Ad Hoc Networks 24 (2015) 57–72

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Taming epidemic outbreaks in mobile adhoc networks ☆

E. Hoque [a,*], R. Potharaju [a,*], C. Nita-Rotaru [a], S. Sarkar [b], S.S. Venkatesh [b]

[a] Department of Computer Science, Purdue University, West Lafayette, IN, USA
[b] Department of Electrical Engineering, University of Pennsylvania, Philadelphia, PA, USA

ARTICLE INFO

Article history:
Received 25 June 2013
Received in revised form 6 May 2014
Accepted 28 July 2014
Available online 19 August 2014

Keywords:
Epidemic
Malware
Defense
Mobile adhoc networks

ABSTRACT

The openness of the smartphone operating systems has increased the number of applications developed, but it has also introduced a new propagation vector for mobile malware. We model the propagation of mobile malware among humans carrying smartphones using epidemiology theory and study the problem as a function of the underlying mobility models. We define the optimal approach to heal an infected system with the help of a set of static healers that distribute patches as the T-Cover problem, which is NP-Complete. We then propose three families of healer protocols that allow for a trade-off between the recovery time and the energy consumed for deploying patches. We show through simulations using the NS-3 simulator that despite lacking knowledge of the exact future, our healers obtain a recovery time within a $7.4\times \sim 10\times$ bound of the oracle solution that has knowledge of the future arrival time of all the infected nodes.

1. Introduction

With the advent of Google's Android, the number of wireless devices with complex capabilities and the support to open source operating systems has significantly increased. While the openness of operating systems induces developers' motivation, it also introduces a new propagation vector for mobile malware. Recent reports show a significant increase in malware targeting Android devices [2,3].

Significant research focused on propagation modeling, detection, and application profiling of malware in the context of wired networks [4–8]. Those results do not model mobile malware which spreads directly from device to device by using short-range communication such as WiFi, Bluetooth or NFC [9–12]. Mobile malware propagation has been studied using mean field compartmental models [13] which assume that each infected node will contact every neighbor once within one time step, *i.e.*, the infectivity is equal to the connectivity. Such models do not take into account that mobile malware does not spread at an even contact rate, as spreading requires devices to be within each other's proximity which in turn depends on user mobility. Most previous research on mobile malware has either not considered mobility [14–16] or has given limited considerations to it [17,18]. Approaches that have considered mobility have used popular models like the random waypoint model which, as it has been shown, does not realistically mimic human mobility [19].

While there has been work studying mobile malware propagation, the problem of *infection containment* in wireless networks was less studied. The work of [20] analytically studies containment of infection in a mobile network through countermeasures such as reducing communication range of nodes during an infection outbreak. The work does not consider realistic mobility models and does not propose concrete protocols to deploy and activate such countermeasures. The work in [21]

☆ Parts of this work were presented at the 2012 IEEE Conference on Mobile Adhoc and Sensor Systems (MASS) [1] as a regular paper.

* Corresponding authors.
E-mail addresses: mhoque@cs.purdue.edu (E. Hoque), rpothara@cs.purdue.edu (R. Potharaju), crisn@cs.purdue.edu (C. Nita-Rotaru), swati@seas.upenn.edu (S. Sarkar), venkates@seas.upenn.edu (S.S. Venkatesh).

http://dx.doi.org/10.1016/j.adhoc.2014.07.031
1570-8705/

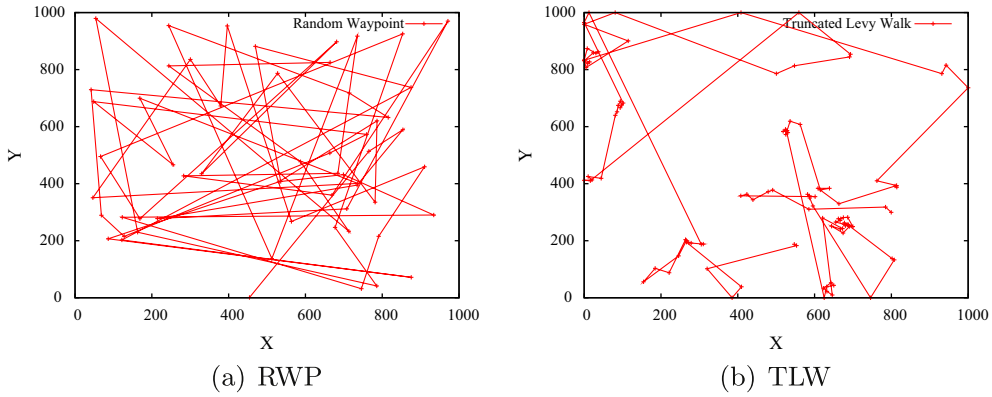


Fig. 1. (a) and (b) Tracing the path of a single node for RWP and TLW: Observe the short paths in RWP and bursty long paths in TLW.

introduces replicative and non-replicative patch disseminations assuming a network cost function and proves that the dynamic control strategies have a simple optimal structure. However, the impractical determination of the healer activation time and the lack of inclusion of the resource cost incurred by each patch dissemination make the techniques difficult to apply directly to energy constrained realistic scenarios.

In this paper, we take the first step towards designing countermeasures for malware propagation under the presence of realistic mobility in a practical scenario. We investigate the dependence of infection spread on the underlying mobility model in order to systematize the design of countermeasures. We introduce the concept of *healers* to mimic the recovery process in a standard epidemic model and we focus on *static healers*, i.e., *immobile* healers, which represent a realistic model because they can be directly mapped to real-world scenarios. For instance, static healers can be considered as cellular base stations (where no two stations cover the same cell in most cases) and the mobile nodes can be considered as users carrying mobile phones (moving with a certain mobility model). Unlike [17], our static-healers are not white-worms and do not deactivate infected nodes. Our contributions are:

- We show that the infection spread in mobility models that mimic human behavior is slower than standard mobility models due to different contact rate and spatial distribution characteristics. We compare the Truncated Levy Walk (TLW) and Random Waypoint (RWP) mobility models and show that the epidemic spread in TLW is *relatively slower* compared to RWP. This finding indicates that when designing countermeasure mechanisms, the time constraints are less tight than believed and that time-dependent assumptions can be relaxed to some extent, resulting in relatively lower consumption of energy.
- We model countermeasures to malware spread using static healer nodes. Static healers once placed in the area, act *independently* to deploy a patch when they sense nodes in their proximity. A healer-based solution optimizes: (i) the time it takes to heal the entire system by patching all the infected nodes and (ii) the total number of patches broadcasted. We formulate the optimal solution based on static healers as a T-Cover problem, which is NP-Complete.
- We use ORACLE, a $log(n)$ greedy approximation algorithm, that computes the optimal healing time knowing the placement of the static healers and *the future*, i.e. the exact time instances when the infected nodes arrive within each healer's proximity.
- We propose a novel healer placement strategy using blue-noise distribution generating Poisson Disk Sampling. We show that unlike random placement that results in many overlapping healers which cover the same area, our method allows healers to cover disjoint areas, thus enabling them to independently cover more infected nodes.
- We design three families of healer protocols: *randomized* (RH), *profile* (PH), and *prediction* (PDH), that allow for a trade-off between the energy consumed for sending patches and the time taken to recover the entire system. The intuition behind each protocol is as follows: (1) RH uses randomization to ensure simplicity in healers' functionality and achieves reasonable performance. (2) PH uses system feedback to optimize the energy consumed for sending patches, but may result in a larger recovery time. (3) PDH predicts the cost of waiting for a suitable time instance to deploy a patch thereby achieving a smaller recovery time but has the side-effect of utilizing more patches. We compare our protocols with the ORACLE protocol and show through simulations that despite lacking knowledge of the future, our healers obtain a recovery time within a $7.4\times \sim 10\times$ bound of the ORACLE.

The rest of this paper is organized as follows. Section 2 describes our system model, our assumptions and introduces the mobility models used in this paper. Section 3 analyzes the infection dynamics as a function of the underlying mobility models. Section 4 introduces our healer-based protocols and Section 5 provides simulation results. Section 6 discusses related research and Section 7 concludes the paper.

2. System model

In this section, we construct a framework for analyzing the propagation of malware over a mobile ad hoc network that relies on epidemic theory to capture both the spatial interaction of nodes and the temporal dynamics of infection propagation.

2.1. Mobility models

Due to the difficulties in adapting real-trace data to long running simulations [16], we decided to use analytical models derived from real-trace data instead. Specifically, we use the *Random Waypoint* (RWP) and *Truncated Levy walk* (TLW) mobility models to generate synthetic mobility traces. We selected RWP because it is a typical mobility model used to study mobile malware propagation. We selected TLW because it provides more realistic representations of statistical patterns found in human mobility. Unless otherwise noted, we use a node velocity of 0.6 m/s to mimic low velocity realistic human mobility in both mobility models throughout the paper.

Random Waypoint (RWP): RWP is a widely used mobility model [22–24] and includes pause times between changes in direction and/or speed [25]. A mobile node begins by staying in one location for a certain time period (pause time). Once this time elapses, the mobile node chooses a random destination in the simulation area and a speed that is uniformly distributed between $[v_{min}, v_{max}]$. The mobile node then travels toward the newly chosen destination at the selected speed. Upon arrival, the node pauses for a specified time period and starts the whole process again (see Fig. 1(a)). RWP is heavily used for mobile ad hoc network simulation [26] to simulate mobile nodes that can move randomly and freely in a mobility area without any restriction. This model is super-diffusive because of high-probability of long flights. On the contrary, human walks have heavy-tail flight distributions [27] that are not captured by common mobility models such as RWP.

The initial random distribution of mobile nodes is not representative of the manner in which nodes distribute themselves when moving as the instantaneous mobile node neighbor percentage possess high variability [28]. We use the approach suggested by [26] and discard the initial 1000 s of simulation time produced by RWP in each simulation trial.

Truncated Levy Walk (TLW): Based on the empirical studies performed on human mobility data collected through mobile devices carried by humans, Rhee et al. [19] reported that human walks performed in outdoor settings of tens of kilometers resemble a truncated form of Levy walks commonly observed in animals such as spider monkeys, birds and jackals. A *Levy walk* is a type of random walk in which the increments are distributed according to a heavy-tailed probability distribution, *i.e.*, their tails are not exponentially bounded. The distribution used is a power law of the form $y = x^{-\alpha}$ where $1 < \alpha < 3$. TLW is a random equivalent mobility model for human walks in that it can describe some important characteristics of human walks (e.g. flight length, pause time and inter-contact time) despite being a random model. Inter-contact times are defined to be the time durations between two consecutive meeting events of the same two nodes. Human walks have long inter-contact times, which is intuitive in a sense that as humans do not move much, they will not meet each other very often. The distributions of these inter-contact times, which follows a power-law distribution with an exponential tail, are similar to those observed in case of Levy walks. Similarly, the heavy-tail distributions of flight length and pause time can be captured by Levy walkers moving in a confined area. Intuitively, Levy walks consist of many short flights and exceptionally long flights that nullify the effect of such short flights (see Fig. 1(b)).

