Ad Hoc Networks 7 (2009) 1489–1500

Contents lists available at ScienceDirect

Ad Hoc Networks

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

Self-propagating mal-packets in wireless sensor networks: Dynamics and defense implications [☆]

Bo Sun [a,*], Guanhua Yan [b], Yang Xiao [c], T. Andrew Yang [d]

[a] Department of Computer Science, Lamar University, Beaumont, TX 77710, USA
[b] Information Sciences (CCS-3), Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[c] Department of Computer Science, University of Alabama, 101 Houser Hall, Box 870290, Tuscaloosa, AL 35487, USA
[d] Division of Computing and Mathematics, University of Houston – Clear Lake, Houston, TX 77058, USA

ARTICLE INFO

Article history:
Available online 12 April 2009

Keywords:
Wireless sensor networks
Mal-packets
Percolation
Propagation dynamics

ABSTRACT

Self-propagating mal-packets have become an emergent threat against information confidentiality, integrity, and service availability in wireless sensor networks. While playing an important role for people to interact with surrounding environment, wireless sensor networks suffer from growing security concerns posed by mal-packets because of sensor networks' low physical security, lack of resilience and robustness of underlying operating systems, and the ever-increasing complexity of deployed applications.

In this paper, we study the propagation of mal-packets in 802.15.4 based wireless sensor networks. Based on our proposed mal-packet self-propagation models, we use TOSSIM, a simulator for wireless sensor networks, to study their propagation dynamics. We also present a study of the feasibility of mal-packet defense in sensor networks. Specifically, we apply random graph theory and percolation theory to investigate the immunization of highly-connected nodes, i.e., nodes with high degrees of connectivity. Our goal is to partition the network into as many separate pieces as possible, thus preventing or slowing down the mal-packet propagation. We study the percolation thresholds of different network densities and the effectiveness of immunization in terms of connection ratio, remaining link ratio, and distribution of component sizes. We also present an analysis of the distribution of component sizes.

1. Introduction

Wireless sensor networks (WSNs) have become a promising technology which has the potential to change our everyday life. However, because of their typically weak physical security, lack of resilience and robustness of underlying operating systems [1], inadequacy of basic building blocks for reliable WSN software systems [4], and the ever-increasing complexity of deployed applications, new system vulnerabilities exploited by mal-packets will continue to plague WSNs. Once the mal-packets start spreading by exploiting the monoculture of WSN applications, manual human intervention is hardly effective based on the past experience gained from defending against Internet mal-packets such as worms [5]. Therefore, self-propagating mal-packets have become an emergent threat against information confidentiality, integrity, and service availability for WSNs.

New security concerns about WSN mal-packet propagation are hard to exaggerate. The weak physical security makes sensor nodes easily compromised, thus rendering embedded secrets open to attackers and various keying

[☆] This research was supported in part by the US National Science Foundation (NSF) under Grants DUE-0633445 (Bo Sun), CCF-0829827, CNS-0716211, and CNS-0737325 (Yang Xiao) and DUE-0633469 (T. Andrew Yang).
[*] Corresponding author.
E-mail addresses: bsun@my.lamar.edu (B. Sun), ghyan@lanl.gov (G. Yan), yangxiao@ieee.org (Y. Xiao), yang@uhcl.edu (T. Andrew Yang).

1570-8705/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.adhoc.2009.04.003

mechanisms helpless. Recently, emerging viruses, like *Cabir* [3], have demonstrated the ability to spread over the air interfaces, making it possible to develop worms replicating among sensor nodes. In [15], it has been demonstrated that self-propagating mal-packets can exploit memory-related vulnerabilities to propagate itself, thus taking over the whole network. Therefore, it is anticipated that WSNs will be plagued by mal-packets designed in the future. This is especially true when WSNs are increasingly deployed in critical infrastructures.

Motivated by these concerns, we aim at studying mal-packet propagation dynamics and defense in the context of 802.15.4 [20] WSNs. We illustrate how a compromised node can obtain an unfair bandwidth share by not adhering to the carrier sense multiple access-collision avoidance (CSMA-CA) protocol of 802.15.4, thus facilitating the fast spread of mal-packets. We then use TOSSIM [16], a simulator for wireless sensor networks, to study the mal-packet propagation dynamics. TOSSIM captures the behavior and interactions of networks of thousands of TinyOS [18] motes at network bit granularity. It can generate discrete-event simulations directly from TinyOS component graphs. Therefore, TOSSIM exploits the WSN domain and TinyOS design, and is thus suitable for our research on mal-packet propagation in WSNs. We study the unicast and broadcast mal-packet propagation models and present their dynamics.

Most existing work only focus on mal-packet propagation dynamics in sensor networks [11,12,15]. Few works consider how to prevent such kind of propagation. Therefore, based on percolation theory and random graph theory [14], we further study the feasibility of defending against mal-packet propagation in WSNs. Specifically, we model the deployment of sensor nodes as a homogeneous spatial Poisson process in a two-dimensional space and study the effectiveness of immunizing some sensor nodes in order to protect WSNs. Immunizing a node means that the node cannot be infected by the self-propagating mal-packets. Therefore, our purpose is to know how to choose an appropriate set of nodes in order to partition the WSN into as many separate pieces as possible, therefore preventing or slowing down the mal-packet propagation. Percolation theory [14] tells us that with the increase of the number of immunized nodes, there exists a “critical phenomenon” at which the network suddenly becomes disintegrated. Intuitively, we select the most connected nodes to immunize. We study the impact of the selective immunization on the network topology in terms of *connection ratio*, *remaining link ratio*, and the *distribution of component sizes*. Simulation results demonstrate that the selective immunization can effectively prevent or slow down the large-scale outbreaks of mal-packets.

Our proposed partition based approach enables us to decide the critical locations at which sensor nodes should be immunized. After collecting these locations, some further actions can be taken to improve WSNs' immunity against mal-packet propagation. For example, *survivability through heterogeneity* philosophy [21] can be adopted through software or hardware heterogeneity. Different versions of software or hardware implementing the same functionality will not suffer from the same vulnerability exploited by attackers. Therefore, sensor nodes deployed at these critical locations may be installed with an appropriate different version of software or hardware.

The rest of the paper is organized as follows. Section 2 briefly introduces 802.15.4, an IEEE standard for low-rate wireless personal area networks. Section 3 discusses our proposed mal-packet propagation model in WSNs. Section 4 presents our simulation results to evaluate mal-packet propagation dynamics. In Section 5, we present the network model and methodology which we use to immunize WSNs in order to prevent or slow down mal-packet propagation. Section 6 presents related work and Section 7 concludes this paper.

2. 802.15.4 Primer

The 802.15.4 standard [20] was designed to support wireless radios and protocols for low-power devices, such as wireless sensor networks. In the physical layer, the standard encompasses one channel in the 868 MHz band with a data rate of 20 kbps, 10 channels in the 915 MHz band with a data rate of 40 kbps, and 16 channels in the 2.4 GHz band with a data rate of 250 kbps.

An 802.15.4 network can work either in a *beacon-enabled* or a *nonbeacon-enabled* mode. In a beacon-enabled mode, a *network coordinator* transmits beacons periodically for synchronization and a slotted CSMA-CA mechanism is used to transmit data frames. Whenever a device wishes to transmit data frames during a contention access period, the device needs to locate the boundary of the next backoff slot and then waits for a random number of backoff slots. If the channel is busy after this random backoff, the device needs to wait for another random number of backoff slots before trying to access the channel again. If the channel is idle, the device can begin transmitting on the next available backoff slot boundary. In a nonbeacon-enabled mode, unslotted CSMA-CA is used for the transmission of data frames. In this case, if a device wants to transmit data frames, it needs to wait for a random period. If the channel is found idle after the random backoff, the device will transmit the data. Otherwise, the device needs to wait for another random period before trying to access the channel again.

