An Epidemic Theoretic Framework for Vulnerability Analysis of Broadcast Protocols in Wireless Sensor Networks

Pradip De, Yonghe Liu, and Sajal K. Das

Abstract—While multihop broadcast protocols, such as Trickle, Deluge, and MNP, have gained tremendous popularity as a means for fast and convenient propagation of data/code in large-scale wireless sensor networks, they can, unfortunately, serve as potential platforms for virus spread if the security is breached. To understand the vulnerability of such protocols and design defense mechanisms against piggybacked virus attacks, it is critical to investigate the propagation process of these protocols in terms of their speed and reachability. In this paper, we propose a general framework based on the principles of epidemic theory, for vulnerability analysis of current broadcast protocols in wireless sensor networks. In particular, we develop a common mathematical model for the propagation that incorporates important parameters derived from the communication patterns of the protocol under test. Based on this model, we analyze the propagation rate and the extent of spread of a malware over typical broadcast protocols proposed in the literature. The overall result is an approximate but convenient tool to characterize a broadcast protocol in terms of its vulnerability to malware propagation. We have also performed extensive simulations that have validated our model.

Index Terms—Reprogramming protocols, epidemic theory, analytical model.

1 Introduction

As wireless sensor networks are being widely deployed in a plethora of applications [5], [2], security remains one of the most critical challenges yet to be fully addressed. Recent emergence of viruses like Cabir[1] that spread over air interfaces in cellular networks indicates that wireless sensor networks are also extremely vulnerable to malwares. Although Cabir uses the bluetooth interface to spread between cell phones, it proves that viruses have perpetrated the wireless domain too, and it is a matter of time before we would find a virus that spreads over the 802.15.4 medium, which is now used by most sensor nodes. Furthermore, the unique characteristics of wireless sensor networks further extend this vulnerability. First of all, the high-density deployment of wireless sensors implies that any malware infection can be highly contagious. Second, sensor nodes are severely resource constrained, and hence lack sophisticated defense mechanisms to fight virus attacks. Attacks that might seem apparently simple in a normal network might prove detrimental in the case of a sensor network. For instance, a piece of software that can spread and at the same time performs a complex action that depletes battery can be malicious from the sensor network's standpoint. Moreover, the fact that it would be difficult to continuously monitor a deployed network makes it even more vulnerable.

In recent years, several over-the-air reprogramming protocols such as Trickle [16], Firecracker [19], Deluge [11], or MNP [15] have been proposed in the sensor network literature, to reprogram or reconfigure a sensor network over the air. These protocols are based on epidemic propagation principles. They are extremely useful, when troubleshooting the entire network, as the sensors are generally considered inaccessible after deployment. However, 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 reprogramming protocol's propagation mechanism to transport itself to the entire network.

Therefore, it is critical to understand the behavior of these protocols in the light of malware propagation in a sensor network in order to be able to devise effective security mechanisms in the event of a security breach.

A set of recent works have focused on authentication mechanisms for securing broadcast protocols, particular for code-update [9], [8], [10], using a combination of hash trees or hash-chain-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 [24], [8] with hash trees being generated for each one. Moreover, if the source of a network-wide broadcast is a regular sensor

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

- *P. De is with Sentilla Corporation, 201 Marshall Street, Redwood City, CA 94063. E-mail: pradip.de@gmail.com.*
- *Y. Liu and S.K. Das are with the Department of Computer Science and Engineering, University of Texas at Arlington, 416 Yates Street, Arlington, TX 76019. E-mail: {yonghe, das}@uta.edu.*

Manuscript received 19 Sept. 2007; revised 10 Mar. 2008; accepted 8 July 2008; published online 4 Aug. 2008.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2007-09-0283.
Digital Object Identifier no. 10.1109/TMC.2008.115.

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 malicious code propagation vehicle.

In this paper, we consider such a general scenario for network-wide broadcasting of information and instead of looking into particular cryptographic techniques to secure broadcast protocols, we focus on the understanding and modeling of their working process in terms of data propagation speed and reachability. In particular, we look at a scenario where a source node has been compromised and is being used along with the communication mechanism of the broadcast protocol to compromise the rest of the nodes by propagating a piece of malware over 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 malware over different broadcast protocols (e.g., Trickle, Firecracker, Deluge, and MNP), thus facilitating a comparative analysis of their potential vulnerabilities. The epidemic model for data propagation is constructed based on the local spatial interaction of nodes in a neighborhood. In order to map a specific broadcast protocol to our model, we derive the spreading rate from the general communication pattern of the protocol, incorporating the physical communication constraints of the wireless network. Subsequently, we use this rate in our epidemic model to observe the dynamics of the malware infection spread over the particular protocol. In addition, the model also allows for the study of the effects of simultaneous recovery processes on the infection spread effected by the broadcast protocol.

It should be emphasized here that our model is very generic and could serve a broader objective of effectively comparing different broadcast-based multihop communication protocols in terms of their speed of propagation and coverage of the network. The flexibility of the model allows different protocols to be plugged into the framework for comparison on a common platform. However, in this paper, we look at a specific problem of characterizing malware spread over broadcast-based dissemination/reprogramming protocols and exposing their vulnerabilities.

Indeed, several protocols and algorithms for data dissemination in sensor networks [13], [1], [16], [11], [14], [17], [18] have been proposed in the literature that are based on the philosophy of epidemic data propagation. These protocols are either meant to flood a piece of information to the entire network or disseminate code for reprogramming all the nodes of the network. Keeping the resource constraints of the sensors in mind, they try to optimize the performance by reducing the number of redundant transmissions at each node as much as possible.