Note that while there are other recent human mobility models similar to TLW such as the ones proposed by Lee et al. [29], Boldrini et al. [30] and Isaacman et al. [31], our end goal is to advocate the use of one of these human mobility models while designing defenses against epidemic outbreaks.

2.2. Infection and recovery models

We adapt two classic epidemic models (SI and SIR) to take into account mobility. First we give a brief overview of the SI and SIR models, then describe how we use them to model malware propagation and node recovery in a mobile network.

SI Model. The SI-model is a two-state compartmental epidemic model, *i.e.*, a node can stay in one of two states: *susceptible* and *infected*. A susceptible node is vulnerable and can be exploited to be infected which in turn can infect other susceptible nodes. In this model, once a susceptible node is infected, it stays that way. The parameter that characterizes the model is the infection rate, β.

SIR Model. The SIR Kermack–McKendrick model [32] assumes that an infected node can be recovered. Specifically a node can be in one of the following states: *susceptible*, *infected*, and *recovered*. Nodes flow from the susceptible group to the infected group and then to the recovered group [33] as shown in Fig. 2. The model is characterized by two parameters, the infection rate β and the recovery rate α.

Mobile Infection Model. The SI model makes the unrealistic assumption that each infected node will contact every neighbor once within one time step, *i.e.*, the infectivity is equal to the connectivity. To take into account mobility, we assume the nodes are moving according with a mobility model and we define infection spread as a function of a parameter c which we call the *probability of successful*

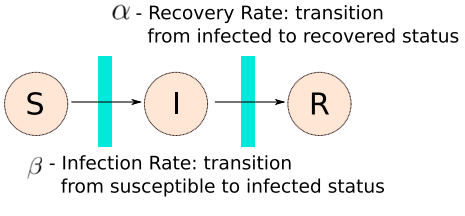


Fig. 2. SIR Model: S, susceptible; I, infected; R, recovered.

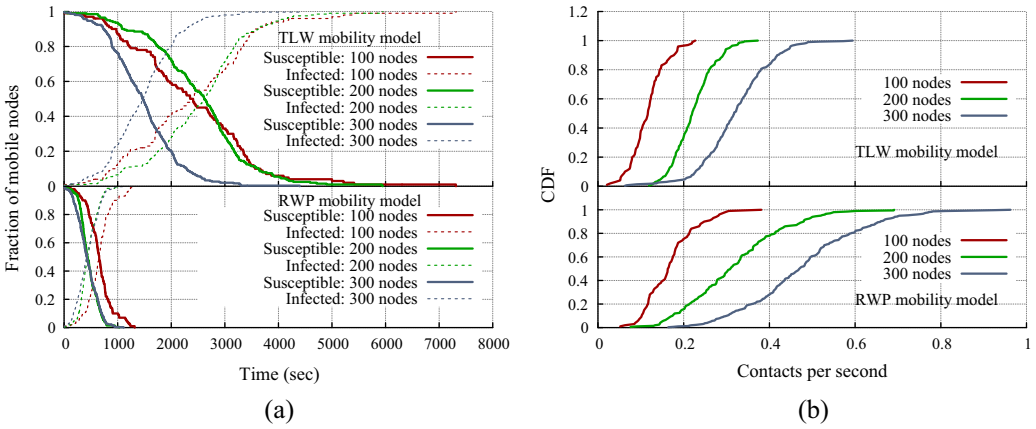


Fig. 3. (a) Inversion point in RWP and TLW: Observe that the infection spread is slower in TLW. (b) Explaining the slow propagation: Observe the contact rate in TLW which is less than RWP.

transmission. At each time step, for every node X, we find the neighbors of X that are capable of infecting X. For each of these neighbors, we generate a random number from a uniform distribution between $[0, 1]$ and if this value is smaller than c, then X becomes infected.

Mobile Recovery Model. We adapt the SIR epidemic model as follows. Infection is modeled as in the mobile infection model above. We map node recovery through a healer that will change the state of an infected node to recovered through a healing mechanism. Once recovered, a node can no longer be infected, thus if no new nodes are added the infection will eventually disappear. The healing mechanism is distributed through a patch, a healer can send at most once during an interval of time called *epoch*, denoted as τ. We assume that healers are static, resource constrained, and act independently. Our assumptions also include that once an infected node receives a patch, the node instantaneously applies the patch and becomes completely recovered. We assume that there is no packet loss but note that it is straightforward to extend our model to a model having packet loss.

This model is characterized by the way the healers are placed and by the frequency with which they send patches. All healers are activated once the number of infected nodes in the system reached a system-wide parameter.

3. Infection dynamics

In order to understand the infection dynamics of the two mobility models, we first describe our methodology and then explain the results that we observed.

3.1. Methodology

We use the infection model described in previous section with the parameter that controls the infection rate, $c = 0.3$ [34] to mimic a more realistic infection scenario where infection spreads slowly. We generate RWP traces by using the methodology outlined in [35] and TLW traces by using the algorithm outlined in [19]. We perform our simulations using NS-3 [36]. We simulate the behavior of a system with 100, 200, and 300 nodes in a fixed area. All results have been averaged over ten simulation runs.

We define an *inversion point* to be the time instant where 50% of the population is infected. We use this metric to indicate the *first* point in time where the number of infected nodes surpasses the susceptible ones, thus *inverting* the scenario. Intuitively, an inversion point characterizes how fast the infection is propagating in an epidemic system.

3.2. Results

Fig. 3 shows the infection dynamics in RWP and TLW mobility models. Observe that the inversion point for RWP occurs quite early in the simulation (Fig. 3(a) indicates a time around 500 s) in comparison with TLW (Fig. 3(a) indicates times between 1500–3000 s). This indicates that the time required to infect the system is far less in case of RWP differing almost by a factor of 3 from TLW. To the best of our knowledge, this phenomenon has not been observed before as most earlier research [17,18] has studied these mobility models in isolation. As protocols are to be designed mostly for realistic mobility models (TLW in this case), this comes as a good news in that certain assumptions such as time-constrained-ness of a protocol can be relaxed to some extent, resulting in relatively lower consumption of energy.

We gain insights into the reasons behind the slow infection propagation for TLW by using two metrics: (i) contact rate and (ii) spatial distributions of node mobility.

1. Contact Rate: Contact rate is the average number of nodes encountered by any given node over the duration of simulation. We plot an empirical cumulative distribution curve (ECDF) of the contact rate in Fig. 3(b) for RWP and TLW. Observe that the median contact rate of nodes in case of RWP is almost always higher than that in TLW. The same effect can be observed for the 95th percentile indicating that in RWP, a given node comes in contact with a relatively higher number of nodes thereby increasing its chances of infecting other nodes or getting infected by other infected nodes.

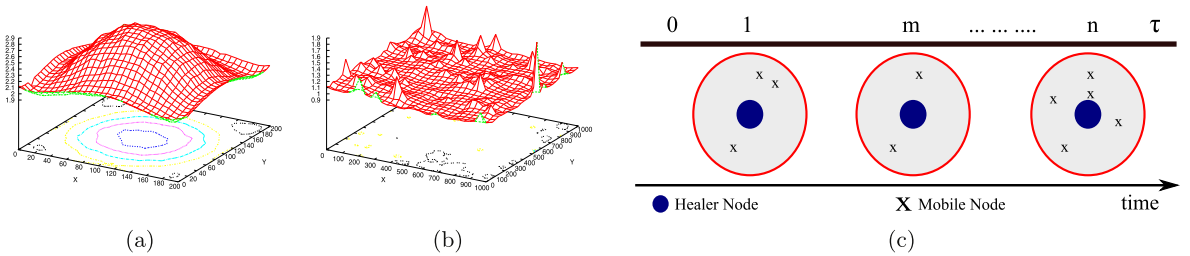


Fig. 4. (a) Spatial Distribution of RWP: The non-homogeneous distribution of node mobility indicates the center to be the most frequented region. (b) Spatial Distribution of TLW: The nearly-homogeneous distribution of node mobility indicates that all regions are equally frequented. (c) Healer Activation Problem: Without information about its future states, predicting when to broadcast a patch to optimize the healing time is hard.

2. Spatial Distributions: The spatial distribution (*i.e.*, frequency of visits in the simulation area) of the mobility models reveals another reason behind the slow infection propagation. In order to evaluate the spatial distribution of infected nodes that move according to each of the models, we take an approach similar to [37]. Specifically, we divide the simulation area into small size cells (e.g., divide a $1000 \times 1000\ m^2$ into $20 \times 20\ m^2$ size cells) and characterize each one of them using a histogram that captures the duration of how long an infected node stays in a particular cell. We end the simulation after 50,000 s.

Fig. 4(a) shows the resulting spatial distribution and contour lines for a particular simulation run using RWP. We observe that the spatial distribution has a peak in the middle of the area, i.e., an infected node is most likely to be found in the central cells of the simulation area and the probability that a node is located at the border of the area goes to zero. Fig. 4(b) shows the spatial distribution and contour lines for TLW. Observe that the non-homogeneous behavior seen in the case of RWP is absent in the case of TLW, i.e., TLW exhibits a homogeneous spatial distribution. The reason for the non-homogeneous behavior in RWP is well known [38,39,37]. In short, RWP chooses a uniformly distributed destination point rather than a uniformly distributed angle. This means that nodes located at the border of the simulation area are very likely to move back toward the middle of the area. However, this is not the case as per the original definition [19] of TLW. Under a TLW, at the beginning of each step, an infected node chooses a direction randomly from a uniform distribution of angle within $[0, 2\pi]$, a finite flight time randomly based on some distribution, and its flight length and pause time from some chosen probability distributions. In the long run, the positions of the random walker (infected node in our case) has been shown to converge to another distribution, called the *Levy stable distribution*, which leads to super-diffusive paths, thus making the infected nodes cover the area in a nearly homogeneous manner.

