCS), 2003.

[42] J. O. Kephart, D. M. Chess, and S. R. White, "Computers and Epidemiology". In *IEEE Spectrum*, 1993.

[43] J. O. Kephart and S. R. White, "Measuring and Modelling Computer Virus Prevalence". In *Proceedings of the IEEE Symposium on Security and Privacy* (SP), 1993.

[44] J. Kephart and S. White. Directed-graph epidemiological models of computer viruses. In Proceedings of the IEEE Computer Symposium on Research in Security and Privacy, pages 343359, May 1991.

[45] S. A. Khayam and H. Radha, "A Topologically-Aware Worm Propagation Model for Wireless Sensor Networks", In *IEEE ICDCS International Workshop on Security in Distributed Computing Systems* (SDCS), 2005.

[46] J. Kulik, W. R. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," In *Wireless Networks*, Volume: 8, pp. 169-185, 2002.

[47] J. Kulik, W. Rabiner, H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks", In *Proceedings of the Fifth ACM/IEEE Mobicom Conference*, 1999.

[48] S. S. Kulkarni and M. Arumugam, "INFUSE: A TDMA based Data Dissemination Protocol for Sensor Networks, *Technical Report MSU-CSE-04-46, Department of Computer Science, Michigan State University*, 2004.

[49] S. S. Kulkarni, L. Wang, " MNP: Multihop Network Reprogramming Service for Sensor Networks", In *The 25th IEEE International Conference on Distributed Computing Systems* (ICDCS), 2005.

[50] P. Kyasanur, R. R. Choudhury, and I. Gupta. "Smart gossip: An adaptive gossip-based broadcasting service for sensor networks", In *The Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems* (MASS) 2006.

[51] P. E. Lanigan, R. Gandhi, P. Narasimhan, "Sluice: Secure Dissemination of Code Updates in Sensor Networks", In *IEEE International Conference on Distributed Computing Systems* (ICDCS) 2006.

[52] P. Levis and D. Culler "The Firecracker Protocol", In *The 11th ACM SIGOPS European Workshop*, 2004.

[53] P. Levis, N. Patel, D. Culler, and S. Shenker. "Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks", In *First USENIX/ACM Symposium on Network Systems Design and Implementation*, (NSDI) 2004.

[54] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks", In *Proceedings of the 10th ACM Conference on Computer and Communications Security*(CCS) 2003.

[55] D. Malan, M. Welsh, M. Smith, "A Public-Key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography", In *Proceedings of the IEEE International Conference on Sensor and Ad Hoc Communications and Networks* (SECON) 2004.

[56] R. M. May and A. L. Lloyd, "Infection dynamics on scale-free networks", In *Phys. Rev. E*, Vol 64, 066112, 2001.

[57] C. Moore and M. E. J. Newman, "Epidemics and percolation in small- world networks". In *Phys. Rev. E*, Vol 61, pp 5678-5682, 2000.

[58] V. Naik, A. Arora, P. Sinha, and H. Zhang. "Sprinkler: A reliable and energy efficient data dissemination service for wireless embedded devices, In *Proceedings of the 26th IEEE Real-Time Systems Symposium*, December 2005.

[59] M. E. J. Newman, S. H. Strogatz, D. J. Watts, "Random graphs with arbitrary degree distributions and their applications". In *Phys Rev E* Vol 64, 026118, 2001.

[60] M. E. J. Newman, "Spread of epidemic disease on networks", In *Phys. Rev. E*, Vol 66, 016128, 2002.

[61] R. Pastor-Satorras and A. Vespignani, "Epidemic Spreading in scale-free networks". In *Phys. Rev. Lett* Vol 86, pp 2909-2912, 2001.

[62] R. Pastor-Satorras and A. Vespignani, "Epidemic dynamics and endemic states in complex networks", In *Phys. Rev. E*, Vol 63, 066117, 2001.

[63] M. D. Penrose, "On k-connectivity for a geometric random graph. *Random Structures and Algorithms*, 1999.

[64] M. D. Penrose, "Random geometric graphs", *Oxford Studies in Probability*, vol. 5, Oxford University Press, May 2003.

[65] R. D. Pietro, A. Mei, L. V. Mancini, A. Panconesi, and J. Radhakrishnan. "Sensor Networks that Are Provably Resilient", In *Proceedings of the 2nd IEEE International Conference on Security and Privacy for Emerging Areas in Communication Networks*, 2006.

[66] J. Polastre, J. Hill, and D. Culler. "Versatile low power media access for wireless sensor networks", In *The Second ACM Conference on Embedded Networked Sensor Systems* (SenSys), 2004.