The study of the propagation of worms or viruses in the Internet [26], [27] has also been based on this theory. Most of these models assume network properties that are characteristic of the Internet topology. However, the same epidemic models are unsuitable for use in sensor networks. Several random graph theoretic models [20] exist for the spread of epidemics in a network. The authors in [6] used this model to study the spread of node compromise in a wireless sensor network where communication is secured by a shared secret-key-based mechanism. The malware in a compromised node uses the secret keys shared with its neighbors to securely communicate with them and infect them in the process. However, their analytical model fails to capture the temporal dynamics of the compromise propagation and only succeeds in capturing the final outcome of the infection. In [12], the authors propose a topology-aware temporal and spatial worm propagation model in sensor networks. Although they present a closed form solution for computing the infected fraction of the network, their model assumes a structured grid topology and also does not consider the simultaneous effects of any recovery process on the infection spread. Moreover, it is difficult to use the model to represent different broadcast protocols and study their epidemic characteristics against each other.

Therefore, our main difference from existing works lies in two aspects: 1) The flexibility by which different broadcast protocols can be plugged into the model to capture an approximate behavior of their propagation speed and reachability into the network and 2) the incorporation of a simultaneous recovery process and its effect on the infection propagation dynamics.

A preliminary version of this work has been accepted for publication in [7].

The remainder of this paper is organized as follows: In Section 2, we present some preliminary discussions relevant to the work in this paper. In Section 3, we present our proposed epidemic modeling framework with the detailed analysis. In Section 4, we map three broadcast protocols onto our epidemic model. In Section 5, we present the experimental details, and conclude this paper in Section 6.

2 Preliminaries

In this section, we present some preliminaries including network topology model, basic concepts of epidemiology, and briefly overview the broadcast protocols we will later analyze.

2.1 Sensor Network Model

We model a wireless sensor network as an undirected geometric random graph $G_{p(d_{uv})}(N)$ [21] of N nodes, 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

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

Correspondingly, the mean degree η of a node is defined as

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

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 [23], the assumption is that the logarithm of $\mathbf{P}(r)$ is normally distributed.

2.2 Basics on Epidemic Theory

Epidemic theory [3] aims to measure the infection outcomes in relation to a *population at risk* comprising 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 and so on, which define the potential of the disease causing the infection. Various models exist in epidemic theory that characterize an infection spread, such as Susceptible-Infected-Susceptible (S-I-S) model and Susceptible-Infected-Recovered (S-I-R) model. In the former model, a susceptible individual acquires infection and then after an incubation period (i.e., the time the infection persists), the individual becomes susceptible again. On the other hand, in the latter model which is more generic, the individual recovers and becomes immune to further infections.

In the general deterministic S-I-R model, $S(t)$, $I(t)$, and $R(t)$ denote the number of susceptible, infected, and recovered (or immunized) individuals at time t, respectively. If β denotes the infection rate and γ denotes the removal rate of infected individuals, then assuming a homogeneous mixing model (i.e., each of the susceptibles can get in contact with any of the infectives), the basic differential equations that describe the rate of change of susceptibles, infectives, and recovered individuals are given by

$$\frac{dS(t)}{dt} = -\beta S(t)I(t), \tag{3}$$

$$\frac{dI(t)}{dt} = \beta S(t)I(t) - \gamma I(t), \tag{4}$$

$$\frac{dR(t)}{dt} = \gamma I(t). \tag{5}$$

The above equations can be solved either approximately or precisely based on certain boundary conditions.

2.3 Broadcast Protocols

In this section, we provide an overview of three broadcast protocols proposed for data dissemination in sensor networks. We will later analyze their properties, and how, if compromised, they can be employed for malware propagation.

2.3.1 Trickle

Trickle [16] is an algorithm for propagating and maintaining code updates in wireless sensor networks. The basic approach of Trickle is based on a "*polite gossip*" policy where nodes periodically broadcast code summary advertisements to local neighbors but remain silent if they have recently heard a summary identical to theirs. 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. Thus, a node will gradually reduce its gossip rate if it does not hear new information. When it indeed overhears any new gossip, the rate will be increased.

For further details, the reader is referred to the original paper [16].

2.3.2 Firecracker

In [19], the authors have introduced a technique which is a combination of both routing and broadcasts in order to rapidly deliver a piece of data to every node in a network. To start the dissemination process, the data source first routes data to certain distant points termed seed points in the network. Once the data reaches the respective destinations, broadcast-based dissemination begins along the path like a string of firecrackers. Nodes that fall along the routes chosen by the routing protocol store the data. Once a node has received the data, it then uses a broadcast-based local dissemination protocol, such as Trickle [16], to spread the data.

2.3.3 Deluge

Deluge [11] is a reliable data dissemination protocol for propagating large data objects from one or more source nodes to many other nodes. It is based on prior work on density-aware, maintenance protocols. Deluge operates as a state-machine where each node follows a set of strictly local rules to achieve a desired global behavior. In order to accommodate the large size of a data object, Deluge divides it into fixed size *pages* which is a basic unit for data transferring. A page is composed of a fixed number of P packets. Thus, Deluge uses the reliability of gossiping in Trickle while increasing the data propagation speed by pipelining the pages through the network.

Specifically, a node operates in one of the three states at any time: *MAINTAIN*, *RX*, or *TX*. In the *MAINTAIN* state, a node is responsible for ensuring that all nodes within its communication range have all the available data of the newest version. In this state, the node could essentially use the same mechanism as Trickle [16] to control the transmission of redundant messages. Also, nodes advertise a summary of their data locally. In the *RX* state, a node sends out requests to those nodes from whom it has previously heard advertisements about newer pages. In the *TX* state, a node transmits page updates after receiving requests from nodes that contain older versions of the pages of the data object.

2.3.4 MNP

MNP [15] is another broadcast-based multihop reprogramming protocol for sensor networks. It, in particular, addresses the issue of message collisions and hidden terminal problems in previous reprogramming protocols like Deluge. It incorporates several functionalities of Deluge for code dissemination, such as periodic advertisement of metadata. 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. An advertising node does not service requests from other nodes immediately but waits until it has transmitted a threshold number of advertisements to gather the requests from other nodes. During this time, nodes in a neighborhood decide who should serve as the sender. Moreover, MNP also propagates the entire code image over each hop before propagating it across the next hop. MNP saves energy by reducing the active radio time of a sensor node by forcing it into sleep state when its neighbors are transmitting a segment which is not of interest.