In summary, in case of RWP, depending on the origin of the infection, the spread can progress rapidly because most nodes have to pass through a common point in the center which also explains why the contact rate of the nodes is higher than that in TLW. In case of TLW, due to the underlying homogeneous behavior, the rate of infection propagation is nearly the same irrespective of the point of origin of the infection.

Impact on the design of countermeasures:

- *Static healers placement*: In case of RWP, positioning a few static healers somewhere near the center of the field in a non-overlapping manner should suffice because most nodes will traverse the central point in the field anyways. However, this is not the case for TLW, because the node distribution is uniform across the field, thus requiring a way to optimize healer placement such that they cover as much field as possible.
- *Healer patch dissemination*: In case of designing a healer for TLW, having a higher patch dissemination rate will result in a lot of patches being delivered to the same set of nodes since due to the low velocity (and thus low contact rate) many nodes may continue to stay within the proximity of the healer. Therefore, for a system optimizing energy, healer patch dissemination is a function of the contact rate (details in Section 4.5).

4. Defense protocols based on static healers

In this section, we discuss defense protocols against mobile malware. We first present the problem definition, formally define the static optimal healer activation problem, and design a greedy approximation algorithm. We discuss strategies for healer placement and present three families of static healers heuristics: *randomized*, *profile*, and *prediction*.

4.1. Problem definition

Healers have the ability to broadcast a patch periodically at every epoch τ. We consider the decision problem of when the healer node should be activated (i.e. switched on) within this time period to deliver a patch, to optimize along two dimensions: (i) the time it takes to heal the entire system and (ii) the total number of patches broadcasted.

We assume that healers are not susceptible to the infection. In addition to this, we also assume that healers can

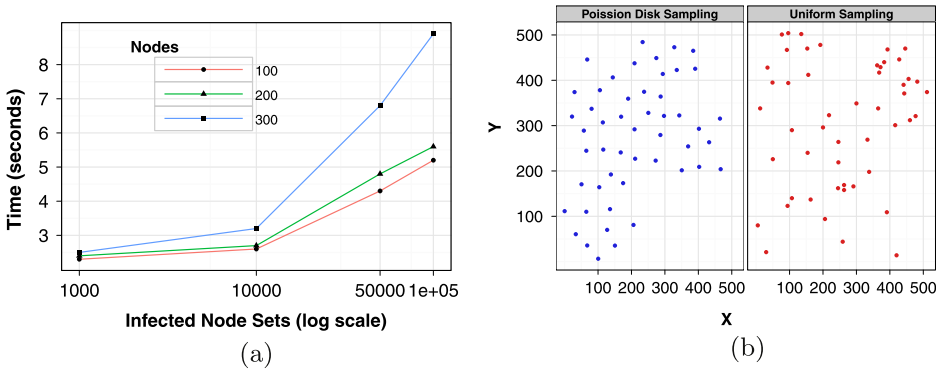


Fig. 5. (a) Oracle Performance: The algorithm terminates in less than 8 s even in long simulation scenarios. (b) Poisson Disk Sampling: Healers are no longer placed very close to each other thereby increasing their coverage of the simulation area.

sense the number of neighbors surrounding them but cannot determine which of the nodes are infected/susceptible/recovered.[1] Note that this increases the complexity of the problem significantly. Consider the example in Fig. 4(c). At time slot 1, if the healer decides to utilize its patch, it will heal at most three nodes whereas at time slot m, it can heal at most two nodes and at time slot n, it can heal at most five nodes. An oracle that has access to the future will pick a time slot that will make an effective use of the patch to heal the maximum number of nodes (in this case, time slot n). However, in practice, the future is not available to healer nodes.

We ask the questions: *What is an effective strategy for positioning the static healers so that two healers will avoid healing the same set of infected nodes?* and *How does the healer decide whether it should deliver the patch or wait in anticipation of a higher number of nodes in the future?* Without loss of generality, we consider that the energy consumption in delivering the patch is much higher than any other communication activity initiated by a healer. Intuitively, we are solving the problem of effectively distributing a patch without knowing the arrival distribution of infected nodes.

4.2. Design of an oracle optimal healer

In the following, we formally define the static optimal healer activation problem, and design a greedy approximation algorithm instead.

Let us call the task of designing a strategy for an optimal healer as the T-Cover problem.

Definition 1 (*T-Cover Problem*). Let $\mathcal{I}$ be the set of all infected nodes in the network and $\mathcal{T} = \bigcup_i T_i$, where each T_i is the set of infected nodes seen by all the healers at time instance i. Furthermore, let no two healers exist within the range of each other, and a patch from a healer can heal all infected nodes within its range and will consume one time unit. The T-Cover problem of $I_t = (\mathcal{I}, \mathcal{T})$ is to find a set $W \subseteq \mathcal{T}$ such that it covers the entire set of infected nodes $\mathcal{I}$ (i.e. $\bigcup_{T_i \in W} T_i = \mathcal{I}$) and W has minimum cardinality.

Here, the cost c_i associated with T_i is equivalent to the time instance value, i.e., i. Minimizing the total cost $\sum_{i \in W} c_i$ is equivalent to minimizing both the total time to recover the infected nodes and the required number of patches to do so. For example, let $\mathcal{I} = \{1,2,3,4\}$ and $\mathcal{T} = \{T_1, T_2, T_3\}$ where $T_1 = \{1\}$, $T_2 = \{1,2,3\}$ and $T_3 = \{3,4\}$ be the sets of infected nodes seen by the healer, then the T-Cover is $W = \{2,3\}$ meaning that a patch should be deployed at time $t = 2$ and $t = 3$ for optimality. We can restate this optimization problem as a decision problem.

Definition 2 (*T-Cover Decision Problem*). Given a system $I_t = (\mathcal{I}, \mathcal{T})$, the T-Cover decision problem is to determine whether I_t has a T-Cover of size at most k.

In other words, we wish to determine whether there is a set $W \subseteq \mathcal{T}$ such that $|W| \leqslant k$ and $\bigcup_{T_i \in W} T_i = \mathcal{I}$. In essence, T-Cover problem is the same as the min set cover problem, which we define below for completeness.

Definition 3 (*min set cover (MSC) Problem*). Let $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ be a collection of finite sets, where S_i's elements are drawn from a universal set $U = \bigcup_{i=1}^{m} S_i$. The MSC of $I_s = (U, \mathcal{S})$ is a set $C \subseteq \mathcal{S}$ such that $\bigcup_{S_i \in C} S_i = U$ and C has minimum cardinality.

For example, assume $U = \{1,2,3,4,5\}$ and $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$, where $S_1 = \{1,2,3\}$, $S_2 = \{2,4\}$, $S_3 = \{3,4\}$ and $S_4 = \{4,5\}$. The minimum set cover is $C = \{S_1, S_4\}$. Similarly, we can restate this optimization problem as a decision problem.

Definition 4 (*min set cover (MSC) Decision Problem*). Given $I_s = (U, \mathcal{S})$, the MSC decision problem is to determine whether I_s have a set cover of size at most k.

In other words, we wish to determine whether there is a set $C \subseteq \mathcal{S}$ such that $|C| \leqslant k$ and $U = \bigcup_{S_i \in C} S_i$. As the MSC decision problem is NP-Complete, it follows that the T-Cover decision problem is NP-Complete.

[1] Identifying the state of a node based on the interaction with the node is quite similar to the problem of detecting rootkits using the intrusion detection systems (IDSs) that rely on the system itself. In fact, when a system is compromised by rootkits, IDSs must not rely on the system [40]. Hence our healers treat each node equally.

Algorithm 1. Greedy Approximation (ORACLE).

Input Let $\mathcal{I}$ be the list of all infected nodes, S_i be the set of infected nodes seen at each time i, w_i be the list of costs associated with each arrival at i
Initially:
R is the set of elements that are not covered as yet
C is the set of covered elements
w is the weight vector
$R = \mathcal{I}$ and $C = \phi$
repeat
let S_i be the set that minimizes $\frac{w_i}{|S_i \cap R|}$
$C = C \cup \{S_i\}$
$R = R - S_i$
until $R = \phi$
return C

According to the above theorem, we can employ any heuristic that solves the set cover problem to solve the T-Cover problem. Algorithm 1, based on the greedy set cover algorithm [41], gives a greedy approximation for the T-Cover. The algorithm takes as input the arrival times of the infected nodes. Here, S_i is the set of infected nodes seen at any one time instant and we equate the weight vector w_i to the time of arrival – cost of healing nodes at a later time is higher because it introduces delay. The main loop iterates for $O(n)$ time, where $|\mathcal{I}| = n$. The minimum W can be found in $O(\log m)$ time, using a priority heap, where there are m sets in a set cover instance giving us a total time of $O(nlog(m))$. Fig. 5(a) shows that even in the presence of hundreds of thousands of node sets, we are able to compute the optimal solution in under 8 s.

4.3. Effective healer placement

Since the healers are static, the healer placement has an impact on our defense protocols and thus their coverage area depends on their placement strategy. Our simulations showed that a naive placement using uniform random distribution resulted in a scenario where many healers ended up covering the same region thereby leaving a lot of uncovered area. Another naive approach is the grid placement of healers in which healers cover the entire arena and therefore each mobile node will always be in the range of at least one healer. This approach would require N number of healers to cover the entire arena which could be a very large number depending on the size of arena and the range of healers.[2,3] Note that the infection containment problem becomes trivial in case of grid placement. For instance, if healers were placed in grids, the defense protocol would require all the healers to broadcast one patch each at the same time instance t and thus, the entire system would be recovered in one second at the cost of N patches. However, in realistic environments, it is not practical to have so many static healers. We focus instead of scenarios using a much smaller number of static healers.

For our healer placement strategy, what we need is a type of a constraint that rejects certain configurations that place healers very close to each other. This problem can be directly reduced to a problem from the field of computer vision which involves producing sampling patterns with a blue noise Fourier spectrum. Formally, the problem can be defined as the limit of a uniform sampling process with a minimum-distance rejection criterion. Successive points are independently drawn from the uniform distribution $[0, 1]$. If a point is at a distance of at least R from all points in the set of accepted points, it is added to that set. Otherwise, it is rejected. The choice of R controls the minimum allowable distance between points. This procedure called *Poisson Disk Sampling* [42] has been actively studied and many efficient algorithms exist. Due to space constraints, we do not discuss this algorithm further but refer the reader to the linear algorithm outlined by [42]. We adapted this algorithm by setting $R = 2r$, where r is the range of our each healer. Fig. 5(b) clearly highlights the merits of using this specific sampling process – healers are no longer close to each other and hence cover more of the simulation area.