The 802.15.4 specification also specifies default values used in backoff mechanisms. For example, in unslotted CSMA-CA, a node which has a packet ready to send first backs off for a random number of time between 0 and $2^{BE} - 1$, where *BE* is set to 3 by default. If the channel is found to be busy again after the random backoff, *BE* increases by 1. This process is repeated until either *BE* equals *aMaxBE* (which has a default value of 5), at which point *BE* is frozen at *aMaxBE*, or until a certain maximum number of permitted random backoff stages, denoted as *macMaxCSMABackoffs*, is reached, at which point an access failure is declared to the upper layer. The standard sets *macMaxCSMABackoffs* to 5 by default [20].

3. Mal-packet propagation model

In this section, we present the design of a baseline WSN mal-packet propagation model. We assume that one sensor node is compromised by a mal-packet and this infected sensor node attempts to replicate this mal-packet to those sensor nodes within its transmission range. In this paper, we only focus on medium access control (MAC) layer propagation dynamics and thus ignore infecting sensor nodes multiple hops away. The one-hop mal-packet propagation in WSNs represents one of the key differences from Internet worms.

In this paper, we focus on the mal-packet study in an *nonbeacon-enabled* mode. We illustrate how a compromised node can obtain an unfair bandwidth share by not adhering to 802.15.4, thus facilitating the fast spread of mal-packets. Our proposed mal-packet model can also be applied to *beacon-enabled* mode after slight modifications.

After a mal-packet has established a presence, one of its basic goals is to spread itself quickly to other vulnerable nodes. Therefore, an infected node may use selfish strategies to obtain an unfair share of the channel. Taking IEEE 802.15.4 unslotted CSMA-CA mechanisms as an example, this may include:

1. Select a smaller average backoff value, as specified by the parameter *wormBE*.
2. Use a different retransmission strategy that does not increase *BE* after collision.
3. Increase the maximum possible value for *NB*, *worm_macMaxCSMABackoff*, so a node may have more chances to compete for channels.

After we consider these factors, the modified unslotted CSMA-CA for the propagation of a WSN mal-packet is illustrated in Fig. 1.

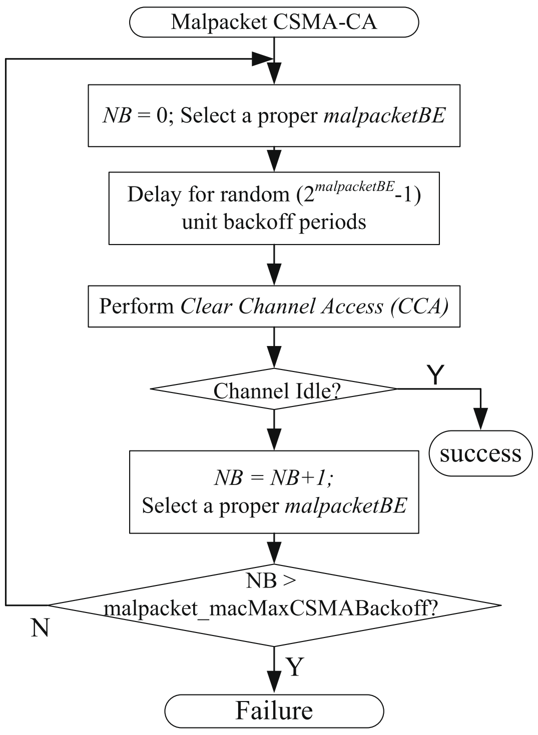

Malpacket CSMA-CA
NB = 0; Select a proper malpacketBE
Delay for random (2^malpacketBE-1) unit backoff periods
Perform Clear Channel Access (CCA)
Channel Idle?
Y
success
NB = NB+1; Select a proper malpacketBE
NB > malpacket_macMaxCSMABackoff?
N
Y
Failure

Fig. 1. Unslotted CSMA-CA for WSN Mal-packets in 802.15.4 *nonbeacon-enabled* mode.

Depending on the requirements of different applications, we propose two types of mal-packet models: *unicast* WSN mal-packet model and *broadcast* WSN mal-packet model. For both types of mal-packets, when a WSN mal-packet is activated, the mal-packet starts spreading itself in its vicinity. In the unicast mal-packet model, the infected sensor node can only unicast the mal-packet to one of its neighbors at a time. For example, a neighbor discovery protocol enables the compromised node *A* to keep the list of one-hop neighbors. When node *A* wants to spread the mal-packet, *A* picks up one neighbor, constructs the mal-packet, and sends it out. In the broadcast mal-packet model, by contrast, the infected sensor node *A* constructs a broadcast packet and sends it out.

Extensive research work has been devoted to pairwise key establishment mechanisms in WSNs, in which communication between nodes are secured by various pairwise keying mechanisms. A unicast model is necessary for a mal-packet to spread in environment where pairwise keying mechanisms are adopted. Correspondingly, for applications which are protected by groupwise keying schemes, a broadcast mal-packet model can be used.

When a node (say, *A*) propagating mal-packets sends a mal-packet to its neighbor (say, *B*), there exist three possible *states* with respect to how the recipient may respond:

1. *Invulnerable*: An *invulnerable* sensor node *B* is not vulnerable to propagated mal-packets. When node *B* receives a mal-packet, node *B* silently drops the data packet that contains the malicious code.
2. *Vulnerable and infected*: A *vulnerable and infected* sensor node *B* is vulnerable and has been infected with propagated mal-packets. When node *B* receives a mal-packet from node *A*, node *B* may send back an INFECTED packet to node *A*, informing node *A* that node *B* has already been infected. In this case, node *A* will update its neighbor information correspondingly.
3. *Vulnerable and uninfected*: A *vulnerable and uninfected* sensor node *B* is vulnerable to propagated mal-packets but has not been infected. When node *B* receives a mal-packet from node *A*, node *B* sends back a SUCCESS packet to node *A*, informing node *A* that node *B* is now infected with the mal-packet.

This is illustrated in Fig. 2.

3.1. Unicast Mal-packet model

In the unicast mal-packet model, we assume that a neighbor discovery protocol is used for one node to discover its one-hop neighbors. This is a reasonable assumption given that most applications and upper layer protocols in WSNs require neighborhood information. Therefore, each node can collect a list of neighbors within its radio range.

Each node may start replicating mal-packets based on the state of its neighbors. For situations illustrated in Fig. 2b and c, on the arrival of an INFECTED or SUCCESS response packet, node *A* removes the victim node *B* from the neighbor list and attempts to infect the next one.

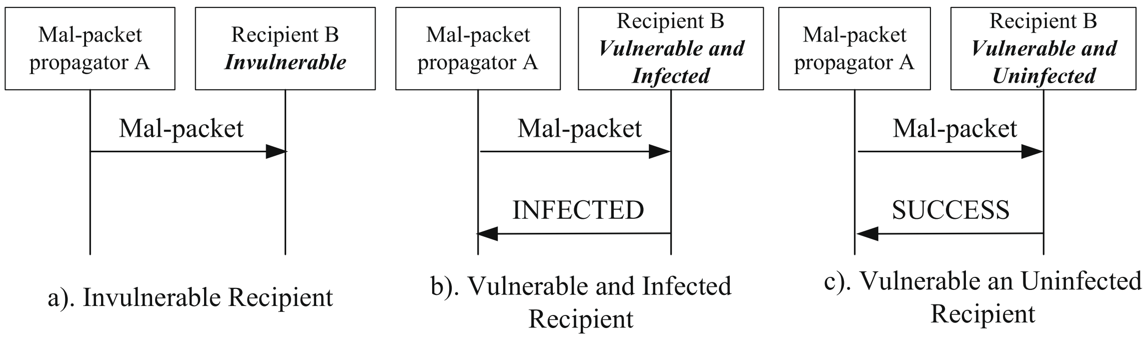

Mal-packet propagator A
Recipient B *Invulnerable*
Mal-packet
a). Invulnerable Recipient
Mal-packet propagator A
Recipient B *Vulnerable and Infected*
Mal-packet
INFECTED
b). Vulnerable and Infected Recipient
Mal-packet propagator A
Recipient B *Vulnerable and Uninfected*
Mal-packet
SUCCESS
c). Vulnerable an Uninfected Recipient

Fig. 2. Neighbor feedback.