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 not only propagates large data chunks like Deluge but also employs a sender selection mechanism to enforce a single data source in a neighborhood.

2.4 Attack Model

In this section, 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. First, 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.

Second, 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 [8], [9]. 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 [28] as being a piece of malicious software which could include computer viruses, worms, trojan horses, and spywares. 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.

3 An Analytical Framework Based on Epidemic Theory

In this section, we propose a novel framework based on epidemic theory for analyzing the propagation of malware 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 basic idea is to model the communication between a compromised and a noncompromised node as a contact between an infected individual and a susceptible one. Moreover, just as a susceptible individual might get infected with a certain probability once it is in contact with an infective one, a noncompromised node would be compromised if it receives any malware through a communication with a compromised node.

TABLE 1
Notation Summary

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

3.1 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 compromised by a malware spreading over the broadcast protocol. Likewise, $S(t)$ and $R(t)$, respectively, denote the set of susceptible and 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: 1) probability ρ representing the infectivity of the malware which is a measure of how contagious it is and 2) 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 affected either by injection of an antivirus to disinfect the virus-infected nodes or revoking the compromised nodes. Table 1 provides a summary of the parameters used in the model.

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, as a stealth technique, uses the normal operation of a broadcast protocol to infect

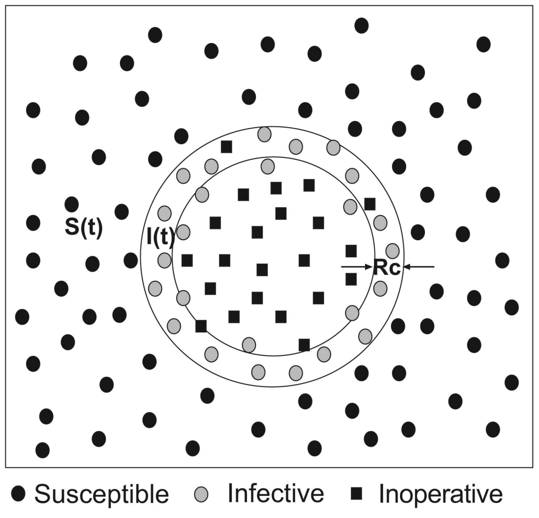


Fig. 1. The spreading phenomenon in a sensor field.

its neighbors. Thus, the infection rate is dependent on the communication rate of the broadcast protocol.

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 wave front 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. 1. We consider two situations in our model: 1) where nodes are *not* recovered once they are infected and 2) where nodes can be recovered from the infection. We derive analytical expressions for each subpopulation function $S(t)$, $I(t)$, and $R(t)$ for both cases.

3.2 Model Analysis

In this section, we present a detailed analysis of the propagation mechanism by deriving the functions describing the dynamics of each subpopulation 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 1. *Given that sensor nodes are uniformly randomly distributed in a field, the numbers 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 $O(\sqrt{N})$. In other words, we can claim that if nodes are uniformly randomly distributed in a unit square such that the average distance between any two nodes is $O(N)$, then there are $O(N^2)$ nodes present in the unit square. In a similar manner, 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. □

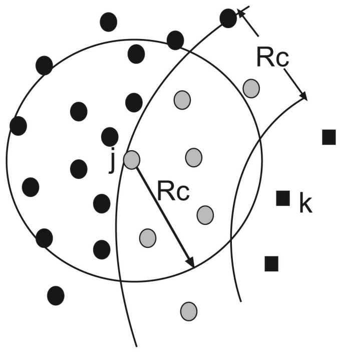


Fig. 2. Circular strip of thickness R_c. Only a fraction of neighboring nodes of an infected node is the potentially susceptible ones.

We now present our model separately for the cases without node recovery and with node recovery in the network.

3.2.1 No Recovery

In this case, we assume that the nodes, once infected, are compromised and cannot be recovered. Thus, $R(t) = 0$ and $\gamma = 0$ in (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 nonhomogeneous 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. 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

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

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

$$I'(t) = (2\sqrt{\sigma\pi}R_c)\sqrt{I(t)} - \sigma\pi R_c^2. \tag{6}$$

We observe that I' is of the order of $O(\sqrt{I})$ as also proven in 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. 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 malware 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

$$N(t) = S(t) + I(t) \tag{7}$$

and the differential equations as

$$\frac{dI}{dt} = \beta c \sqrt{I} \frac{(N-I)}{N} \eta, \tag{8}$$

$$\frac{dS}{dt} = -\beta c \sqrt{I} \frac{(N-I)}{N} \eta. \tag{9}$$

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

$$\frac{dU}{U^2 - 1/N} = -\frac{\beta c \eta}{2} dt, \tag{10}$$

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:

$$I(t) = N \left(\frac{2}{1 + \left(\frac{\sqrt{N}-1}{\sqrt{N}+1} \right) e^{-\frac{\beta c \eta}{\sqrt{N}} t}} - 1 \right)^2. \tag{11}$$

Note that in the above equation when $t = \infty$, $I(t) = N$, i.e., asymptotically all the nodes will be compromised. Equation (11) basically 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.

Apart from the study of malware spread, which is the main focus of this paper, our nonrecovery model would also serve as a tool to analyze general information propagation in wireless sensor networks under different conditions of network connectivity. Moreover, as we would see in subsequent sections, incorporation of the communication rate of different broadcast protocols into our model would permit a comparative analysis of their data propagation potential against each other.

3.2.2 With Recovery

In this case, 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. 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.

Moreover, we have also assumed that it is only an infected node that can be recovered. Thus, our model does not encompass the cases where susceptible nodes are made immune before they are infected.