4.4. Family of randomized healers

We first present a heuristic where a healer randomly decides at what time within an epoch to send a patch. Note that a healer will decide to send a patch regardless of the number of nodes in its vicinity. Fig. 6 depicts the state machine of the randomized healer (*RH*). It contains two states, an *initialization phase* where an *epoch timer* is started and an *execution phase* where the healer prepares to deliver a patch. The *epoch timer* fires a callback function that has two responsibilities: (i) pick a random time from the interval $[0, \tau]$, where τ is the *epoch length*, and use this random time to schedule a broadcast, called the *patch timer* and (ii) re-schedule the *epoch timer* to be fired for the next epoch. τ depends on the range of the healer and velocity of the mobile node. When the *patch timer* expires, the healer broadcasts a patch with a probability p, we call it the *patch deployment probability*.

Algorithm 2. Randomized Healers (RH).

Input Epoch length τ and patch deployment probability p
Initially:
start *epoch_timer*(τ)
Upon the expiration of *epoch_timer*:
select a duration t randomly from $(0, \tau)$
start *patch_timer*(t)
start *epoch_timer*(τ)
Upon the expiration of *patch_timer*:
Broadcast a patch with probability p

Algorithm 2 outlines the pseudo-code for the randomized healer. Varying p will generate a family of randomized healers. On one hand, setting $p = 1$ ($RH_{(p=1)}$) makes the healer broadcast a patch at every epoch and thus

[2] For an arena of 500×500 (m)^2 and 20 m healer-range, N would be at least 157, whereas we used $N = 20$ healers for the same setup.
[3] N would no longer be a fixed number.

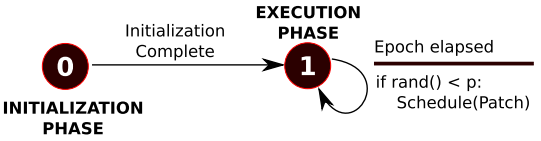


Fig. 6. State machine of a randomized healer.

attempts to minimize the time it takes to heal the system. However, notice that the number of patches delivered would be equal to $\frac{D_{sim}}{\tau}$, where D_{sim} is the simulation duration. On the other hand, setting $p < 1$ makes the healer broadcast a patch only during certain epochs. The time taken to heal the system is inversely proportional to p whereas the number of patches delivered is directly proportional to it.

4.5. Family of profile healers

One limitation of the RH approach is that healers may send more patches than needed since they decide to send patches regardless of how many infected nodes are present in their proximity. We propose a new approach, PH, where a healer attempts to learn the arrival distribution of nodes and subsequently determine whether or not it is cost effective to deliver a patch. The decision is made based on a threshold that captures the number of nodes in its vicinity.

Each healer can exist in one of three states as depicted in Fig. 6 – an *initialization phase* which prepares the healer, a *learning phase* where the healer passively records the number of neighbors it is observing during each epoch (in general), and an *execution phase* where the healer utilizes information that it learnt during the previous phase to decide whether or not to deliver a patch. Algorithm 3 describes this healer (PH) in detail. In the initialization phase, each healer sets its own state to `LEARNING` and starts the sensing timer (*sensing_timer*) with a duration of 1 s. Upon the expiration of *sensing_timer*, the healer checks whether the observation time T has elapsed yet. If not, the healer records the number of neighbors (*num_of_neighbors*) in its proximity and restarts *sensing_timer*. When the observation time T has elapsed, the healer first estimates the *threshold* from the recorded information and moves to the `EXECUTION` state. Now at every second, the healer checks whether the number of neighboring nodes exceeds the *threshold*. If so, the healer deploys a patch and starts a timer (*epoch_timer*), which expires at the end of the current epoch. Until then, the healer does no sensing at all. Upon the expiration of *epoch_timer*, the healer starts sensing again by setting *sensing_timer*.

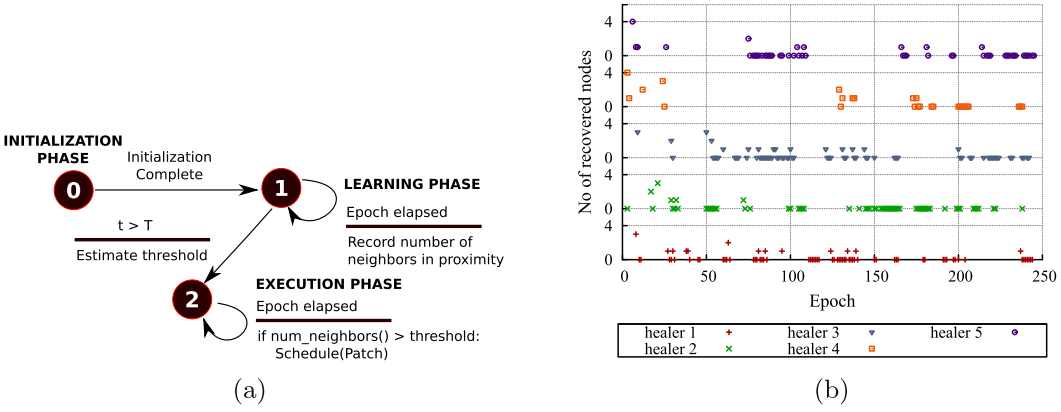


Fig. 7. (a) State Machine of a profile healer. (b) Motivating Backoff: Most consecutive patches do not heal infected nodes indicating that it is better to backoff after a patch delivery.

Table 1
Summary of protocols we propose and evaluate.

Protocol	Description	Parameters (default values)
RH_p	Randomized healers	Patch deployment probability p (=1)
PH_{MSD}	Profile healers	Decision threshold $MSD = Mean + 1.5 \times stddev$
PH_M	Profile healers	Decision threshold $M = Median$
PHB_M	Profile healers with backoff	Maximum No. of epochs to backoff η (=2) and decision threshold M
D-PHB_M	Profile healers with backoff and dynamic threshold scheme	Observation epochs Γ (=10), maximum No. of epochs to backoff η (=2) and decision threshold M
PDH	Prediction healers	

Algorithm 3. Profile healers (PH).

Input Epoch length τ, observation time T such that $T > 1$
Initially:
$t \leftarrow 0, \Delta \leftarrow 1, state \leftarrow$ LEARNING, $next\_epoch\_time \leftarrow 0$
Start $sensing\_timer(\Delta)$ ▷ Start timer with duration Δ
Upon the expiration of $sensing\_timer$:
$t \leftarrow t + \Delta$
if $state =$LEARNING **then**
if $t < T$ **then**
Record $num\_of\_neighbors$ in proximity
else
Estimate $threshold$ from the recorded $num\_of\_neighbors$ at each Δ
$state \leftarrow$ EXECUTION
$next\_epoch\_time \leftarrow t + \tau$
end if
Start $sensing\_timer(\Delta)$
else
if current $num\_of\_neighbors > threshold$ **then**
Broadcast a patch
Start $epoch\_timer(next\_epoch\_time - t)$
else
Start $sensing\_timer(\Delta)$
end if
end if
Upon the expiration of $epoch\_timer$:
$t \leftarrow next\_epoch\_time$
$next\_epoch\_time \leftarrow t + \tau$
Start $sensing\_timer(\Delta)$

The goal of *learning phase* is to learn the distribution of node arrivals specific to a healer's locality for a certain *observation time* T which is a multiple of τ. Specifically, the goal is to learn a threshold of nodes that will determine whether the healer should send a patch or not. We use two metrics described below.

- $MSD = Mean + 1.5 \times StandardDeviation$: MSD is well-known for normal distributions and makes the healer broadcast a patch only if the number of neighbors exceeds its estimate of the 95th percentile.
- $M = Median$: M is the median of the observed distribution. Median is very robust to outliers – it handles cases where a healer observes a burst of infected nodes during an epoch.

During our simulations, we observed that relying solely on a *threshold* was leading to a wastage of patches – due to the low contact rate we observed in Section 3.2. Consider Fig. 7(b) which depicts the healing sequence of a set of five healers during the epochs of one simulation run. Points situated at 0 indicate that the healer deployed a patch as the number of neighbors was above the threshold but the patch *did not* heal any infected nodes. Any other number indicates the number of infected nodes healed with that patch. Observe that most patches are going to waste, *i.e.*, they are not healing any nodes. In the worst case, it takes at least $\frac{healerrange}{nodevelocity}$ seconds for a node to go out of range of a healer. Therefore, for shorter epochs, consecutive patches are delivered to the same set of nodes. We address this issue by introducing a *random backoff*, i.e., once a patch has been broadcast, the healer selects a random backoff delay κ from the interval $(0, \eta)$, where η is the maximum backoff in epochs, and skips that many epochs. Algorithm 4 also describes the backoff algorithm in detail. We refer to this algorithm as PHB. This algorithm is similar to Algorithm 3 except that each healer now selects a random backoff delay $\kappa \in \{1, \ldots, \eta - 1\}$ to remain silent after the deploy of a patch. Upon the expiration of this remaining period, the healer starts sensing again.

Algorithm 4. Profile healers with backoff (PHB).

Input Epoch time τ, observation time T such that $T > 1$ and maximum backoff η such that $\eta > 1$
Initially:
$t \leftarrow 0, \Delta \leftarrow 1, state \leftarrow$ LEARNING, $next\_epoch\_time \leftarrow 0$
Start $sensing\_timer(\Delta)$ ▷ Start timer with duration Δ
Upon the expiration of $sensing\_timer$:
$t \leftarrow t + \Delta$
if $state =$ LEARNING **then**
if $t < T$
Record $num\_of\_neighbors$ in proximity
else
Estimate $threshold$ from the recorded $num\_of\_neighbors$ at each Δ
$state \leftarrow$ EXECUTION
$next\_epoch\_time \leftarrow t + \tau$
end if
Start $sensing\_timer(\Delta)$
else
if current $num\_of\_neighbors > threshold$ **then**
Broadcast a patch
Randomly select κ between $(0, \eta)$
Start $epoch\_timer(next\_epoch\_time - t + \kappa \times \tau)$
else
Start $sensing\_timer(\Delta)$
end if
end if
Upon the expiration of $epoch\_timer$:
$t \leftarrow get\_current\_time()$
$next\_epoch\_time \leftarrow t + \tau$
Start $sensing\_timer(\Delta)$

Both PH and PHB have two shortcomings. First, both require to wait until the end of the learning phase (i.e., a certain observation time T to learn the distribution of node arrivals) to start healing the system. Second, the healers learn and estimate the threshold only once. This may not characterize the node arrival distribution of the system accurately. Note that any attempts to improve one

shortcoming will worsen the other. For instance, on one hand, decreasing the observation duration T, to start healing early, introduces the possibility of inaccurately estimating the threshold (due to insufficient data points) and hence leads to consuming more patches. On the other hand, if T were to be increased (to better capture the node arrival distribution), it results in an increased system recovery time. To address these limitations, we adopt a hybrid approach where healers perform online learning and heal the system simultaneously. This approach is an extension of the PHB algorithm where each healer never stops learning. Moreover, at the end of every Γ epochs, each healer dynamically estimates a new decision threshold based on what it has learned in the last Γ epochs and uses the newly estimated threshold for the next Γ epochs. We refer to this algorithm as D-PHB (see Algorithm 5). Note that, unlike both PH and PHB, each healer can be either in LEARN_EXEC state when it both learns and heals or in ONLY_LEARN state when it only learns. However, each D-PHB healer uses random backoff mechanism like PHB healers.