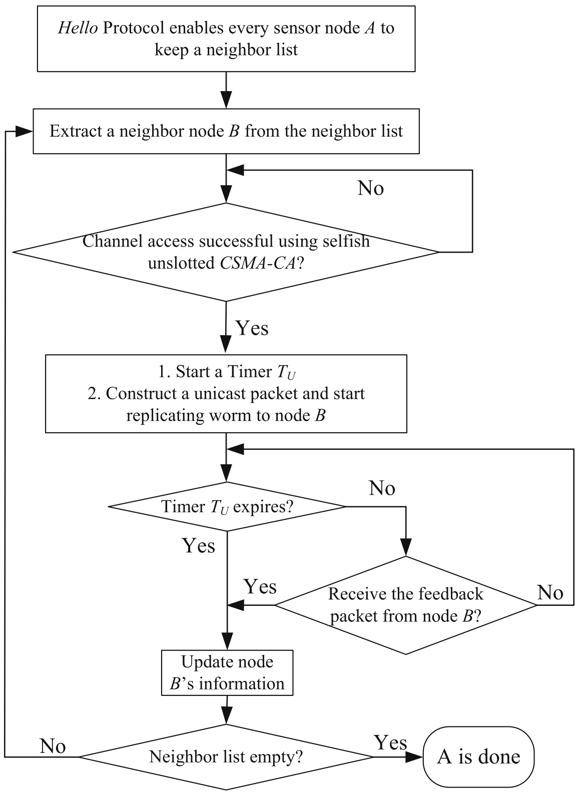
The sending node A may also need to set up a timer T_U, which expires after T_U time units if no response is received. In situations illustrated in Fig. 2.a, node A may not receive any feedback from node B. The mal-packet and feedback packet from node B may also be lost in Fig. 2b and c. When T_U expires, node A may update the information about node B in its neighbor list based on the response packet from node B. A proper T_U should give node A sufficient time to wait for the feedback from node B. If no response is received, node A may mark that node B as *invulnerable* and attempt to infect the next neighbor.

Because of the existence of links with asymmetric transmission delays [23], the value of T_U may be set to the summary of forward and reverse per-hop round trip time of a link. The forward per-hop round trip time, d_f, is the time that it takes for a packet to arrive at the recipient of the link; the reverse per-hop round trip time, d_r, is the time that it takes for the feedback packet to arrive at the sender. The value of T_U may be set to $d_f + d_r$.

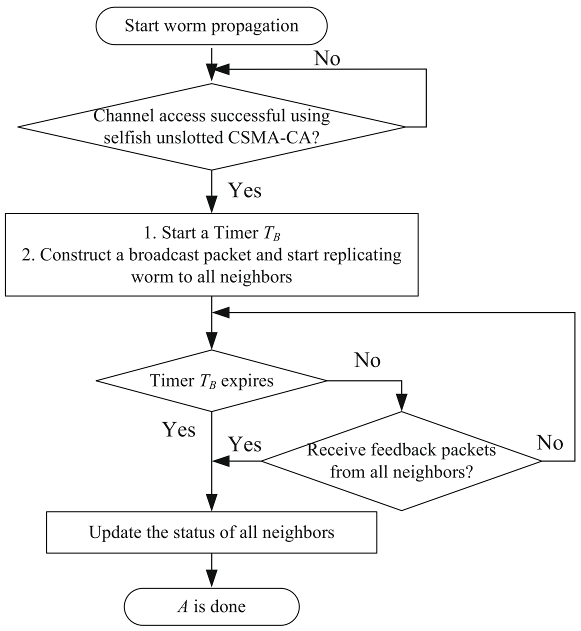
After we consider all these, the infection cycle of a unicast mal-packet is illustrated in Fig. 3a.

3.2. Broadcast Mal-packet Model

A corresponding modification is the broadcast mal-packet model, in which the mal-packet can broadcast in


Hello Protocol enables every sensor node *A* to keep a neighbor list
Extract a neighbor node *B* from the neighbor list
No
Channel access successful using selfish unslotted *CSMA-CA*?
Yes
1. Start a Timer T_U
2. Construct a unicast packet and start replicating worm to node *B*
Timer T_U expires?
No
Yes
Yes
Receive the feedback packet from node *B*?
No
Update node *B*'s information
No
Neighbor list empty?
Yes
A is done

(a) Unicast Mal-packet Infection.


Start worm propagation
No
Channel access successful using selfish unslotted CSMA-CA?
Yes
1. Start a Timer T_B
2. Construct a broadcast packet and start replicating worm to all neighbors
Timer T_B expires
No
Yes
Yes
Receive feedback packets from all neighbors?
No
Update the status of all neighbors
A is done

(b) Broadcast Mal-packet Infection.

Fig. 3. Mal-packet propagation model in WSNs.

one-hop to its neighbors. Every neighbor may generate a corresponding reply based on schemes illustrated in Fig. 2. Based on Fig. 2, node *A* may modify the status of its neighbors correspondingly.

The broadcast mal-packet model is illustrated in Fig. 3b. In Fig. 3b, timer T_B gives node *A* sufficient time to wait for feedback from all neighbors. Following the notation used in Section 3.1, the value of T_B for one node may be set to the largest per-hop round trip time of a link among all neighbors.

The neighbor list is a common component in many existing WSN protocols aiming at improving protocol performance [24]. Therefore, the compromised node *A* may utilize the existing neighbor list in the broadcast mal-packet model to obtain which neighbors are compromised. Because the neighbor list is already existent in node *A*, this maintenance of this list will not incur extra costs to mal-packet propagation. Moreover, a neighbor list is also desirable for secure group broadcasting [27]. In protocols where no neighbor list is kept, timer T_B and the neighbor list may be removed in order to reduce mal-packet propagation costs.

The infection cycle of a broadcast mal-packet is illustrated in Fig. 3b.

4. Mal-packet propagation dynamics

Unfortunately, the 802.15.4 standard has not been fully supported in TinyOS 2.x. Instead, a basic CSMA-CA algorithm is adopted (See *TossimPacketModelC.nc* for the implementation under TOSSIM). Its backoff mechanism is similar to those described in 802.15.4. Specifically, two important parameters play the same role as *NB* and *BE*. The *max_iterations()* value in TOSSIM denotes the parameter *macMaxCSMABackoff* in 802.15.4, while the *init_low()* value denotes the lower bound of the backoff range. Therefore, based on TOSSIM, we observe the impacts of these two parameters on mal-packet propagations.

We adopt the default signal-strength based radio model used by TOSSIM [16]. This radio model is based on CC2420 radio, which is used by popular wireless sensor modules like MicaZ, TelosB, and IMote2 [19]. We adopt the noise trace provided by TOSSIM from Meyer Library at Stanford University as the noise trace.

We randomly pick one node as an *infected* node. All other nodes are set to *vulnerable and uninfected*. Before the mal-packet begins propagation, every node uses a *Hello* protocol to discover its neighbors. To measure the propagation dynamics of mal-packets, we set up three scenarios, including the uniform distribution of 700 nodes, 800 nodes, and 900 nodes in a 400 m $\times$ 400 m area. We set the transmission range of each node to 25 m. Based on the locations of sensor nodes and the transmission range, radio connectivity data which defines the propagation gain when one sending node transmits to the receiving node are created. These radio connectivity data are then provided to TOSSIM running scripts. The propagation dynamics are measured against various variables, including the impact of *max_iterations*, the impact of *init_low*, the impact of network density, the ratio of packet loss, and the average number of devices infected per second during the attack.