Similar to the previous nonrecovery case, 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:

$$N(t) = S(t) + I(t) + R(t). \tag{12}$$

Similar to (6), 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 nonrecovery case, each infected node interacts with the susceptible fraction of its neighbors. Therefore, the nonlinear ordinary differential equations describing the process can be defined as

$$\frac{dI}{dt} = \beta c \sqrt{N - S} \frac{S}{N} \eta - \gamma I, \tag{13}$$

$$\frac{dS}{dt} = -\beta c \sqrt{N - S} \frac{S}{N} \eta, \tag{14}$$

$$\frac{dR}{dt} = \gamma I. \tag{15}$$

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

$$S(t) = N - N \left(\frac{2}{1 + \left(\frac{\sqrt{N}-1}{\sqrt{N}+1} \right) e^{-\frac{\beta c \eta}{\sqrt{N}} t}} - 1 \right)^2. \tag{16}$$

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

$$\frac{dI}{dt} = \frac{\beta c \eta}{\sqrt{N}} \left[\left\{ N - N \left(\frac{2}{1 + C_1 e^{C_2 t}} - 1 \right)^2 \right\} \left(\frac{2}{1 + C_1 e^{C_2 t}} - 1 \right) \right] - \gamma I. \tag{17}$$

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

$$\frac{d(I e^{\gamma t})}{dt} = -C_2 N \left[\left\{ 1 - \left(\frac{2}{1 + C_1 e^{C_2 t}} - 1 \right)^2 \right\} \left(\frac{2}{1 + C_1 e^{C_2 t}} - 1 \right) \right] e^{\gamma I}. \tag{18}$$

We use the Gaussian Hypergeometric Function *Hy2F1* [25],[2] to solve the above equation and obtain a closed form expression for $I(t)$. As with the nonrecovery case, we use the same boundary condition, i.e., at $t = 0$, $I(t) = 1$:

$$\begin{aligned} I(t) = {} & \frac{4C_1C_2Ne^{C_2t}}{(C_2+\gamma)(C_2+C_1C_2e^{C_2t})^2}[A] + e^{-C_2t} \\ & - \frac{4C_1C_2Ne^{-\gamma t}}{(C_2+\gamma)(C_2+C_1C_2)^2}[B], \end{aligned} \tag{19}$$

where

$$\begin{aligned} A = {} & (C_2+\gamma)(-C_2+\gamma+\gamma C_1e^{C_2t}) - (\gamma+\gamma C_1e^{C_2t})^2 \\ & Hy2F1(1, 1+\gamma/C_2, 2+\gamma/C_2, -C_1e^{C_2t}), \\ B = {} & (C_2+\gamma)(-C_2+\gamma+\gamma C_1) - (\gamma+\gamma C_1)^2 \\ & Hy2F1(1, 1+\gamma/C_2, 2+\gamma/C_2, -C_1). \end{aligned}$$

Equation (19) 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 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.

In our analytical study, we have made certain assumptions which 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.
- 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.
- The recovery scheme assumes that it is only possible to recover an infected node. In other words, it does not incorporate immunization of susceptible nodes prior to infection.

4 Analysis of Individual Broadcast Protocols

Given the above framework, in this section, we address each of the three broadcast protocols, Trickle, Firecracker, and Deluge. 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.

4.1 Trickle Protocol

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

Proof. If the packet loss rate is denoted by l, then the expected number of transmissions for a given packet is $\frac{1}{1-l}$. 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 $E[T_{adv}] = \frac{T_p}{2} \cdot \sum_{i=1}^{k} \binom{\eta+1}{i} p_k^i (1-p_k)^{\eta+1-i} \cdot \frac{1}{1-l}$. □

In Trickle, the moment a 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

$$E[T_{tx}] = T_{pkt} \cdot \frac{1}{1-l}. \tag{20}$$

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

$$E[T_{CU}] = E[T_{adv}] + E[T_{tx}]. \tag{21}$$

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 a malware. This is the inherent characteristic of the malware, its type and what mechanism it adopts to spread. We term this factor to be ρ, the infectivity of the malware. This parameter ρ differentiates one malware from another. Incorporating this factor into our model, the infection spread rate over Trickle protocol is given by

$$\beta = \frac{\rho}{E[T_{CU}]}. \tag{22}$$

4.2 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

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

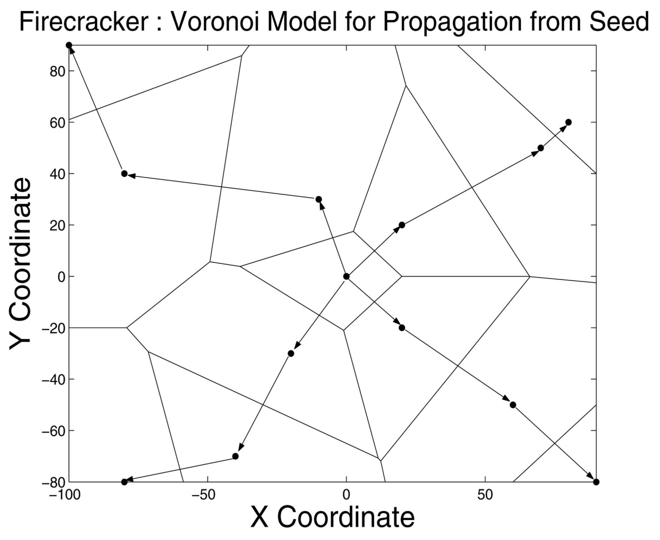


Fig. 3. Firecracker protocol model: voronoi partitioning.

making each of the end nodes of the routing phase (seeds) to be a source of a subinfection 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 subprocesses work in the same manner as the broadcast mechanism in Trickle. The population of nodes for each of these subprocesses 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. 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 ith 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

$$f(t) = \frac{\left|\bigcup_i^S v_i(t)\right|}{N}, \tag{23}$$

where $N = \sum_i^S n_i$.