Algorithm 5. Profile healers with backoff and dynamic threshold scheme (D-PHB).

Input Epoch time τ, observation epochs Γ such that $\Gamma > 1$, initial threshold α, and maximum backoff η such that $\eta > 1$
Initially:
$t \leftarrow 0, \Delta \leftarrow 1, next\_epoch\_time \leftarrow \tau$, $state \leftarrow$ LEAR_EXEC
$threshold \leftarrow \alpha, epoch\_count \leftarrow 0, time\_to\_switch\_state \leftarrow 0$
Start $sensing\_timer(\Delta)$ ▷ Start timer with duration Δ
Upon the expiration of $sensing\_timer$:
$t \leftarrow t + \Delta$
Record $num\_of\_neighbors$ in proximity into Σ
if $state =$ LEAR_EXEC **then**
if current $num\_of\_neighbors > threshold$ **then**
Broadcast a patch
Randomly select κ between $(0, \eta)$
$time\_to\_switch\_state \leftarrow next\_epoch\_time - t + \kappa \times \tau$
$state \leftarrow$ ONLY_LEARN
end if
end if
if $t = next\_epoch\_time$ **then**
$epoch\_count \leftarrow epoch\_count + 1$
if $epoch\_count = \Gamma$ **then**
Estimate $threshold$ from the recorded $num\_of\_neighbors$ at each Δ
Clear records from Σ
$epoch\_count \leftarrow 0$
end if
$next\_epoch\_time \leftarrow next\_epoch\_time + \tau$
end if
if $t = time\_to\_switch\_state$ **then**
$state \leftarrow$ LEAR_EXEC
end if
Start $sensing\_timer(\Delta)$

4.6. Family of prediction healers

The optimal healer ORACLE (see Algorithm 1) has several advantages compared to the profile healers. Firstly, an optimal healer has the global view of the entire network, whereas a profile healer has only the local view (neighbors at its vicinity). Secondly, an optimal healer can explicitly identify the state (e.g., susceptible, infected, recovered) of every mobile node, but a profile healer is not capable of identifying the state of a mobile node in its proximity. Finally, while the former knows the future (i.e., time instance at which each healer is going to observe the maximum number of infected nodes), the latter has no such knowledge. All these advantages make the optimal healers ideal, but impractical. Having the first two capabilities of an optimal healer would make any healer impractical for real world. However, in case of the third capability, we can equip a healer with the ability to predict the event of observing relatively higher number of mobile nodes[4] in the near future with the goal of hitting a middle ground between the optimal healers and the profile healers. Therefore, we propose a new family of healers called *prediction healers* (PDH).

Similar to a profile healer, each prediction healer can exist in one of the three states as shown in Fig. 7(a), except it does not estimate a threshold. Instead, each healer now computes a *stationary transition probability matrix*.[5] Let X_t^h be the state, i.e., the total number of mobile nodes observed by the hth healer at time instance t. Further, assume that the stationary transition probability matrix for the healer h is $\mathcal{P}^h = [p_{ij}^h]_{n\times n}$ where $p_{ij}^h = \Pr[X_{t+1}^h = j | X_t^h = i]$, i.e., the probability of observing j nodes in the next time instance given that the healer has seen i nodes at the current time instance and n be the total number of mobile nodes in the system.[6] Each healer must deploy a patch during every epoch, but it is free to choose the deployment time instance within an epoch. This deployment time instance is chosen based on whether it is worth deploying the patch now or to hold off for a better future state that may be observed within this epoch. If the healer reaches the deadline of the current epoch and has not deployed the patch yet, it must deploy the patch right away. Each prediction healer uses a prediction function $\mathcal{F}$ which is based on this intuition. We define the function $\mathcal{F}$, formally, as follows:

$$\mathcal{F}(\lambda, x|\mathcal{P}) = \begin{cases} 1 & \text{if } \mathcal{G}(\lambda, x|\mathcal{P}) > \sum_y p_{xy}\mathcal{G}(\lambda - 1, y|\mathcal{P}) \\ 0 & \text{otherwise} \end{cases}$$

where λ is the remaining time to the deadline of the current epoch, x is the current state of the healer, y is any possible next state and $y \in [0, n]$, $\mathcal{P}$ is the transition probability matrix of the healer, and

[4] Mobile nodes in general, not only infected ones.
[5] Similar to the transition probability matrix of a Markov Model.
[6] Superscript means the identity of the healer, not the h-step transition probabilities of Markov chain.

$$\mathcal{G}(\lambda, x|\mathcal{P}) = \begin{cases} 0 & \text{if } \lambda < 0 \\ x & \text{if } \lambda = 0 \\ \max\left\{x, \sum_y p_{xy}\mathcal{G}(\lambda-1, y|\mathcal{P})\right\} & \text{otherwise} \end{cases}$$

Note that $\mathcal{G}(\lambda, x|\mathcal{P}) \geqslant \mathcal{G}(\lambda-1, x|\mathcal{P})$, for all λ, x. That is, the worth of the patch either stays the same or diminishes with the decrease in λ (equivalently, with the increase in time). In other words, the more a healer waits to deploy a patch, the more the patch loses its worth. Note that it captures the time constraint of the T-Cover problem. Fig. 8 shows the internals of the prediction function where x is the current state of the healer. In the next time instance, the healer can move to any state $j \in [0, n]$ with probability p_{xj}. The healer predicts the future and decides on whether or not to deploy a patch using $\mathcal{F}(\lambda, x|\mathcal{P})$ at the current state x.

Algorithm 6. Prediction Healers (PDH).

Input Epoch length τ, observation time T such that $T > 1$
Initially:
  $t \leftarrow 0, \Delta \leftarrow 1$, *state* $\leftarrow$ LEARNING, *deploy_status* $\leftarrow$ **false**
  $\lambda \leftarrow 0$ ▷ λ is the time to be elapsed until the deadline of the current epoch
  Start *sensing_timer*(Δ) ▷ Start timer with duration Δ
Upon the expiration of *sensing_timer*:
  $t \leftarrow t + \Delta$
  if *state* = LEARNING **then**
    if $t < T$ **then**
      Record *num_of_neighbors* in proximity and store in $\mathcal{S}$
    else
      Compute $\mathcal{P}$ from the recorded $\mathcal{S}$ ▷ $\mathcal{P}$ is the transition probability matrix
      state $\leftarrow$ EXECUTION
      $\lambda \leftarrow \tau$
    end if
  else if *state* = EXECUTION **then**
    $\lambda \leftarrow \lambda - 1$ ▷ Elapsed one second
    $x \leftarrow$current *num_of_neighbors* in promixity
    if *deploy_status* = **false then**
      if $\lambda = 0$ **or** $\mathcal{F}(\lambda, x|\mathcal{P}) = 1$ **then**
        Broadcast a patch
        deploy_status $\leftarrow$ **true**
      end if
    end if
    if $\lambda = 0$ **then**
      $\lambda \leftarrow \tau$
      deploy_status $\leftarrow$ **false**
    end if
  end if
  Start *sensing_timer*(Δ)

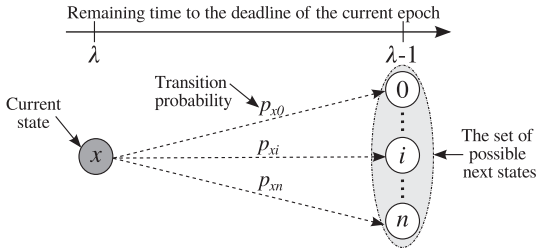


Fig. 8. The internals of the prediction function used by a prediction healer.

Algorithm 6 describes the healer in detail. During the LEARNING state, each healer records the number of neighbors observed at each time instance. After the observation period, each healer computes the transition probability matrix $\mathcal{P}$ and stores it for future reference. In the EXECUTION state, each healer decides to deploy the patch if (1) the healer has reached the deadline of the current epoch or (2) it is worth deploying at the current time instance based on $\mathcal{F}(\lambda, x|\mathcal{P})$. Whenever λ becomes zero, it performs some reinitialization to prepare itself for the next epoch.

Computing $\mathcal{F}(\lambda, x|\mathcal{P})$ requires a healer to compute the $\mathcal{G}(.)$ recursively from λ to 0. Recursive implementations of $\mathcal{F}(.)$ and $\mathcal{G}(.)$ are highly expensive when the system contains hundreds of nodes and the epoch length is in the order of minutes. In addition, recursive implementation wastes computations by solving the same subproblem multiple times. To overcome these challenges, we leverage *dynamic programming*, to efficiently implement $\mathcal{G}$ and $\mathcal{F}$. The intuition behind dynamic programming is to first solve the smaller subproblems and then utilize the answers to solve the overall problem. The pseudocode for $\mathcal{G}(.)$ is shown in Algorithm 7. A healer computes $\mathcal{F}(.)$ in a similar way.

Algorithm 7. Compute $\mathcal{G}(.)$.