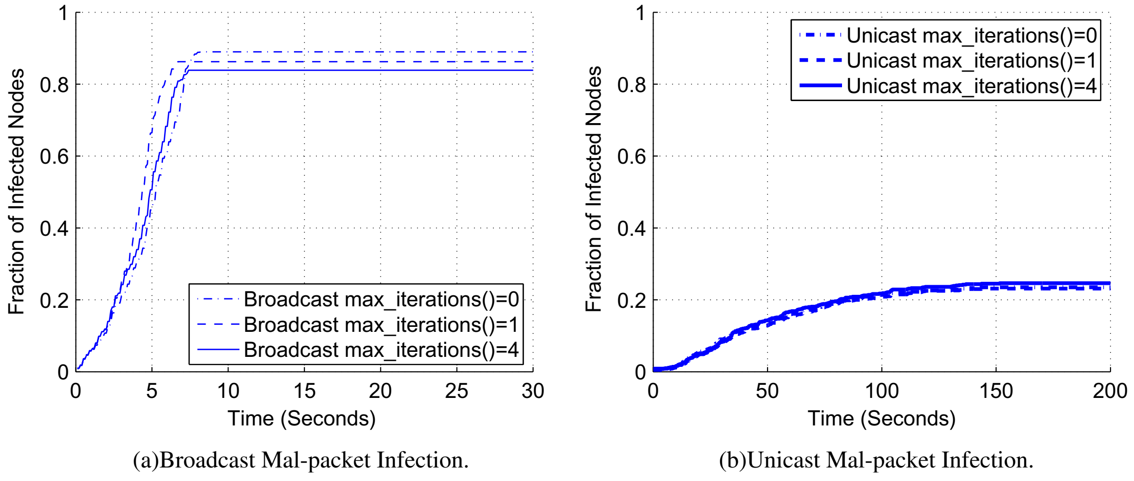
4.1. Impacts of max_iterations()

In this set of simulation runs, we use the normal backoff values, and present the propagation dynamics under different *max_iterations()* values for the 800 nodes in Fig. 4.

First, it is obvious that the broadcast propagation is much faster than that of the unicast. Also, broadcast propagation may infect many more nodes. The results are what we expect. Unicast propagation needs to transmit more mal-packets. However, the low transmission rates in WSNs may introduce more packet collisions. This will greatly lead to the situation where more mal-packets are dropped.

Second, even for broadcast propagation, it is still difficult to reach a 100% infection. WSNs suffer from very low transmission speed and unreliable transmission links. This may make packets prone to being dropped.

Third, given the broadcast propagation, the larger the value of *max_iterations()*, the lower the infection rate. Note



(a)Broadcast Mal-packet Infection. (b)Unicast Mal-packet Infection.

Fig. 4. Impact of *max_iterations()*.

that in TOSSIM, an 0 value of *max_iterations()* means infinite. This is because a larger value of *max_iterations()* could give the nodes more chances to exploit the channel, thus increasing the probability of successful transmissions of mal-packets.

Fourth, as indicated in Fig. 4b, we also observe that the value of *max_iterations()* does not have an obvious impact on the infection speed and the fraction of infected nodes in unicast mal-packet propagation. This is because in unicast propagation, the transmitted number of mal-packets is much larger than that in the broadcast propagation. Therefore, mal-packets may have a much higher probability of collisions in the unicast propagation. Furthermore, because of the low transmission data rate of WSNs, if *CCA* detects a *busy* channel, the next assessment by *CCA* is very likely to detect a *busy* channel. Based on our proposed unicast model illustrated in Fig. 3a, the value of *max_iterations()* affects the propagation speed to a lesser degree.

4.2. Impact of *init_low()*

In this set of simulation runs, we set the value of *max_iterations()* to 0, and present the propagation dynamics under different backoff values for the 800 nodes in Fig. 5. In Fig. 5, when the value of *init_low()* is 1, it means that the backoff value is set to $1 * init\_low()$. Similarly, an *init_low()* of 3 means a backoff value of $3 * init\_low()$ is adopted.

Besides similar observations presented in Fig. 4, we can also see that, for the broadcast propagation, a smaller value of *init_low()* may make the propagation faster than that of a larger value of *init_low()*. A smaller *init_low()* value decreases the random waiting time when sensor nodes perform the backoff, thus speeding up the mal-packet propagation.

As indicated in Fig. 5b, we also observe that the value of *init_low()* does not have an obvious impact on the infection speed and the fraction of infected nodes in unicast mal-packet propagation. The reason is similar to that of Fig. 4b. Mal-packets may have a much higher probability of collisions in the unicast propagation. Furthermore, because of the low data rate of WSNs, if *CCA* detects a *busy* channel, the next assessment by *CCA* is very likely to detect a *busy* channel even if a different value of backoff is adopted. Based on our proposed unicast model illustrated in Fig. 3a, the value of *init_low()* affects the propagation speed to a lesser degree.

4.3. Impact of network density

In this set of simulation runs, we set *max_iterations()* and *init_low()* to default values and measure the impact of network density on the propagation dynamics, as illustrated in Fig. 6.

From Fig. 6a, we observe in the broadcast propagation that the denser the network, the higher the percentage of nodes is infected. Denser networks make nodes have higher node degrees and higher connectivity. This can facilitate the mal-packet propagation.

From Fig. 6b, we observe in the unicast propagation that different network density does not have an obvious impact on the infection speed and the fraction of infected nodes. The reason is similar to that of Fig. 4b. Mal-packets may have a much higher probability of collisions in the unicast propagation. Furthermore, because of the low data rate of WSNs, if *CCA* detects a *busy* channel, the next assessment by *CCA* is very likely to detect a *busy* channel. A node with higher degree of connectivity needs to transmit more mal-packets to infect its neighbors. However, many of these transmissions will likely to fail because of the busy channels detected by *CCA* and unpredictable channel conditions in WSNs. Therefore, relatively different network density affects the propagation speed to a lesser degree. We, however, do not expect such an observation from Fig. 6b holds for all network densities. For example, mal-packets in an extremely sparse network may not be able to spread at all.

We also observe that in Figs. 4–6, the unicast model may not reach 100% infection of all network nodes. In fact, the infection ratio of unicast model is much smaller than

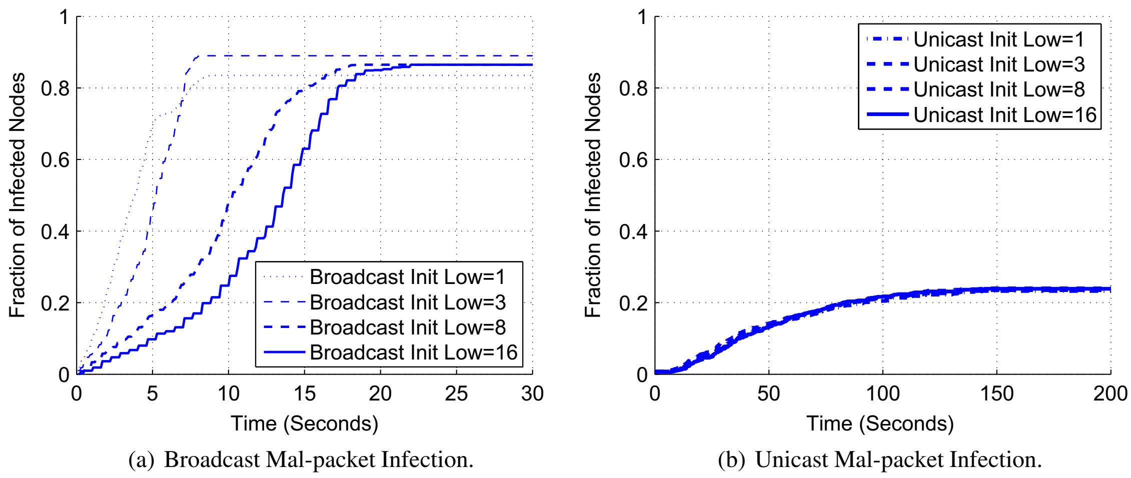


Fig. 5. Impact of *init_low()*.