4.3 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 parameters 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 the packet loss rate as l, the expected number of transmissions for a packet is given by $E[N_{pkt}] = \frac{1}{1-l}$. 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

$$E[T_{adv}] = \frac{T_p}{2} \cdot \sum_{i=1}^{k} \binom{\eta+1}{i} p_k^i (1-p_k)^{\eta+1-i} \cdot \frac{1}{1-l}. \tag{24}$$

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

$$E[T_{req}] = \frac{T_p}{2} \cdot E[N_{pkt}] \cdot E[N_{req}]. \tag{25}$$

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

$$E[T_{tx}] = P \cdot E[N_{pkt}] \cdot T_{pkt}, \tag{26}$$

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

$$E[T_{fallback}] = \left\lfloor \frac{E[N_{req}]}{\lambda} \right\rfloor \cdot E[T_{adv}]. \tag{27}$$

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

$$E[T_{page}] = E[T_{adv}] + E[T_{req}] + E[T_{tx}] + E[T_{fallback}]. \tag{28}$$

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 $\beta_D = \frac{\rho}{E[T_{page}]}$.

4.4 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. Along with 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 analysis, we formulate the expression of the average page transfer rate from a source node to recipients in a neighborhood. We argue that the malware uses the data propagation 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 of 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 interarrival 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, 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 interarrival time of successfully transmitted advertisement messages is given by

$$T^i_{adv} = \frac{1}{\delta} + E[N_{pkt}] \cdot T_{pkt}. \tag{29}$$

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

$$E_\kappa[T_{adv}] = \frac{\kappa}{\delta_{eff}}. \tag{30}$$

We are analyzing the propagation rate that MNP offers to a malware 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 $E[N_{req}]$ number of 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

$$E[T_{pg}] = E_\kappa[T_{adv}] + E[N_{pkt}] \cdot T_{pkt}(P + 2). \tag{31}$$

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

4.5 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/second with a packet size of 32 bytes. This results in a packet transmission time of 0.027 second. 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, $l = 0.1$ and $T_{pkt} = 0.027\ \text{second}$. Simulation results of Deluge [11] have shown that the average number of requests for acquiring a page $E[N_{req}]$ is approximately equal to 5.4.

Figs. 4 and 5, respectively, illustrate the analytical plots depicting the propagation dynamics for each protocol in a network of 1,000 nodes, both for the nonrecovery case and the case where nodes have an expected recovery time. Figs. 4a and 4b show the dynamics of the malware 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 malware, increases the speed of infection by more than 20 percent. The next two subfigures 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 increasing the number of strategically placed source nodes to spread the infection has a significant impact on the subsequent rate of spread of the malware. 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. 4e and 4f for Deluge and Figs. 4g and 4h 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. From a vulnerability standpoint, we can say that this feature also 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.

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 malware 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

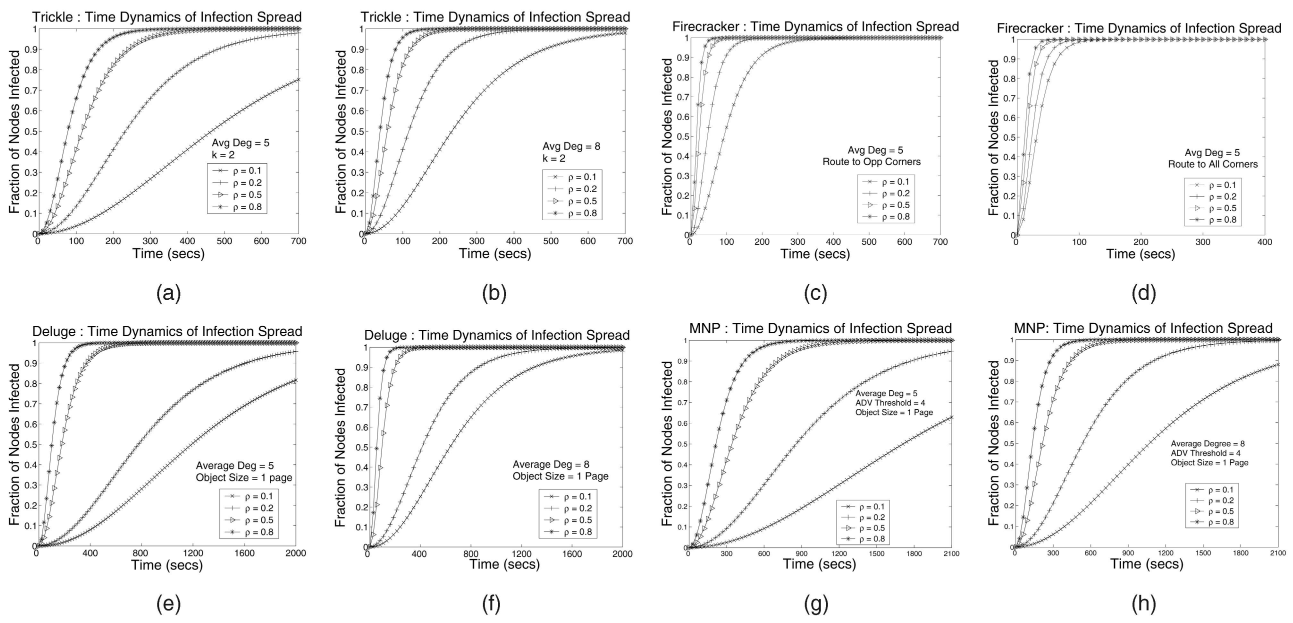


Fig. 4. Without recovery. Growth of infected nodes $I(t)$ with time for different values of ρ (malware infectivity) and average network degree for different broadcast dissemination protocols. (a) For $\mathrm{average\ degree} = 5$. (b) For $\mathrm{average\ degree} = 8$. (c) For $\mathrm{average\ degree} = 5$, Route to Opp Corners. (d) For $\mathrm{average\ degree} = 5$, Route to All Corners. (e) For $\mathrm{average\ degree} = 5$. (f) For $\mathrm{average\ degree} = 8$. (g) For $\mathrm{average\ degree} = 5$. (h) For $\mathrm{average\ degree} = 8$.