Input Total number of nodes n, epoch size τ, the transition probability matrix PT
Output A matrix GT of size $\tau \times n$
1: Declare a matrix GT of size $\tau \times n$
2: **for** $j \leftarrow 0$ **to** n
3: $GT(0, j) \leftarrow j$
4: **end for**
5: **for** $i \leftarrow 1$ **to** τ
6: **for** $j \leftarrow 0$ **to** n
7: $sum \leftarrow 0$
8: **for** $k \leftarrow 0$ **to** n
9: $sum \leftarrow sum + PT(j, k) * GT(i-1, k)$
10: **if** $j > sum$ **then**
11: $GT(i, j) \leftarrow j$
12: **else**
13: $GT(i, j) \leftarrow sum$
14: **end if**
15: **end for**
16: **end for**
17: **end for**
18: **return** GT

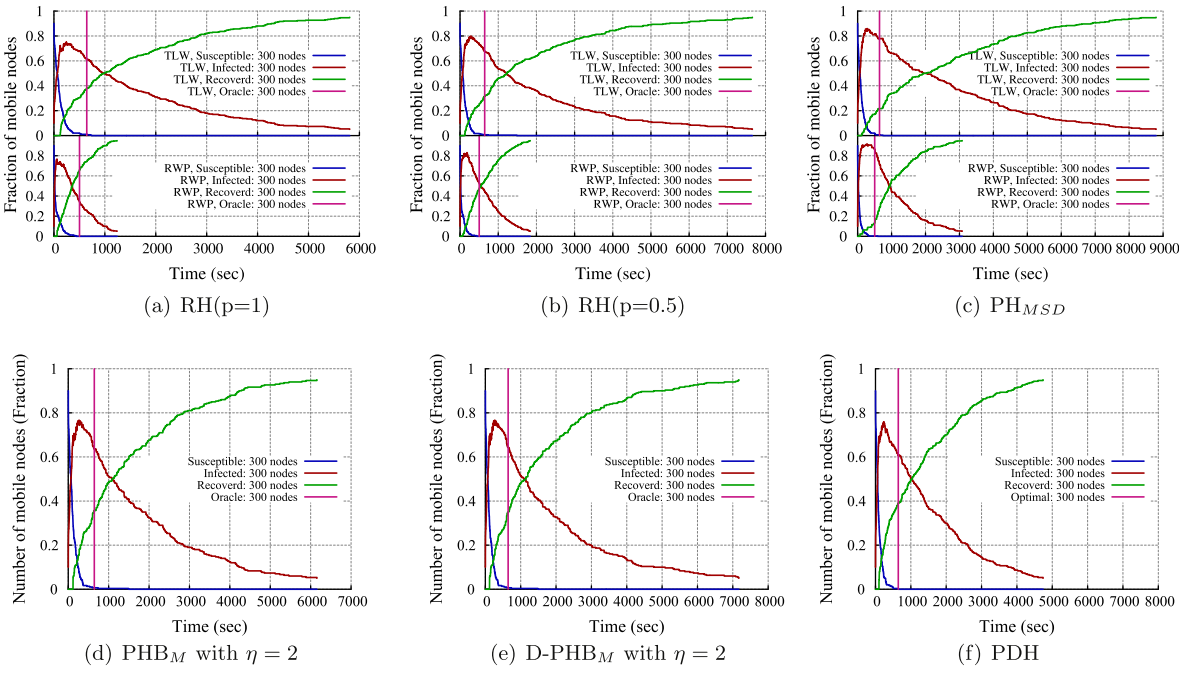


Fig. 9. Infection propagation and recovery in case of different healer families along with the values obtained using our ORACLE shown as the vertical lines.

5. Healer-based protocols evaluation

In this section, we describe our evaluation methodology and present the performance of the various healer-based defense mechanisms outlined in Section 4. Table 1 summarizes the notations and the parameters of the healer algorithms that we evaluate in the section.

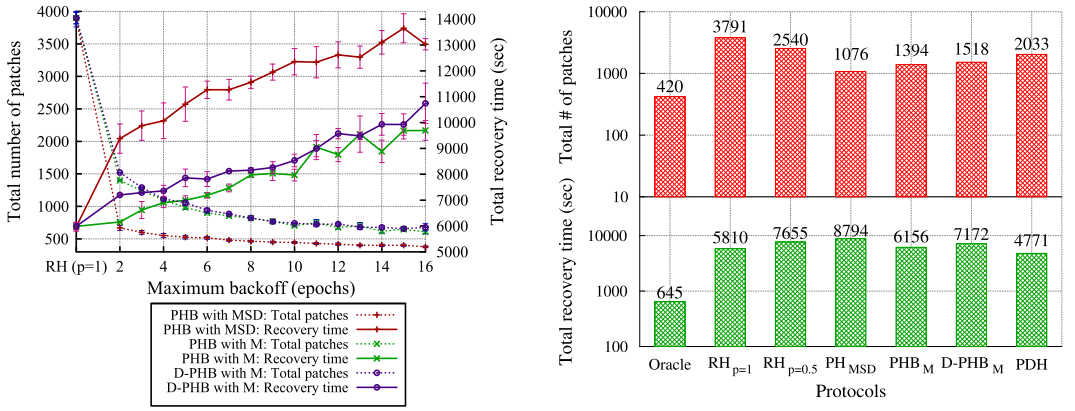
5.1. Evaluation methodology

To evaluate the performance of the families of healers we proposed, we simulate the various healer-based protocols we described (see Table 1) using the NS-3 [36] network simulator on a network containing 300 nodes. We perform two different sets of experiments, one with nodes having RWP and the other with TLW as their mobility model. We assume that the range of each healer is 20 meters and the *epoch length*, τ, is 30 s (so that each node stays within the range of a healer for one epoch length on an average before leaving the coverage area of the healer). In addition, 10% of the population is assumed to be initially infected to enable bootstrapping the system. We can technically start with one infected node (which was our initial attempt), but we observe that this only delays infection spread and increases the chance that infection will disappear. Healers are placed in the system using the strategy outlined in Section 4.3 and are activated (i.e., started) when the fraction of infected nodes exceeds 70% of the total population to give the system sufficient time to warm-up. We note that 70% is one possible worst case scenario and projects the capabilities of the healer. In real-world scenarios, this value depends on how fast one can setup healers during an epidemic outbreak. However, to analyze the sensitivity of these two parameters on the performance we conducted two-way analysis of variance (ANOVA)[7] tests with significance level of 0.05. We varied the number of initially infected nodes as 10%, 15%, and 20% and the time to activate healers when the 50%, 60%, and 70% of the population become infected. Both the parameters, either with or without the interaction between them, did not make any statistically significant impact on the performance. We also conducted pairwise comparison tests among the different categorical values of a parameter to see what significant differences are present among the values of the parameter. For both the parameters, the results indicate that there are no statistically significant pairwise differences between the categorical values. Therefore, we choose to proceed with 10% of the nodes as initially infected and to activate healers when 70% of the nodes becomes infected considering less aggressive nature in the both the cases.

Once the healers are activated, they follow the protocols outlined in Sections 4.4, 4.5, and 4.6. All results are averaged across 10 runs of each experiment, to obtain statistically significant results, by varying the *seed* of a pseudo-random number generator. To measure the performance of each protocol, we define:

- *Total recovery time*: It represents the amount of simulation time required by the set of healers to recover at least 95% of the nodes in the system.
- *Total number of patches*: It represents the count of patches deployed by the set of healers to heal the system such that at least 95% of the total number of nodes are recovered.

[7] In statistics, a two-way ANOVA test is used to examine the influence of different categorical independent variables on one dependent variable [43].



(a) Effect of varying maximum backoff (b) Summary

Fig. 10. Comparison of Healer families (a) D-PHB$_M$ performs as good as PHB$_M$ in terms of both the metrics with the added advantage that it does not have an observation time. (b) Summary of the performances of healer families for TLW.

Note that we choose 95% to account for scenarios similar to the *rare block problem* [44] in P2P networks – we observe the presence of some infected nodes that take exceedingly long time to enter the range of healers because they are wandering along the edge of the field and hence prolong our simulation.

5.2. Results for family of randomized healers

Fig. 9(a) and (b) show the temporal view of infection propagation and the recovery of the system for RH family using RWP and TLW. The graphs show that regardless of the protocol, the required recovery time is always smaller in case of RWP than TLW which is due to RWP's higher contact rate.

Fig. 9(a) shows the required recovery time for randomized healers with $p = 1$, *i.e.*, $RH_{(p=1)}$. The upper graph is for TLW and the lower one is for RWP. Additionally, we also point out the recovery time required by ORACLE using a vertical line. In case of RWP, ORACLE requires 502 s to heal the system whereas $RH_{(p=1)}$ requires almost double this time, *i.e.*, 1241 s. In case of TLW, ORACLE needs 645 s to heal the system whereas $RH_{(p=1)}$ requires about nine times the optimal recovery time. We also note that the recovery time required by $RH_{(p=1)}$ is the minimum time that we can achieve using healers that do not depend on system feedback (*e.g.*, estimating the arrival distribution of nodes).

Fig. 9(b) shows the results for $RH_{(p=0.5)}$, i.e., each healer deploys a patch per epoch with a probability $p = 0.5$. It is expected that $RH_{(p=0.5)}$ requires more time than $RH_{(p=1)}$ to heal the system since now the healers skip some epochs. In comparison with the recovery time required by $RH_{(p=1)}, RH_{(p=0.5)}$ shows 48% increase in case of RWP and 31% increase in case of TLW.

5.3. Results for family of profile healers

To measure the impact of different maximum backoff values on the PHB$_{MSD}$ and the PHB$_M$, we varied the maximum backoff from 2 epochs to 16 epochs. Fig. 10(a) shows the results of this experiment. We also include the results of $RH_{(p=1)}$ as a baseline of the performance. We use two *Y*-axes for this graph: the left one for the total number of patches and the right one for the total recovery time. Each point is the average of 10 different runs of the simulation and is plotted along its 95% confidence intervals. With the increase in maximum backoff values, the total recovery time is increasing rapidly in case of PHB$_{MSD}$ in comparison with PHB$_M$. On the other hand, the total of number of patches is decreasing rapidly for PHB$_{MSD}$ in comparison with PHB$_M$. We conjecture that if the recovery time is to be optimized, then PHB$_M$ is a better solution; but if the energy of the healers is to be optimized, then the PHB$_{MSD}$ is a better choice. However, the downside of PHB is its large observation time. D-PHB is a solution to this downside of PHB. We also include the performance of D-PHB$_M$ in Fig. 10(a). The results demonstrate that D-PHB$_M$ performs as good as PHB$_M$ in terms of both the metrics. So if the large observation time is unacceptable, D-PHB$_M$ heals the system as fast as PHB$_M$ and does not require any observation time.