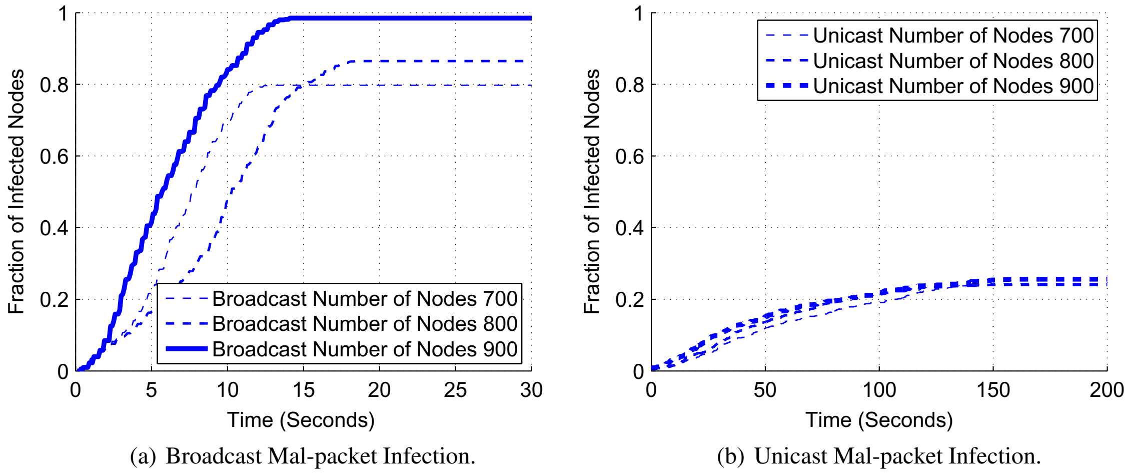


Fig. 6. Impact of network density.

that of the broadcast model. This is because in unicast models, a much larger number of mal-packets are transmitted compared to that of broadcast models. These many more mal-packets may lead to severe packet collisions considering the low transmission data rate in WSNs. Therefore, as the infection progresses, too many collisions will prevent the compromised nodes from transmitting mal-packets.

4.4. Packet loss ratio

To further understand the impact of packet loss on worm propagation, we perform another set of simulations. In this set of simulation runs, 100 sensor nodes are randomly deployed in a $70 \times 70\,\text{m}^2$ square area. We set *max_iterations()* and *init_low()* to default values and measure MAC layers packet collisions. We disable the MAC layer packet loss functions, and plot the curve in Fig. 7. This can help us further understand the propagation dynamics illustrated in previous sections.

As we expect, after we disable the packet collision functionality at the MAC layer, more nodes are compromised.

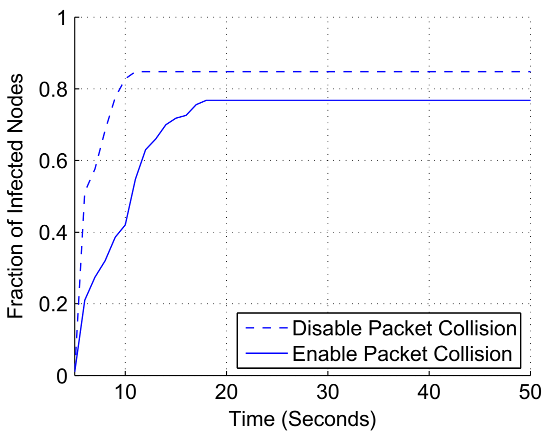


Fig. 7. Propagation curve considering packet collision.

However, it is still difficult to achieve 100% inflection. This is because we only disable packet collisions at the MAC layer. The complexities of low-power wireless networking may make WSNs suffer packet losses due to a variety of other reasons. For example, multi-path signal delivery may constitute grey region, in which node reception rates vary dramatically. Furthermore, radio irregularity and asymmetry may also have impact on higher layer protocols and lead to packet loss [7]. As indicated in Fig. [22], radio irregularity may result in asymmetric links and hence, it may have an adverse impact on protocols that use path-reversal techniques and neighbor discovery techniques.

4.5. Modeling considerations

There have been many research efforts devoted to the modeling of Internet worms. An accurate model can provide insight for detection and defense. In this section, we answer this question: *whether existing Internet worm propagation models can be directly applied to WSN worm propagation?*

In [8], a logistic model is proposed as a general model to analyze the propagation of worms, such as *Code Red I*:

$$N(t) = N_{dev} * \frac{e^{K(t-T)}}{1 + e^{K(t-T)}}, \qquad (1)$$

where $N(t)$ is the number of infected nodes at time t. N_{dev} denotes the total number of devices in the network. K and T are two parameters used in the model.

Because in our case, only neighbor list information is available, an infected node can only infect its neighbors. Therefore, we set the initial point $N(5)$ to 1 (recall that the first worm instance starts spreading itself at simulation time 5) and derive the inflection point from the simulation results [9]. We assume that the model takes the same amount of time to infect half of the vulnerable population as in the simulation results. We derive K and T based on these two points. The figure illustrating the comparison between the derived logistic model and the simulated curve is plotted in Fig. 8.

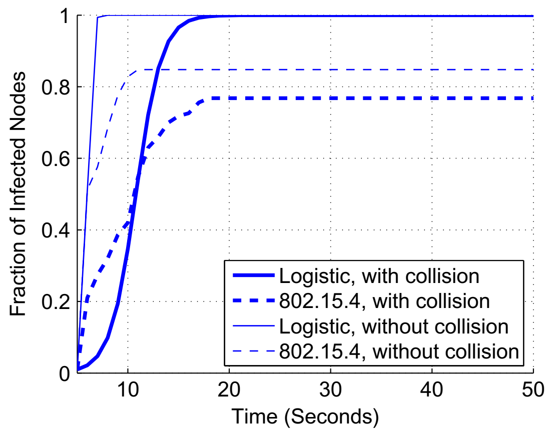


Fig. 8. Compare with logistic model.

It is clear that in the early phase of propagation, the logistic model underestimates the model. This is because at the early phase, the infected node may pick up a neighbor, and start replicating the mal-packet. While in the model illustrated in Eq. (1), there is an infection phase. This may slow down the mal-packet propagation.

In the late phase, the logistic model overestimates the mal-packet propagation. This is because the model in Eq. (1) does not consider the congestion and packet losses, which can slow down mal-packet propagation. Also, the logistic model predict a 100% infection ratio, which is not the case in our simulation.

Based on Fig. 8, one natural question is whether disabling packet losses could lead to a better model fitting. To answer this question, we disable packet collision, and plot the fraction of infected nodes and the fitted logistic curve, which is also illustrated in Fig. 8. As we can see, although both curves move left relatively, the logistic curve still overestimates the propagation speed at a late phase. At the early stage, both curves illustrate a very fast propagation.

4.6. Average number of devices infected per second

We plot the average number of infected nodes per second in Fig. 9. We can see that as more sensor nodes are compromised, the average number of infected nodes is decreasing. This can slow down the mal-packet propagation.

This also helps explain the difference demonstrated in Fig. 8. The model illustrated in Eq. (1) assumes the same discovery ratio every time, which is not the case in our situation.

5. Immunization-based mal-packet defense

We present an immunization-based mal-packet defense mechanism in this section. The following notations in Table 1 are used throughout the rest of the paper.

5.1. Network model as random graph

We use a homogeneous Poisson point process [2] to model the distribution of sensor nodes. We assume that each node has a communication range of radius r. Therefore, two nodes are linked together if and only if they are not farther apart than a certain threshold. Consider N nodes that are uniformly distributed in a square area with side length of X. Therefore, for a node A, the number of nodes falling inside the circle around A, i.e., the number of neighbors of node A, is equal to $r^2\lambda$, where λ is the network density and is equal to $\frac{N}{X^2}$.

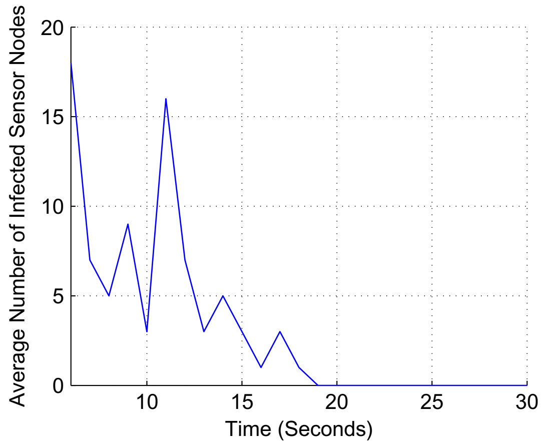


Fig. 9. Average number of infected sensor nodes per second as simulation advances.

Table 1
Notations.