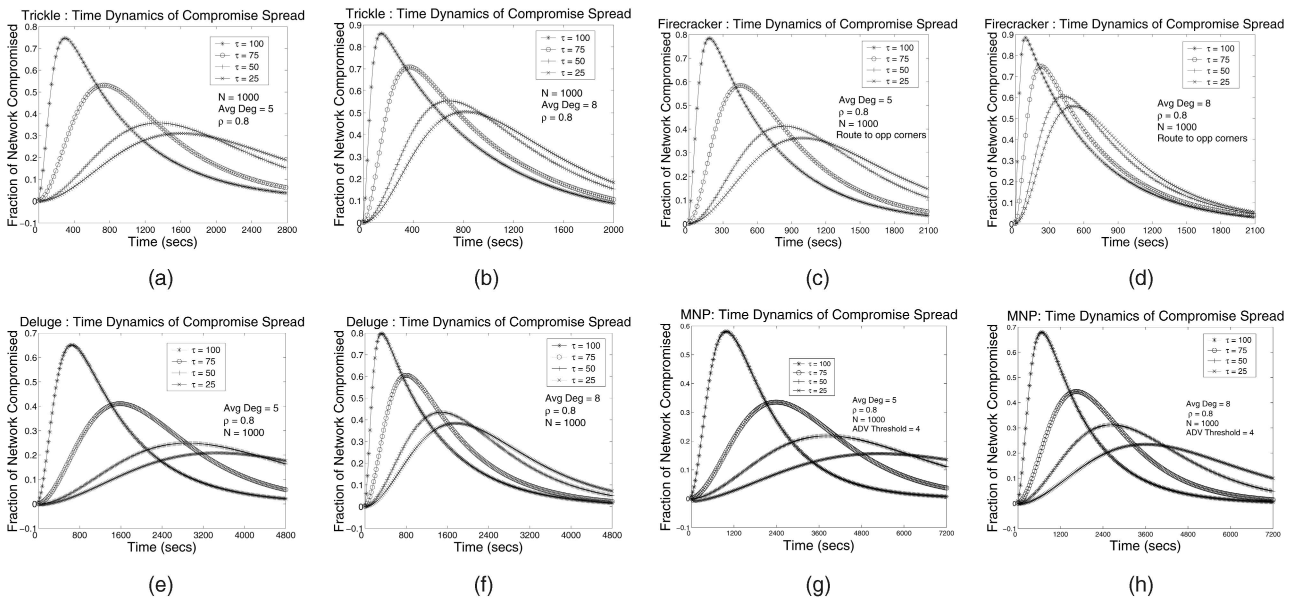


Fig. 5. With recovery. Time dynamics of infected nodes $I(t)$ for different values of τ (average infectivity duration) and average network degree for different broadcast dissemination protocols. (a) For $\mathrm{average\ degree} = 5$. (b) For $\mathrm{average\ degree} = 8$. (c) For $\mathrm{average\ degree} = 5$. (d) For $\mathrm{average\ degree} = 8$. (e) For $\mathrm{average\ degree} = 5$. (f) For $\mathrm{average\ degree} = 8$. (g) For $\mathrm{average\ degree} = 5$. (h) For $\mathrm{average\ degree} = 8$.

discussed and, thus, poses a very high threat for malware transfer in the event of an outbreak. Comparing Figs. 4a and 4c, we observe that even with a lower node degree, Firecracker, with route points as opposite corners, achieves network compromise in about 25 percent to 40 percent of the time as Trickle. However, comparing Figs. 4e and 4g, we observe that MNP with an advertisement threshold of 4 takes almost 70 percent more time than taken by Deluge.

Fig. 5 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 subfigures in Fig. 5, 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. 5f and 4f, 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 nonrecovered case, while it reaches a value close to 450 seconds with recovery. Similarly in MNP, comparison of Figs. 5g and 4g 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.

We, thus, observe that our model can successfully capture the propagation process and shed significant light on the temporal dynamics of how a malware 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 become 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.

Moreover, in our analytical formulations, we have assumed that there is no channel contention delay. This delay gets reflected in our simulations which probably accounts for the slight discrepancy between the simulation and analytical curves.

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

5 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 malware spread is captured with varying degrees of infectivity on the whole network. We have used JProwler [4], a probabilistic, event-driven wireless network simulator in Java, for our experiments.

In the simulation experiments, we assume $N = 1{,}000$ sensor nodes with uniform random deployment in the network. We have used the Gaussian radio model in JProwler. We assume that the link qualities are high enough to guarantee very high packet reception rates and, thus, 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 [22]. 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 20 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 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.

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. 6 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. 4 and 6. 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.

The simulation results for recovery are depicted in Fig. 7. With predefined infection rates derived from the broadcast protocols, we have simulated a simultaneous recovery process given that an infection spread is active.

6 Conclusion

Broadcast protocols in sensor networks are vulnerable as potential carriers of malwares/viruses that spread over air interfaces. In this paper, we have provided a common