Let X_{MSD} and X_M represent a profile-based healer X that utilizes $MSD(= Mean + 1.5 \times Standard\ Deviation)$ and $M(= Median)$ as its threshold, respectively. Fig. 9(c) shows the performance of PH for the RWP and the TLW mobility models. PH$_{MSD}$ requires more time to heal the system in comparison with the other two *RH* healers. Since we are more interested in the human-mimicking mobility model, we evaluate PHB and D-PHB for only TLW in Fig. 9(d) and (e), respectively. Due to space limitation, we present the performance of PHB and D-PHB with maximum backoff $\eta = 2$ and M as the threshold value in Fig. 9(d)–(e). When we compare PH$_{MSD}$, PHB$_M$, and D-PHB$_M$ using the TLW mobility model, PHB$_M$ outperforms the other two in terms of total recovery time.

5.4. Results for family of prediction healers

Fig. 9(f) shows the temporal view of the infection propagation and recovery of the system using prediction healers in case of the TLW mobility model. The prediction healers require the least amount of time to recover the sys-

tem when compared to the *RH* and *PH* families. This is due to the prediction capability of the healers that allow them to deploy patches efficiently. The recovery time required by the prediction healers is 18% less and 22.5% less than the best random healers $RH_{(p=1)}$ and the best profile healers PHB_M, respectively.

Summary: Fig. 10(b) summarizes the results consisting of both the metrics obtained by each of the healers for the TLW mobility model. In terms of the number of patches, PH_{MSD} requires the least number of patches but at the cost of a larger recovery time. The prediction healers PDH outperforms the others in terms of the total recovery time. However, it requires 89% more patches than PH_{MSD}. Next comes the $RH_{(p=1)}$ that requires 21% more recovery time than PDH. However, in order to achieve this recovery, $RH_{(p=1)}$ has to deploy the maximum amount of patches. In fact, PHB_M performs the best since it requires only 29% more recovery time in comparison to PDH and only 30% more patches than that of PH_{MSD}.

Our results show that each of the schemes has advantages and disadvantages. First, randomized healers offer the immediate advantage that they do not rely on system feedback nor do they have to learn the system before starting to recover the system. Second, prediction healers would be beneficial in a time-constrained system as these healers are fastest at recovering an infected system by utilizing prediction capability. However, they result in using 1.8× patches than the profile healers (with *MSD* threshold). Finally, profile healers with backoff offer intelligent decision making thereby saving energy in the form of utilizing less number of patches and would benefit the most in an energy-constrained environment. However, they result in taking 1.8× time to recover in comparison to PDH healers. On the other hand, when compared with the ORACLE, we observe that PDH, RH, and PHB_M healers take 7.4×, 9×, and 9.5× recovery time, respectively. Furthermore, to recover the system PH_{MSD} healers require 2.5× patches than the ORACLE.

6. Related work

We divide this section into two parts. In the first part, we discuss related work in the area of mathematical modeling and analysis of worms and viral epidemics. We then move on to discussing the existing works on controlling the worm propagation.

Epidemic models. Wired networks have been the focus of most literature on worm propagation. A comprehensive overview of major malware outbreaks in networks with a discussion of their trends is given in [45]. There are two popular models that are generally used to describe worm propagation: deterministic [5,7,46–50,6] and stochastic [51,52] epidemiological models. Staniford et al. [48] use the SIR epidemiological model to capture the effects of human countermeasures and the congestion due to the worm spread. Shen et al. [53] provide a discrete-time worm model that considers patching, cleaning and certain local scanning techniques. All these approaches abstract network topology and change in the size of vulnerable population as the worm spreads. Theodorakopoulos et al. [54] take deterministic modeling one step further and combine it with a game theoretic process that involves learning. A probabilistic queueing framework has been proposed in [55] to model the spread of mobile viruses using short range wireless interfaces (e.g., Bluetooth) of mobile devices. While similar in spirit, our work focuses on modeling infection dynamics in MANETs as a function of the mobility models.

Peng et al. [56] propose a two-dimensional cellular automata to characterize the propagation dynamics of worms in smartphones. Their scheme integrates an infection factor evaluate the spread degree of infected nodes, and a resistance factor to evaluate the degree that susceptible nodes resist. Wang et al. [57] deploy agents in the form of hidden contacts on the device to capture messages sent from malicious applications. The authors combine these captured messages in conjunction with a latent space model to estimate the current dynamics of the system and use this to predict the future state of malware propagation within the mobility network. Our work is complementary to these efforts in that our decentralized algorithms can utilize their models during the learning phase. Szongott et al. [58] present a prototype of a replicating mobile malware that spreads from device to device in downtown Chicago. Using simulations, they show that smartphones create a viable substrate for epidemic mobile malware. Our work differs from them in two key aspects. First, unlike them, we use a more realistic truncated levy walk mobility model. Second, they only study infection propagation whereas we propose several algorithms for recovery.

Worm containment. There have been some works in controlling the spread of worms inside a wireless network [59,60,8,61,20,21]. Williamson et al. [59] present a technique to limit the rate of connections to "new" machines. This is effective at both slowing and halting virus propagation without affecting normal traffic. Their work is based on heuristics and simulations which consider a static choice of reduced communication rate. Wong et al. [60] present a technique that relies on limiting the contact rate of worm traffic. Specifically, they investigate rate control at individual end hosts and at the edge and backbone routers, for both random propagation and local-preferential worms. They show that both host and edge-router based rate control result in a slowdown that is linear to the number of hosts implementing the rate limiting filter.

More recently, Barbera et al. [62] consider the problem of computing an efficient patching strategy to stop worm spreading between smartphones. They consider cases where the worm spreads between the devices and where the worm attacks the cloud before moving to the device. Tang et al. [63] propose distributing patches to certain key nodes so they can opportunistically disseminate them to the rest of the network. In their work, they present a predictive mobile malware containment system where devices collect co-location data in a decentralized manner and report to a central server which processes and targets delivery of hot fixes to a small subset of k devices at runtime. In contrast, our work does not assume a central server and all our algorithms are fully decentralized.

Cole et al. [61] present both analytic and simulation analysis of worm propagation focusing specifically on the

features of a tactical, battlefield MANETs which are unique to a defense environment. Their goal was to develop an accurate set of performance requirements on potential mitigation techniques of worm propagation for such MANETs. Zou et al. [64] compare email worm propagation on three topologies: power law, small world, and random graph topologies; and then study how the topology affects immunization defense on email worms. Their email worm model includes the effect of human interactions. Yang et al. [65] utilize a software diversity approach to deal with the spread of worm in wireless sensor networks. Zhu et al. [66] take into account the social relationship of mobile users to contain MMS worms within a limited range in cellular networks. Unlike them, we introduce a suite of defense protocols used by a set of static healers to thwart the epidemic spread inside MANETs.

7. Conclusion

Mobile malware have become an emerging problem that threatens smartphones which are growing significantly in recent days. In this paper, we considered realistic mobility patterns to model proximity dependent malware and compared them against de facto models like random waypoint mobility model. We presented several defense mechanisms that allow tuning of parameters to control two dimensions of optimization – either time to recovery or energy utilized. The extensive evaluation of all our defense mechanisms shows that prediction healers would be more effective in a time constrained environment whereas profile healers would benefit the most in an energy constrained environment.