Symbol	Meaning
N	Number of sensor nodes deployed
r	Communication range of sensor nodes
λ	Node density
p_k	The probability that a randomly picked node has k neighbors

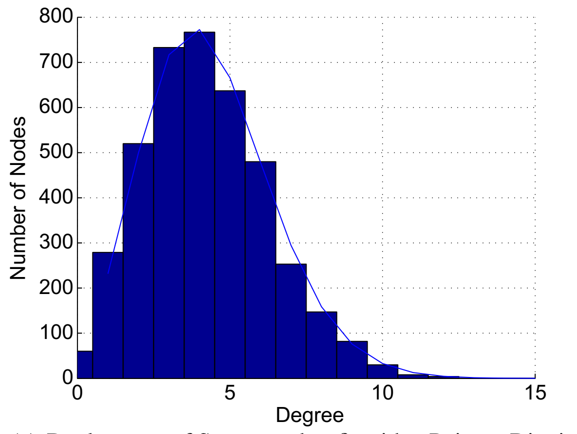
Under this assumption, we can use a random graph to model the deployment of sensor nods. p_k, the probability that a randomly picked node has k neighbors, is then equal to $\frac{\lambda^k}{k!}e^{-\lambda}, k = 0, 1, 2, \ldots$.

We perform a simple simulation to demonstrate this match. We simulate the deployment of 4000 nodes in a square area of 1250 by 1250 m^2. Each node has a transmission radius of 25. We plot the degree distribution of these sensor nodes, as illustrated in Fig. 10a. In Fig. 10a, the solid line indicates the theoretical Poisson distribution.

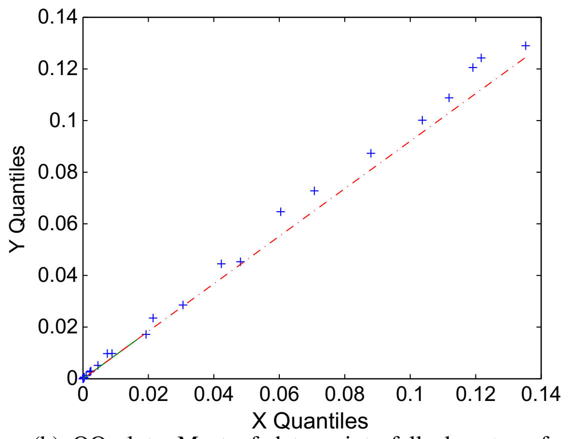
Based on this simulation, we can calculate p_k. To illustrate whether p_k follows a Poisson distribution, we make a Quantile–Quantile plot (Q–Q plot) between a theoretical Poisson distribution and p_k, as illustrated in Fig. 10b. As seen in Fig. 10b, most of data points fall almost perfectly along the line, which is a good indicator that p_k is Poisson distributed. The parameter for the Poisson distribution can be estimated as $\lambda = 5.2160$. Therefore, in the following, we assume that p_k follows a Poisson distribution and perform the analysis in 5.2.

5.2. Effect of selective immunization

Our basic idea is to select an appropriate set of nodes to immunize. By immunization, we mean that these nodes



(a) Deployment of Sensor nodes, fit with a Poisson Distribution.



(b) QQ-plot: Most of data points fall almost perfectly along the line, which is a good indicator that the degree is Poisson-distributed.

Fig. 10. Node degree distribution.

are immune to the propagated mal-packets. In this way, the selected node can help disintegrate the network, thus preventing or slowing down the large-scale propagation of mal-packets. In practice, after we identify this set of nodes, we can take various approaches to immunize. Therefore, our question becomes *how to choose the appropriate set of nodes in order to partition the WSN network into as many separate pieces as possible*.

It is not a good idea to immunize all the sensor nodes. We expect that different types of sensor nodes may be deployed to avoid monoculture of the underlying WSN hardware, thus realizing immunization. For a WSN consisting of Mica motes from *CrossBow* Corporation [19], the more powerful IMote2 motes can be deployed at critical locations for immunization purposes. Due to cost concerns, a large-scale WSN consisting of only IMote2 motes is much more expensive than a WSN of the same size with most of its nodes being the low-cost Mica motes. As a different example, for applications which are not time-sensitive, we can deploy sensor nodes at these critical locations controlling and intentionally delaying traffic flows in WSNs. These sensor nodes may deploy a different yet more expensive version of software. It is thus desirable to immunize a selective set of sensor nodes due to implementation costs.

Intuitively, the immunization of those most highly-connected nodes can help slow down the propagation of mal-packets because more connections between nodes are removed under this situation. Therefore, in the following, we conduct a series of simulations to demonstrate the effect of selective immunization.

We simulate the deployment of 5,000 nodes over square areas of $1250 \times 1250\,\text{m}^2$, $1300 \times 1300\,\text{m}^2$, and $1350 \times 1350\,\text{m}^2$, respectively. This deployment can lead to reasonable network density. A sparse deployment which leads to network disconnected can prevent the large-scale infection of mal-packets itself. On the other hand, if the network is densely deployed, for example, every sensor node has a connection to almost all the other nodes. It is thus helpless to immunize only a portion of nodes.

We use the following metrics to measure the effect of immunization:

1. *Connection ratio C_p*: The ratio of the size of the largest component to the size of the remaining network when we remove the top p percent most connected nodes (and their related edges);
2. *Remaining link ratio L_p*: The fraction of remained links after we remove the top p percent of most connected nodes;
3. *Distribution of component sizes*: The distribution of the sizes of connected components in terms of nodes. Here, a *component* is a subset of vertices in the graph each of which is reachable from the other through some path.

These metrics can effectively measure the impacts of selective immunization on mal-packet propagation. C_p reflects the size of the largest component, which gives the best situation the infection can reach. L_p indicates the fraction of remained links after selective immunization. Intuitively, the smaller the value of L_p, the slower the mal-packet can spread across the network. *Distribution of component sizes* gives the distribution of the sizes of connected components. The different sizes of the component from which the infected node starts infection may limit the total number of sensor nodes that this infection will infect. In case the initial infected node starts infection from within the largest remaining component, the size of the largest remaining component gives the best situation the infection can reach.

5.2.1. Connection Ratio

Simulation results of the connection ratio are illustrated in Fig. 11. We have the following observations.

First, given a fixed immunization rate, the sparser the network, the smaller the largest remaining component becomes. This is what we expect.

Second, there exists a percolation threshold [14] for these simulations, where C_p drops dramatically when the immunization rate exceeds the threshold. For example,

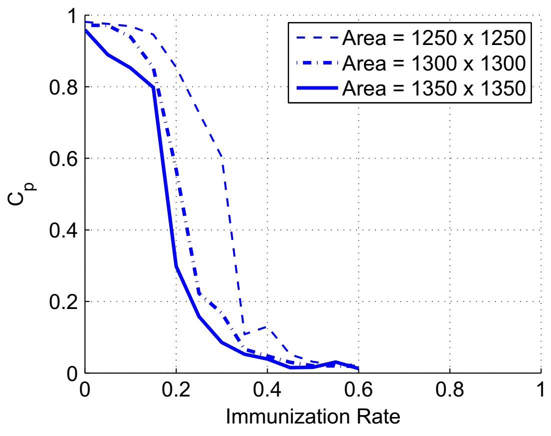


Fig. 11. Connection ratio.

for the area 1350 $\times$ 1350 m^2, the threshold is roughly 0.22. We can see that the sparser the network, the smaller the percolation threshold becomes. This indicates that fewer nodes are needed to be immunized in order to disconnect the network.

We also observe a dramatic decrease of the largest component when the immunization rate reaches some value. For example, given the 1300 $\times$ 1300 area, when the immunization rate reaches 30%, its largest component size drops dramatically. This phenomenon, which has been extensively studied in random graphs, is called percolation [14]. Therefore, in Section 5.2.3, we further extend our study when 30% nodes are removed.

5.2.2. Remaining link ratio

Simulation results of the remaining link ratio are illustrated in Fig. 12. We observe a slight decrease of the remaining link ratio when the network becomes sparser. This is also what we expect.

Based on Fig. 12, we can see that the remaining link ratio L_p decreases dramatically with the increase of immunization rate. A smaller value of L_p indicates a slower mal-packet propagation. This is also what we expect.