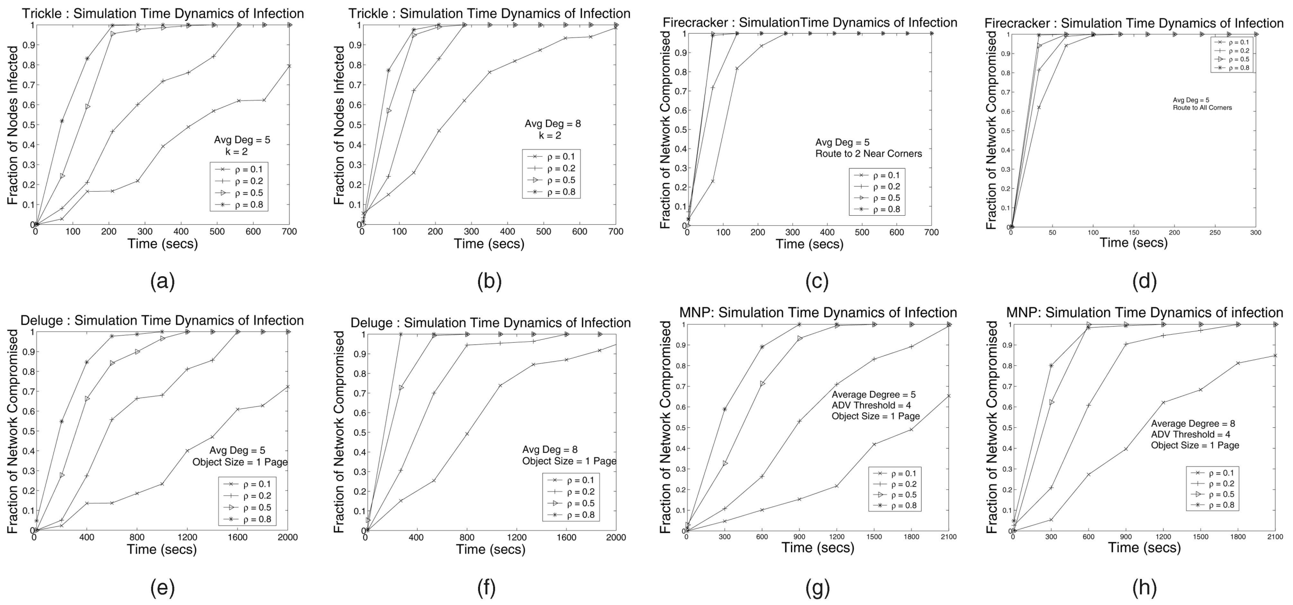


Fig. 6. Simulation without recovery. Growth of infected nodes $I(t)$ with time for different values of ρ (malware infectivity) and average network degree for different broadcast dissemination protocols. (a) For $\text{average degree} = 5$. (b) For $\text{average degree} = 8$. (c) For $\text{average degree} = 5$, Route to Opp Corners. (d) For $\text{average degree} = 5$, Route to All Corners. (d) For $\text{average degree} = 5$. (e) For $\text{average degree} = 8$. (f) For $\text{average degree} = 5$. (g) For $\text{average degree} = 8$.

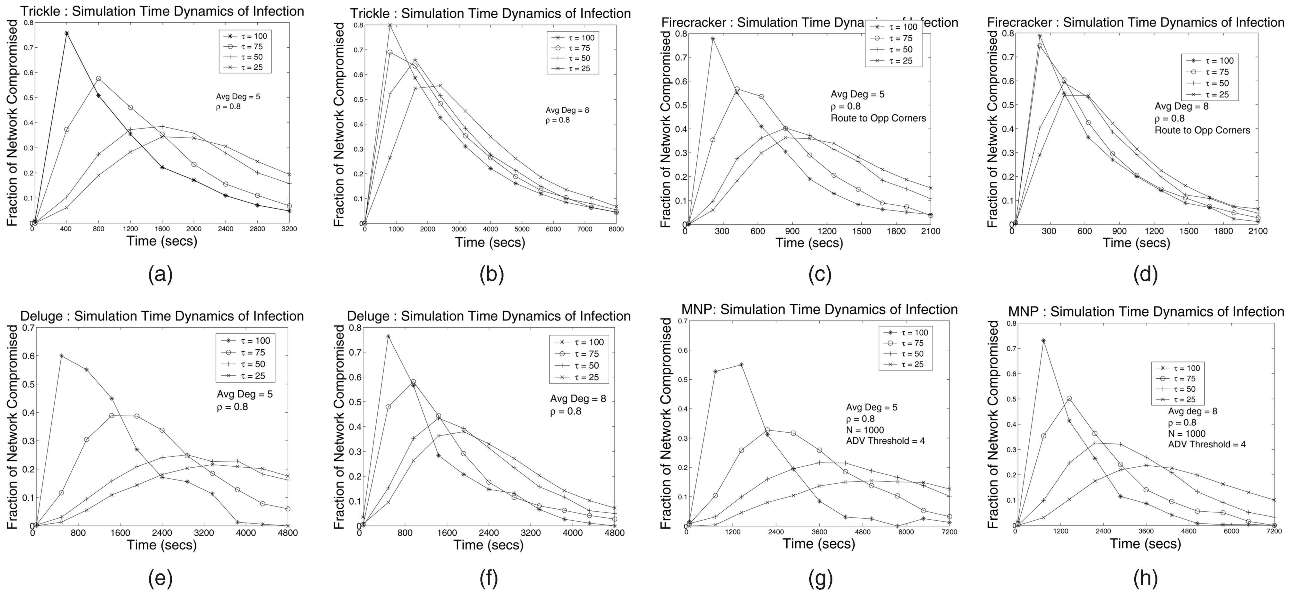


Fig. 7. Simulation with recovery. Time dynamics of infected nodes with time for different values of τ (average infectivity duration) and average network degree for different broadcast dissemination protocols. (a) For $\text{average degree} = 5$. (b) For $\text{average degree} = 8$. (c) For $\text{average degree} = 5$. (d) For $\text{average degree} = 8$. (e) For $\text{average degree} = 5$. (f) For $\text{average degree} = 8$. (g) For $\text{average degree} = 5$. (h) For $\text{average degree} = 8$.

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 wave front 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. A point worth noting is that, although, in this paper, we have focused specifically on the issue of malware spreading over broadcast protocols and their temporal dynamics, our model is very generic and could serve as a tool to compare the general performance of different protocols in terms of speed of information dissemination and network coverage under different states of network connectivity.

As part of our future work, we would like to extend our model to incorporate more realistic deployment strategies like group-based deployment, etc.

ACKNOWLEDGMENTS

This work was supported by the US National Science Foundation ITR Grant IIS-0326505 and Texas ARP Grant 14-748779.

REFERENCES

[1] M. Akdere, C.C. Bilgin, O. Gerdaneri, I. Korpeoglu, O. Ulusoy, and U. Cetintemel, "A Comparison of Epidemic Algorithms in Wireless Sensor Networks," *Computer Comm.*, 2006.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Comm. Magazine,* vol. 40, no. 8, 2002.

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

[4] Institute for Software Integrated Systems at Vanderbilt University JProwler, http://www.isis.vanderbilt.edu/projects/nest/jprowler/, 2008.