References

[1] R. Potharaju, E. Hoque, C. Nita-Rotaru, S. Sarkar, S. Venkatesh, Closing the pandora's box: defenses for thwarting epidemic outbreaks in mobile adhoc networks, in: Proc. of IEEE MASS, 2012, pp. 200–208.
[2] McAfee Threats Report: 3rd Quarter 2011. <http://goo.gl/jIQPJ>.
[3] Juniper Mobile Threats Report 2010-11. <http://goo.gl/v3yFg>.
[4] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, N. Weaver, Inside the slammer worm, IEEE Secur. Privacy 1 (4) (2003) 33–39.
[5] C.C. Zou, W. Gong, D. Towsley, Code red worm propagation modeling and analysis, in: Proc. of CCS, ACM, 2002, pp. 138–147.
[6] A. Wagner, T. Dübendorfer, B. Plattner, R. Hiestand, Experiences with worm propagation simulations, in: Proc. of WORM, ACM, 2003, pp. 34–41.
[7] J.O. Kephart, S.R. White, Directed-graph epidemiological models of computer viruses, in: Proc. of Sym. on Research in Sec. and Priv, IEEE CompSoc, 1991, pp. 343–359.
[8] S.H. Sellke, N.B. Shroff, S. Bagchi, Modeling and automated containment of worms, IEEE TDSC 5 (2) (2008) 71–86.
[9] Single NFC bonk subjugated Samsung Galaxy SIII and slurped it out, 2012. <http://www.theregister.co.uk/2012/09/21/android_nfc/>.
[10] C. Miller, Exploring the NFC attack surface, in: Blackhat, 2012.
[11] McAfee warns of NFC malware risk, 2013. <http://www.itpro.co.uk/malware/19275/mcafee-warns-nfc-malware-risk>.
[12] Wall Of Sheep Hacker Group Exposes NFC's Risks At Def Con 2013, 2013. <http://www.forbes.com/sites/michaelvenables/2013/08/08/wall-of-sheep-near-field-communication-hack-at-def-con/2/>.
[13] A. Khelil, C. Becker, J. Tian, K. Rothermel, An epidemic model for information diffusion in manets, in: Proc. of MSWiM, ACM, 2002, pp. 54–60.
[14] A. Bose, X. Hu, K.G. Shin, T. Park, Behavioral detection of malware on mobile handsets, in: Proc. of ACM Mobisys, ACM, 2008, pp. 225–238.
[15] C. Fleizach, M. Liljenstam, P. Johansson, G.M. Voelker, A. Mehes, Can you infect me now? Malware propagation in mobile phone networks, in: Proc. of ACM WORM, ACM, 2007, pp. 61–68.
[16] A. Bose, K.G. Shin, On mobile viruses exploiting messaging and bluetooth services, in: Proc. of SecureComm, IEEE, 2006, pp. 1–10.
[17] G. Zyba, G.M. Voelker, M. Liljenstam, A. Méhes, P. Johansson, Defending mobile phones from proximity malware, in: Proc. of INFOCOM, IEEE, 2009, pp. 1503–1511.
[18] R. Potharaju, C. Nita-Rotaru, Pandora: a platform for worm simulations in mobile ad-hoc networks, ACM SIGMOBILE Mobile Comput. Commun. Rev. 14 (4) (2011) 16–18.
[19] I. Rhee, M. Shin, S. Hong, K. Lee, S.J. Kim, S. Chong, On the levy-walk nature of human mobility, IEEE/ACM Trans. Netw. 19 (3) (2011) 630–643.
[20] M. Khouzani, E. Altman, S. Sarkar, Optimal quarantining of wireless malware through reception gain control, IEEE Trans. Autom. Control 57 (1) (2012) 49–61.
[21] M. Khouzani, S. Sarkar, E. Altman, Optimal dissemination of security patches in mobile wireless networks, IEEE Trans. Inf. Theory 58 (7) (2012) 4714–4732.
[22] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: Proc. of MobiCom, ACM, 1998, pp. 85–97.
[23] C.-C. Chiang, M. Gerla, On-demand multicast in mobile wireless networks, in: Proc. of ICNP, IEEE, 1998, pp. 262–270.
[24] J.J. Garcia-Luna-Aceves, M. Spohn, Source-tree routing in wireless networks, in: Proc. of ICNP, IEEE, 1999, pp. 273–282.
[25] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, Kluwer Int. Ser. Eng. Comput. Sci. (1996) 153–179.
[26] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, W. Comm. Mobile Comp. 2 (5) (2002) 483–502.
[27] M.C. Gonzalez, C.A. Hidalgo, A.-L. Barabasi, Understanding individual human mobility patterns, Nature 453 (7196) (2008) 779–782.
[28] J. Boleng, Normalizing mobility characteristics and enabling adaptive protocols for ad hoc networks, in: Proc. of LANMAN, 2001, pp. 9–12.
[29] K. Lee, S. Hong, S.J. Kim, I. Rhee, S. Chong, Slaw: a new mobility model for human walks, in: Proc. of INFOCOM, IEEE, 2009, pp. 855–863.
[30] C. Boldrini, A. Passarella, Hcmm: modelling spatial and temporal properties of human mobility driven by users social relationships, Comput. Commun. 33 (9) (2010) 1056–1074.
[31] S. Isaacman, R. Becker, R. Cáceres, M. Martonosi, J. Rowland, A. Varshavsky, W. Willinger, Human mobility modeling at metropolitan scales, in: Proc. of MobiSys, ACM, 2012, pp. 239–252.
[32] V. Capasso, G. Serio, A generalization of the Kermack–McKendrick deterministic epidemic model, Math. Biosci. 42 (1) (1978) 43–61.
[33] C.-Y. Huang, C.-T. Sun, H.-C. Lin, Influence of local information on social simulations in small-world network models, J. Artif. Soc. Soc. Simul. 8 (4) (2005) 8.
[34] R. Potharaju, Infection quarantining for wireless networks using power control, in: DSN Student Forum, 2010.
[35] C. de Waal, M. Gerharz, Bonnmotion: A mobility scenario generation and analysis tool, Communication Systems group, Institute of Computer Science IV, University of Bonn, Germany, 2003.
[36] The Network Simulator – NS-3. <http://www.nsnam.org>.
[37] C. Bettstetter, C. Wagner, et al., The spatial node distribution of the random waypoint mobility model, in: German Workshop on Mobile Ad Hoc Networks (WMAN), 2002, pp. 41–58.
[38] G. Resta, P. Santi, An analysis of the node spatial distribution of the random waypoint mobility model for ad hoc networks, in: Proc. of POMC, ACM, 2002, pp. 44–50.
[39] E. Hyytiä, J. Virtamo, Random waypoint mobility model in cellular networks, Wirel. Netw. 13 (2) (2007) 177–188.
[40] J. Levine, J.B. Grizzard, H.L. Owen, Detecting and categorizing kernel-level rootkits to aid future detection, IEEE Secur. Privacy 4 (1) (2006) 24–32.
[41] V.V. Vazirani, Approximation Algorithms, Springer-Verlag, New York, Inc., 2001.
[42] R. Bridson, Fast poisson disk sampling in arbitrary dimensions, in: ACM SIGGRAPH, vol. 2007, 2007.
[43] Two-way analysis of variance. <http://en.wikipedia.org/wiki/Two-way_analysis_of_variance>.
[44] C. Gkantsidis, P.R. Rodriguez, Network coding for large scale content distribution, Proc. of IEEE INFOCOM, Vol. 4, IEEE, 2005, pp. 2235–2245.
[45] D. Kienzle, M. Elder, Recent worms: a survey and trends, in: Proc. of WORM, ACM, 2003, pp. 1–10.
[46] J. Kephart, S. White, D. Chess, Computers and epidemiology, IEEE Spectrum 30 (5) (1993) 20–26.
[47] J. Kephart, S. White, Measuring and modeling computer virus prevalence, in: Proc. of IEEE S&P, IEEE Computer Society, 1993, p. 2.
[48] S. Staniford, V. Paxson, N. Weaver, How to own the internet in your spare time, in: Proc. of USENIX Security, vol. 8, 2002, pp. 149–167.

[49] G. Serazzi, S. Zanero, Computer virus propagation models, Perform. Tools Appl. Netw. Syst. (2004) 26–50.
[50] G. Kesidis, I. Hamadeh, S. Jiwasurat, Coupled Kermack–McKendrick models for randomly scanning and bandwidth-saturating internet worms, Qual. Ser. Multiserv. IP Netw. (2005) 101–109.
[51] R. Anderson, R. May, Infectious Diseases of Humans: Dynamics and Control, Oxford University Press, 1992.
[52] H. Andersson, T. Britton, Stochastic Epidemic Models and Their Statistical Analysis, Springer Verlag, 2000.
[53] Z. Chen, L. Gao, K. Kwiat, Modeling the spread of active worms, Proc. of INFOCOM, vol. 3, IEEE, 2003, pp. 1890–1900.
[54] G. Theodorakopoulos, J. Baras, J. Le Boudec, Dynamic network security deployment under partial information, in: Proc. of Allerton, 2008, pp. 261–267.
[55] J.W. Mickens, B.D. Noble, Modeling epidemic spreading in mobile environments, in: Proc. of ACM WiSe, ACM, 2005, pp. 77–86.
[56] S. Peng, G. Wang, S. Yu, Modeling the dynamics of worm propagation using two-dimensional cellular automata in smartphones, J. Comput. Syst. Sci. 79 (5) (2013) 586–595.
[57] W. Wang, I. Murynets, J. Bickford, C.V. Wart, G. Xu, What you see predicts what you get lightweight agent-based malware detection, Secur. Commun. Netw. 6 (1) (2013) 33–48.
[58] C. Szongott, B. Henne, M. Smith, Evaluating the threat of epidemic mobile malware, in: 20012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), IEEE, 2012, pp. 443–450.
[59] M.M. Williamson, Throttling viruses: restricting propagation to defeat malicious mobile code, in: Proc. of ACSAC, IEEE, 2002, pp. 61–68.
[60] C. Wong, C. Wang, D. Song, S. Bielski, G.R. Ganger, Dynamic quarantine of internet worms, in: Proc. of DSN, IEEE, 2004, pp. 73–82.
[61] R.G. Cole, Initial Studies on Worm Propagation in MANETs for Future Army Combat Systems, Tech. rep., DTIC Document, 2004.
[62] M.V. Barbera, S. Kosta, J. Stefa, P. Hui, A. Mei, Cloudshield: efficient anti-malware smartphone patching with a P2P network on the cloud, in: 2012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P), IEEE, 2012, pp. 50–56.
[63] J. Tang, H. Kim, C. Mascolo, M. Musolesi, Stop: socio-temporal opportunistic patching of short range mobile malware, in: 2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE, 2012, pp. 1–9.
[64] C.C. Zou, D. Towsley, W. Gong, Email worm modeling and defense, in: Proc. of ICCCN, IEEE, 2004, pp. 409–414.
[65] Y. Yang, S. Zhu, G. Cao, Improving sensor network immunity under worm attacks: a software diversity approach, in: Proc. of ACM MobiHoc, ACM, 2008, pp. 149–158.
[66] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, A. Nucci, A social network based patching scheme for worm containment in cellular networks, in: Handbook of Optimization in Complex Networks, Springer, 2012, pp. 505–533.

Endadul Hoque is a PhD student in the Computer Science Department at Purdue University under the supervision of Professor Cristina Nita-Rotaru. He obtained his MS in Computer Science from Marquette University, WI, in 2010 and BS in Computer Science and Engineering from Bangladesh University of Engineering & Technology (BUET), Bangladesh, in 2008. He is a member of the Dependable and Secure Distributed Systems Laboratory (DS2). His research interests include security in wireless networks and distributed systems. He is a member of the ACM and IEEE Computer Society.

Rahul Potharaju is an Applied Scientist in the Cloud and Information Services Lab at Microsoft. Before that, he obtained his PhD degree in Computer Science from Purdue University and Master's degree in Computer Science from Northwestern University. Rahul is passionate about transforming big data into actionable insights and building large-scale data-intensive systems, with a particular interest in analytics-as-a-service clouds and automated problem inference systems. He is a recipient of the Motorola Engineering Excellence award in 2009 and the Purdue Diamond Award in 2014. His research has been adopted by several business groups inside Microsoft and has won the Microsoft Trustworthy Reliability Computing Award in 2013.

Cristina Nita-Rotaru is an Associate Professor in the department of Computer Science at Purdue University. She leads the Dependable and Secure Distributed Systems Laboratory. She received BS and MS degrees from Politechnica University of Bucharest, Romania, in 1995 and 1996, and a PhD degree in Computer Science from The Johns Hopkins University in 2003. She served on the technical program committee of over 40 conference in networking, distributed systems, and security. She received the NSF CAREER award. Her research interests include security and fault-tolerance for distributed systems, and networks. She is a member of the ACM and IEEE Computer Society.

Saswati Sarkar received ME from the Electrical Communication Engineering Department at the Indian Institute of Science, Bangalore in 1996 and PhD from the Electrical and Computer Engineering Department at the University of Maryland, College Park, in 2000. She joined the Electrical and Systems Engineering Department at the University of Pennsylvania, Philadelphia as an Assistant Professor in 2000 where she is currently a Professor. She received the Motorola gold medal for the best masters student in the division of electrical sciences at the Indian Institute of Science and a National Science Foundation (NSF) Faculty Early Career Development Award in 2003. She was an associate editor of IEEE Transaction on Wireless Communications from 2001 to 2006, and is currently an associate editor of IEEE/ACM Transactions on Networks. Her research interests are in stochastic control, resource allocation, dynamic games and economics of networks.

Santosh S. Venkatesh (S'81–M'86) received the Ph.D. degree in electrical engineering from the California Institute of Technology, Pasadena, in 1986. He is a Professor with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia. His research interests are in probabilistic models, random graphs, epidemiology, and network and information theory.