5.2.3. Distribution of component sizes

Simulation results of the distribution of component sizes are illustrated in Fig. 13. We can see that when the network is sparser, after the removal of top 30% most connected nodes, there are more small components and fewer large components. Smaller components can slow down or prevent the propagation of mal-packets. This is also what we expect.

5.3. Analysis of distribution of component sizes

The distribution of component sizes is an important indication of mal-packet propagation. In this section, with the help of random graph theory [14], we present the theoretical analysis of the distribution of component sizes.

First, we need to study the probability distribution of the number of second neighbors of one node, say node A, in one graph. To do this, we use q_k to denote the normalized distribution of the number of edges k emanating from vertex B other than the edge AB. Based on [14], we have

$$q_k = \frac{(k+1)p_{k+1}}{\sum_j jp_j}, \tag{2}$$

where we recall p_j gives the probability that a randomly picked node has k neighbors. Here, $\sum_j jp_j$ is the average degree of a vertex and its purpose is for normalization [14].

We further define the *probability generating function* for p_k as $G_0(x) = \sum_{k=0}^{\infty} p_k x^k$. The generating function for q_k is defined as $G_1(x) = \frac{G_0'(x)}{z}$ [14], where z denotes the mean number of neighbors of a randomly chosen vertex.

Randomly pick one edge in a graph, for example, edge AB. Following edge AB, we can reach vertex B. Consider the distribution of the sizes of those clusters reachable by node B and let $H_1(x)$ be the probability generating function that generates the distribution of the sizes of these clusters. Based on [14], we have

$$H_1(x) = x\sum_{k=0}^{\infty} q_k[H_1(x)]^k = xG_1(H_1(x)). \tag{3}$$

If we randomly pick one vertex, the distribution of the sizes of the clusters to which this randomly chosen vertex belong to is:

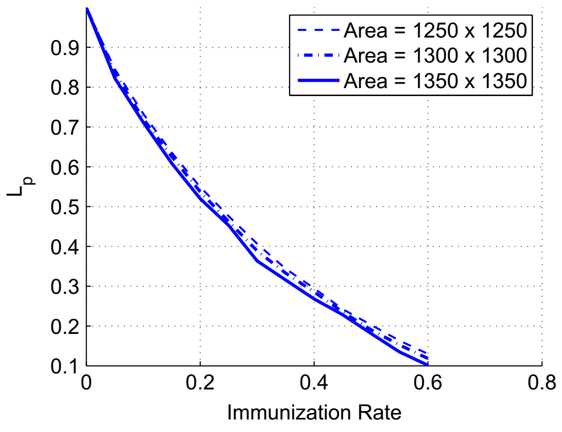


Fig. 12. Remaining link ratio.

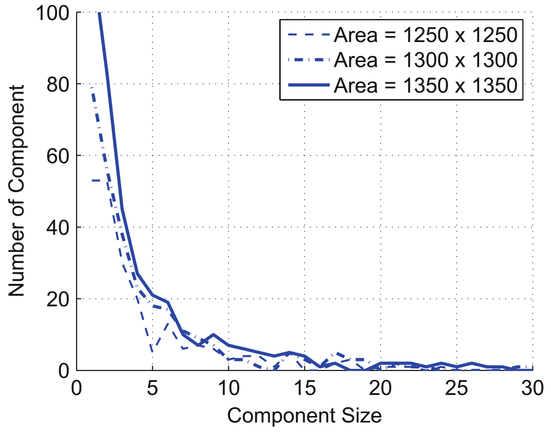


Fig. 13. Component size distribution after immunization.

$$H_0(x) = x\sum_{k=0}^{\infty} p_k[H_1(x)]^k = xG_0(H_1(x)). \quad (4)$$

Assuming that p_k follows a Poisson degree distribution, as we have illustrated in Section 5.1, following the basic ideas proposed in [14], we have $G_0(x) = G_1(x) = e^{z(x-1)}$.

Based on Eq. (3), we have:

$$H_1(x) = x(q_0 + q_1H_1(x) + q_2H_1^{(2)}(x) + \ldots). \quad (5)$$

Note that $q_0 = \frac{p_1}{<k>} = \frac{z^1e^{-z}}{1!}/z = e^z$. Starting from $H_1(x) = q_0x$, substituting this into the right part of Eq. (5), and ignoring the part at order x^2 and higher, we have $H_1(x) = x(q_0 + q_1q_0x)$, where $q_0 = e^z$. Based on Eq. (2), $q_1 = \frac{2p_2}{z} = ze^{-z}$.

Therefore, $H_1(x) = xe^{-z} + (xe^{-z})^2z$. Substituting this into the right part of Eq. (4), we have $H_0(x) = x(p_0 + p_1(xe^{-z} + (xe^{-z})^2z) + \ldots)$.

By computing the coefficient of $\frac{dH_0(x)}{dx^2}$, we can get the probability P_2 of a randomly chosen vertex belonging to components of size 2 is $P_2 = ze^{-2z}$.

Following this approach, we can iteratively compute the component size distribution for a given WSN network [14].

We simulate the empirical distribution of the component sizes. The simulation configuration we use is $1350 \times 1350\ \text{m}^2$ with 5000 nodes, in which top 35% most connected nodes are immunized. We use the Depth-First-Search algorithms to compute the component sizes. Based on our analysis in Section 5.1, we use a Poisson distribution to generate p_k and use the above methodology to calculate the theoretical distribution of component sizes.

The result is illustrated in Fig. 14. Theoretical analysis denotes the analysis results through above methodology. Note that the theoretical analysis proposed in [14] is used for a very large network. Here, we use a limited number of nodes which is typical for WSNs. This may account for the mismatch of the result.

With the above analysis, we present a simple approach to identify immunization nodes in order to prevent or slow down the large-scale mal-packet propagation in WSNs. Based on how many nodes are deployed over an area with a certain size, we can estimate the percolation threshold to disintegrate the network. This threshold enables us to calculate the *number of neighbor* threshold, denoted as N_{th}, above which one node should be immunized. Each node is then pre-equipped with N_{th}. After the deployment, each sensor node can measure connectivity information through a local *Hello* protocol and count the number of one-hop neighbors, denoted as N_n. If N_n of one node A is larger than N_{th}, node A should be immunized. Node A can then report its location to the field officer.

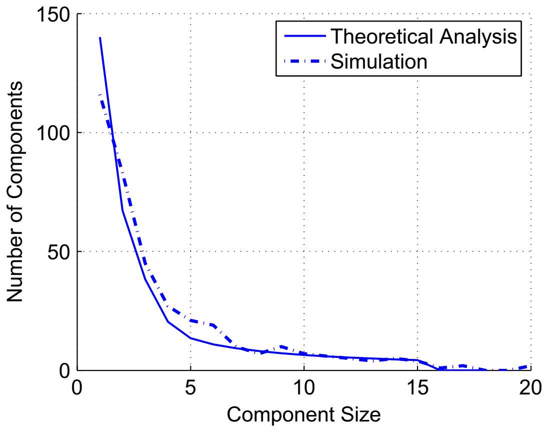


Fig. 14. Theory and simulation – component size distribution after immunization.

6. Related work

Few research works are focused on mal-packets on wireless networks. Yan and Eidenbenz [9,10] analyzed the worm propagation in Bluetooth networks and investigate the impact of mobility patterns on Bluetooth worm propagation. Khayam and Radha [11] proposed a topologically-aware worm propagation model (TWPM) for WSNs. By incorporating MAC and network layer considerations, TWPM captures both time and space propagation dynamics. De et al. [12] modelled the node compromise in WSNs based on epidemic theory. Gu and Noorani [15] presented attack approaches to construct specially crafted data message to facilitate mal-packet propagation in wireless sensor networks. Based on percolation theory, Zou et al. [17] also applied immunization-based approach to protect worm propagation in Email networks. Yang et al. [21] illustrated the feasibility of sensor worms on Mica2 motes. Then the philosophy of *survivability through heterogeneity* is used to explore techniques of software diversities to combat sensor worms. In [25], Xie et al. studied the feasibility of leveraging the existing P2P overlay structure for distributing automated security patches to vulnerable machines and examined two approaches. Zhu et al. [26] proposed a graph-partitioning approach to contain the propagation of a mobile worm in the context of cellular networks. The proposed methodology was shown to effectively limit the spread of MMS and SMS based worms.