[67] G.Pottie and W. Kaiser. Wireless integrated network sensors. Communications of the ACM, 3(5):5158, May 2000.

[68] T. S. Rappaport, "Wireless Communications: Principles and Practice," *Prentice-Hall*, 2nd ed., December 2001.

[69] S. Schlosser, J. Griffin, D. Nagle, and G. Ganger. Designing computer systems with mems-based storage. In Proceedings of ASPLOS, June 2000.

[70] C. Shannon and D. Moore, "The Spread of the Witty Worm", In *IEEE Security & Privacy*, vol. 2, no. 4, 2004.

[71] E. Shih, P. Bahl, and M. J. Sinclair. "Wake on wireless:: an event driven energy saving strategy for battery operated devices". In *Proceedings of the eighth annual international conference on Mobile computing and networking*, pages 160171. ACM Press, 2002.

[72] D. Simon and C. Cifuentes, "The Squawk Virtual Machine : Java(TM) on the Bare Metal, *extended abstract* at OOPSLA 2005.

[73] K. Srinivasan and P. Levis. "Rssi is under appreciated". In *Proceedings of the Third Workshop on Embedded Networked Sensors* (EmNets), 2006.

[74] S. Staniford, V. Paxson, and N. Weaver. "How to Own the Internet in Your Spare Time". In *Proceedings of the 11th USENIX Security Symposium*, August, 2002.

[75] D. Stauffer, A. Aharony, "Introduction to Percolation Theory", Taylor and Francis, London, 1994.

[76] Y. C. Tseng, S. Y. Ni, and E. Y. Shih, "Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network," *IEEE Transactions on Computers*, vol. 52, no. 5, pp. 545-557, May 2003.

[77] L. Wang and S. S. Kulkarni. "Gappa: Gossip based multi-channel repro- gramming for sensor networks". In *Second IEEE International Conference in Distributed Computing in Sensor Systems* (DCOSS) 2006.

[78] E. W. Weisstein, "Birthday Attack", From *MathWorld–A Wolfram Web Resource*. http://mathworld.wolfram.com/BirthdayAttack.html

[79] E. W. Weisstein, "Hypergeometric Function", From *MathWorld–A Wolfram Web Resource*. http://mathworld.wolfram.com/HypergeometricFunction.html

[80] G. Yan, T. Zhou, J. Wang, et al. "Epidemic spread in weighted scale-free networks". In *Chin Phys Lett*, Vol 22, pp 510-513, 2005.

[81] F. Ye, H. Luo, J. Cheng, S. Lu, L. Zhang, "A Two-tier data dissemination model for large-scale wireless sensor networks", In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking* (MOBICOM), 2002.

[82] Z. Yu and Y. Guan. "A key pre-distribution scheme using deployment knowledge for wireless sensor networks", In *Proceedings of ACM/IEEE International Conference on Information Processing in Sensor Networks* (IPSN), 2005.

[83] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks," In *Workshop on Parallel and Distributed Simulation* (PADS), 1998.

[84] Q. Zhang and D. P. Agarwal, "Dynamic Probabilistic Broadcasting in MANETs", In *Journal of Parallel and Distributed Computing*, vol. 65, no. 2, pp. 220-233, 2005.

[85] C. C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis", In *The 9th ACM conference on Computer and Communications Security*, 2002.

[86] Sun^{TM} Small Programmable Object Technology (Sun SPOT), www.sunspotworld.com.

[87] C. Inc. Cc2420 data sheet. *http://www.chipcon.com/files/CC2420_Data_Sheet_1_0.pdf*, 2003.

[88] IEEE 802.11. http://standards.ieee.org/getieee802/802.11.html.

[89] Institute for Software Integrated Systems at Vanderbilt University. JProwler: (http://www.isis.vanderbilt.edu/projects/nest/jprowler/)

[90] http://www.malware.com

BIOGRAPHICAL STATEMENT

Pradip De received his Bachelor of Technology (Honours) degree in Computer Science and Engineering from Haldia Institute of Technology, Haldia, India, in 2001. He spent a year and a half working as a Networking Software Engineer at Alumnus Software India, a networking software development company based in Calcutta, India. He began his graduate studies in the Department of Computer Science and Engineering at The University of Texas at Arlington in Fall 2002 and received his Master of Science degree in Computer Science and Engineering in August 2004. His research interests include modeling and design of communication protocols for wireless sensor and mesh networks, applications of epidemic theory in modeling information propagation in networked systems, security in wireless sensor networks, designing RFID based pervasive systems, and wireless networks.