[5] C. Chong and S. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proc. IEEE,* vol. 91, no. 8, 2003.

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

[7] P. De, Y. Liu, and S.K. Das, "An Epidemic Theoretic Framework for Evaluating Broadcast Protocols in Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems,* 2007.

[8] J. Deng, R. Han, and S. Mishra, "Secure Code Distribution in Dynamically Programmable Wireless Sensor Networks," *Proc. Fifth Int'l Conf. Information Processing in Sensor Networks,* 2006.

[9] P.K. Dutta, J.W. Hui, D.C. Chu, and D.E. Culler, "Securing the Deluge Network Programming System," *Proc. Fifth Int'l Conf. Information Processing in Sensor Networks,* 2006.

[10] P.E. Lanigan, R. Gandhi, and P. Narasimhan, "Sluice: Secure Dissemination of Code Updates in Sensor Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems,* 2006.

[11] J.W. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," *Proc. Second ACM Conf. Embedded Networked Sensor Systems,* 2004.

[12] S.A. Khayam and H. Radha, "A Topologically-Aware Worm Propagation Model for Wireless Sensor Networks," *Proc. IEEE ICDCS Int'l Workshop Security in Distributed Computing Systems (SDCS),* 2005.

[13] J. Kulik, W. Rabiner, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *Proc. Fifth ACM/IEEE MobiCom Conf.,* 1999.

[14] P. Kyasanur, R.R. Choudhury, and I. Gupta, "Smart Gossip: Infusing Adaptivity in Gossiping Protocols for Wireless Sensor Networks," *Proc. Third IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems (MASS),* 2006.

[15] S.S. Kulkarni and L. Wang, "MNP: Multihop Network Reprogramming Service for Sensor Networks," *Proc. 25th IEEE Int'l Conf. Distributed Computing Systems (ICDCS),* 2005.

[16] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle: A Self-Regulating Algorithm for Code Maintenance and Propagation in Wireless Sensor Networks," *Proc. First USENIX/ACM Symp. Network Systems Design and Implementation (NSDI),* 2004.

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

[18] V. Naik, A. Arora, P. Sinha, and H. Zhang, "Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Wireless Embedded Devices," *Proc. 26th IEEE Int'l Real-Time Systems Symp. (RTSS '05),* Dec. 2005.

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

[20] M.E.J. Newman, "Spread of Epidemic Disease on Networks," *Physical Rev. E,* vol. 66, 016128, 2002.

[21] M.D. Penrose, "Random Geometric Graphs," *Oxford Studies in Probability,* vol. 5, Oxford University Press, May 2003.

[22] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," *Proc. Second ACM Conf. Embedded Networked Sensor Systems (SenSys),* 2004.

[23] T.S. Rappaport, *Wireless Communications: Principles and Practice,* second ed. Prentice-Hall, Dec. 2001.

[24] E.W. Weisstein, *Birthday Attack,* From MathWorld—A Wolfram Web Resource, http://mathworld.wolfram.com/BirthdayAttack.html, 2008.

[25] E.W. Weisstein, *Hypergeometric Function,* From MathWorld—A Wolfram Web Resource, http://mathworld.wolfram.com/HypergeometricFunction.html, 2008.

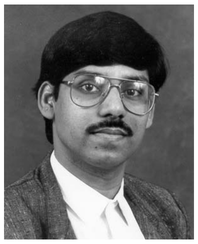
[26] C.C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," *Proc. Ninth ACM Conf. Computer and Comm. Security,* 2002.

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

[28] http://www.malware.com, 2008.

Pradip De received the MS and PhD degrees from the University of Texas at Arlington in 2004 and 2008, respectively. He is a member of technical staff at Sentilla Corp. His research interests include wireless sensor networks, communication protocol design, pervasive computing systems and applications, and RFID systems.

Yonghe Liu received the BS and MS degrees from Tsinghua University in 1998 and 1999, respectively, and the PhD degree from Rice University in 2004. He is an assistant professor in the Department of Computer Science and Engineering, University of Texas, Arlington. His research interests include wireless networking, sensor networks, security, and system integration.



Sajal K. Das is a university distinguished scholar professor in the Department of Computer Science and Engineering and the founding director of the Center for Research in Wireless Mobility and Networking (CReWMaN), University of Texas, Arlington (UTA). He is also a visiting professor at the Indian Institute of Technology (IIT) at Kanpur and IIT Guwahati, an honorary professor of Fudan University in Shanghai, an international advisory professor of Beijing Jiaotong University, and a visiting scientist at the Institute of Infocomm Research (I2R), Singapore. His current research interests include wireless sensor networks, mobile and pervasive computing, design and modeling of smart environments, pervasive security, smart health care, resource and mobility management in wireless networks, mobile grid computing, biological networking, applied graph theory, and game theory. He has published more than 400 papers and over 35 invited book chapters in these areas. He holds five US patents in wireless networks and mobile Internet, and coauthored the books *Smart Environments: Technology, Protocols, and Applications* (Wiley, 2005) and *Mobile Agents in Distributed Computing and Networking* (Wiley, 2008). He is a recipient of several Best Paper Awards in such conferences as EWSN '08, IEEE PerCom '06, and ACM MobiCom '99. He is also a recipient of the IEEE Engineer of the Year Award (2007), UTA Academy of Distinguished Scholars Award (2006), University Award for Distinguished Record of Research (2005), College of Engineering Research Excellence Award (2003), and Outstanding Faculty Research Award in Computer Science (2001 and 2003). He serves as the founding editor-in-chief of *Pervasive and Mobile Computing* (PMC) journal, and an associate editor of the *IEEE Transactions on Mobile Computing*, *ACM/Springer Wireless Networks*, the *IEEE Transactions on Parallel and Distributed Systems*, and the *Journal of Peer-to-Peer Networking*. He is the founder of the IEEE WoWMoM and a cofounder of the IEEE PerCom Conference. He has served as the general or technical program chair as well as a TPC member of numerous IEEE and ACM conferences. He serves on the IEEE TCCC and TCPP Executive Committees.