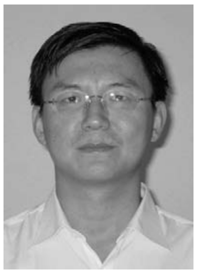
7. Conclusions and future work

In this paper, based on 802.15.4, we present a mal-packet propagation model in wireless sensor networks and a preliminary analysis of this model using TOSSIM. We study the mal-packet propagation dynamics under different protocol parameters and different network density. Based on percolation and random graph theory, we further propose an immunization-based countermeasure to protect WSNs from a large-scale outbreak of mal-packets. We select the most connected nodes to immunize and study their impact on preventing and slowing down mal-packet propagation.

Future work includes demonstrating the impacts of various factors of WSNs on mal-packet propagation dynamics. These factors include different network topology, the realistic characteristics of wireless sensor network including radio irregularity and transmission unreliability, and so on. It is also important to study practical defense schemes to prevent the large-scale outbreaks of mal-packets and systematically evaluate the proposed defense schemes.

References

[1] H. Kim, H. Cha, Towards a resilient operating system for wireless sensor networks, in: USENIX Annual Technical Conference, Boston, MA, June 2006, pp. 103–108.
[2] S. Bandyopadhyay, E.J. Coyle, An energy efficient hierarchical clustering algorithm for wireless sensor networks, in: IEEE INFOCOM 2003, San Francisco, CA, 2003, pp. 1713–1723.
[3] URL: <http://www.f-secure.com/v-descs/cabir.shtml>.
[4] J. Regehr, N. Cooprider, W. Archer, E. Eide, Memory safety and untrusted extensions for TinyOS, Tech Report UUCS-05-009, University of Utah, School of Computing, 2005.
[5] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, N. Weaver, Inside the slammer worm, IEEE Security and Privacy 1 (4) (2003) 33–39.
[7] K. Srinivasan, P. Dutta, A. Tavakoli, P. Levis, Understanding the causes of packet delivery success and failure in dense wireless sensor networks, in: ACM Sensys'06, Boulder, CO, 2006, pp. 419–420.
[8] S. Staniford, V. Paxson, N. Weaver, How to own the internet in your spare time, in: 11th USENIX Security Symposium (Security 02), August 2002.
[9] G. Yan, S. Eidenbenz, Bluetooth worm: models, dynamics, and defense implications, in: 22nd Annual Computer Security Applications Conference (ACSAC'06), Miami, FL, December 2006, pp. 245–256.
[10] G. Yan, S. Eidenbenz, Modeling propagation dynamics of bluetooth worms, in: IEEE ICDCS'07, Toronto, Canada, June 2007.
[11] S.A. Khayam, H. Radha, A topologically-aware worm propagation model for wireless sensor networks, in: IEEE ICDCS International Workshop on Security in Distributed Computing Systems, June 2005, pp. 210–216.
[12] P. De, Y. Liu, S.K. Das, Modeling node compromise spread in wireless sensor networks using epidemic theory, in: International Workshop on Wireless Mobile Multimedia, 2006, pp. 237–243
[14] M.E.J. Newman, Random graphs as models of networks, in: S. Bornholdt, H.G. Schuster (Eds.), Handbook of Graphs and Networks, Wiley-VCH, Berlin, 2003.
[15] Q. Gu, R. Noorani, Towards Self-propagate Mal-packets in Sensor Networks, ACM Wisec, Alexandria, VA, 2008.
[16] P. Levis, N. Lee, M. Welsh, D. Culler, TOSSIM: accurate and scalable simulation of entire TinyOS applications, in: ACM Sensys 2003, Los Angeles, CA, 2003, pp. 126–137.
[17] C.C. Zou, Don Towsley, W. Gong, Modeling and simulation study of the propagation and defense of internet email worm, IEEE Transaction on Dependable and Secure Computing 4 (2) (2007) 105–118.
[18] TinyOS, URL: <http://www.tinyos.net>.
[19] CrossBow Corporation, URL: <http://www.xbow.com>.
[20] Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Standard, 802.15.4-2003, May 2003, ISBN 0-7381-3677-5.
[21] Y. Yang, S. Zhu, G. Cao, Improving sensor network immunity under worm attacks: a software diversity approach, in: ACM Mobihoc'08, Hong Kong, 2008.
[22] G. Zhou, T. He, S. Krishnamurthy, J.A. Stankovic, Models and solutions for radio irregularity in wireless sensor networks, ACM Transactions on Sensor Networks 2 (2) (2006) 221–262.
[23] R. Draves, J. Padhye, B. Zill, Comparison of routing metrics for static multi-hop wireless networks, in: ACM SIGCOM'04, Portland, OR, September 2004, pp. 133–144.
[24] Y. Gu, T. He, Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links, in: ACM Sensys'07, November 2007, Sydney, Australia, pp. 321–344.
[25] L. Xie, Sencun Zhu, A feasibility study on defending against ultra-fast topological worms, in: Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing (P2P'07), Galway, Ireland, September 2007.
[26] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, A. Nucci, A social network based patching scheme for worm containment in cellular networks, To appear in IEEE INFOCOM, Rio de Janeiro, Brazil, 2009.
[27] S.K. Ghosh, R.K. Patro, M. Raina, C. Thejaswi, V. Ganapathy, Secure group communication in wireless sensor networks, in: First International Symposium on Wireless Pervasive Computing, Phuket, Thailand, 2006.



Bo Sun received his Ph.D. degree in Computer Science from Texas A&M University, College Station, USA, in 2004. He is now an assistant professor in the Department of Computer Science at Lamar University, USA. His research interests include the security issues (intrusion detection in particular) of Wireless Ad Hoc Networks, Wireless Sensor Networks, Cellular Mobile Networks, and other communications systems. His research has been supported by the Texas Higher Education Coordination Board's Advanced Research Program (ARP) and the US National Science Foundation (NSF).

Guanhua Yan obtained his Ph.D. degree in Computer Science from Dartmouth College, USA, in 2005. From 2003 to 2005, he was a visiting graduate student at the Coordinated Science Laboratory in the University of Illinois at Urbana-Champaign. He is now working as a Technical Staff Member in the Information Sciences Group (CCS-3) at the Los Alamos National Laboratory. His research interests are cyber-security, networking, and large-scale modeling and simulation. He has contributed about 20 articles in these fields.

Yang Xiao worked in industry as a MAC (Medium Access Control) architect involving the IEEE 802.11 standard enhancement work before he joined Department of Computer Science at The University of Memphis in 2002. He is currently with Department of Computer Science at The University of Alabama. He was a voting member of IEEE 802.11 Working Group from 2001 to 2004. He is an IEEE Senior Member. He is a member of American Telemedicine Association. He currently serves as Editor-in-Chief for International Journal of Security and Networks (IJSN), International Journal of Sensor Networks (IJSNet), and International Journal of Telemedicine and Applications (IJTA). He serves as a referee/reviewer for many funding agencies, as well as a panelist for NSF and a member of Canada Foundation for Innovation (CFI)'s Telecommunications expert committee. He serves on TPC for more than 100 conferences such as INFOCOM, ICDCS, MOBIHOC, ICC, GLOBECOM, WCNC, etc. He serves as an associate editor for several journals, e.g., IEEE Transactions on Vehicular Technology. His research areas are security, telemedicine, sensor networks, and wireless networks. He has published more than 300 papers in major journals, refereed conference proceedings, book chapters related to these research areas. Dr. Xiaos research has been supported by the US National Science Foundation (NSF), US Army Research, Fleet&Industrial Supply Center San Diego (FISCSD), and The University of Alabama's Research Grants Committee.

T. Andrew Yang is currently an associate professor of Computer Science and Computer Information Systems at the University of Houston – Clear Lake, Houston, Texas. His current research interests primarily focus on Computer Security, Network Security, and Wireless Sensor Networks. His research has been supported by the Texas Higher Education Coordination Board's Advanced Research Program (ARP), the Institute of Space Systems Operations (ISSO), and the US National Science Foundation (NSF).