Journal of Mobile, Embedded and Distributed Systems, vol. VI, no. 4, 2014
ISSN 2067 – 4074

Modelling Malware Response in Wireless Sensor Networks Using Stochastic Cellular Automata

Ahmad Uways ZULKURNAIN, Hassan CHIZARI
Department of Computer Science, Faculty of Computing,
Universiti Teknologi Malaysia
MALAYSIA
chizari@fc.utm.my

Abstract. A model based on cellular automata to analyze malware propagation is enhanced to simulate malware response using self-propagating software updates. The model maintains the characteristics of wireless sensor networks while adding states and behavior. The simulation tests variable update sources and variable distances between infection and update source. The simulation results confirm that node density influences the propagation speed. In addition, using more updater nodes greatly increases the speed of recovery and significantly reduces death rate. Finally, attacks occurring further away from updater nodes take longer to detect and cause a greater impact.

Key Words: wireless sensor networks (WSN); malware propagation; cellular automata; remote code update; attack response

1. Introduction

Wireless sensor networks (WSNs) have a variety of applications [1], including environmental studies [2], wildlife monitoring [3], military use [4], agriculture[5], and healthcare[6], among others. Wireless sensor networks can be highly distributed and grow to a large size [7].

Monte Carlo simulations [8], epidemiological models [9], and signal processing techniques [10] have been used to study malware propagation over ad-hoc networks. Song and Jiang [11] proposed a model using cellular automata in WSN and found it to be an appropriate model to simulate worm propagation.

There have been several proposed models for a self-propagating update mechanism for WSNs [12-14]. One of the advantages of having a self-propagating update system in place is that patches for malware can be issued remotely to sensors in a WSN without having to access each node individually. However, it is unknown how such mechanisms will behave or how successful the update system will be at responding to malware infections. Thus, a model for simulating response to malware attack is needed. Enhancing Song and Jiang's model [11] to add behavior of an automated update system can simulate the success of response to an attack.

Cellular automaton (CA) is a mathematical mode for natural systems, containing large number of components with local interactions. CA is easy to construct but exhibit complex behavior [15]. It is an efficient method to investigate the evolution of self-organizing systems, of which WSNs are of such nature. Stochastic cellular automata extend CA by adding probabilistic elements for local interactions and state changes [16].

For this project, an attempt is made to duplicate the model presented by Song and Jiang [11]. Next, an additional element of self-propagating updates is added to simulate response to malware attacks on a WSN. The developed model inserts additional behavior conditions while maintaining the characteristics of Song and Jiang's model. Like the referenced model, parameters can be set to reflect different scenarios of WSN.

Numpy and Matplotlib modules of SciPy [17] was used to create and run the simulations. The results from the SciPy simulations are shown is Section 5.

The report is organized as follows. In the next section, the model of cellular automata is described. In Section 3, the methodology and enhanced model is proposed. Parameters and assumptions

are considered in Section 4. In section 5 the simulation results are presented along with related discussion. Finally, a conclusion is made and future work discussed.

2. Defining the Cellular Automata

A 2-D CA is a discrete dynamical system formed by a finite number of l × r cells arranged uniformly in a two-dimensional space. Each cell is assigned a state from a finite set of states, Q, which changes based on the defined local transition rules. The state of a cell at time t is determined by the previous states at t-1 of a set of cells, called its neighborhood. A cell, s, within the cell space of l × r can be defined as a coordinate where

$$s = (i, j)\,,\ 1 \le i \le l,\ 1 \le j \le r \quad (1)$$

If the neighborhood of s is given a V and the transition function is f(x), the current state of s can be described as

$$s_t = f(V_{t-1}) \subset Q \quad (2)$$

3. Modelling the Problem

The developed model focuses on the stochastic properties of CA to mimic actual behavior of malware propagation and WSN operation.

Consider a flat WSN composed of N identical stationary sensors randomly distributed across a 2-D space of S*S cells. Each cell can be occupied by one node. Density of nodes is denoted by p where p = N / S^2. Each sensor can establish links with nodes within a circular radius, r. It is assumed that the sensors have omnidirectional antennae and can maintain a maximum transmission range of r until power depletion.

The value of r will define the neighborhood of a sensor in the CA space. The possible neighborhood types are shown in Figure 1. A von Neumann neighborhood can be used to simulate r=1 while a Moore neighborhood can be used to represent r=1.5. Higher values of r can also be used by adjusting the neighborhood definitions such as in (c) and (d) of Figure 1.

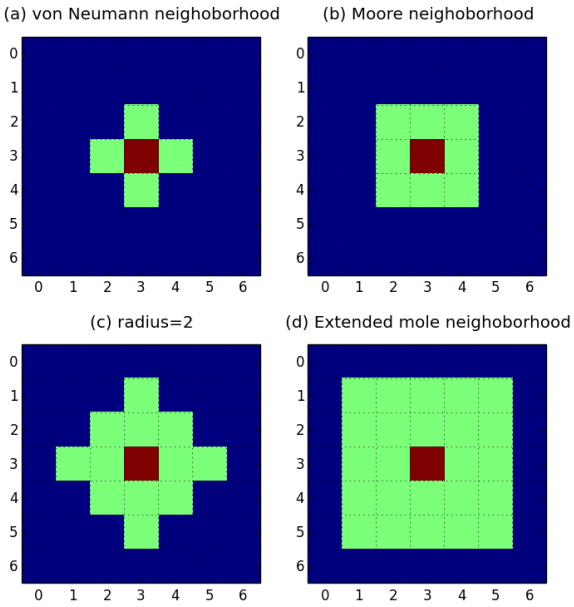

Figure 1. Cellular automata neighborhoods

Song and Jiang [11] used the concept of epidemiology to represent possible state transitions with corresponding probabilities for each transition. They defined four state of a sensor: susceptible, infected, recovery, death. They assumed that unfilled cells were dead nodes, but for this project the state of empty cells and dead nodes are separated to none and dead. For modelling attack response using self-propagating updates, two additional states were added: update source and updated. The model developed for this project uses integer values to represent each state as shown in Table 1.

Susceptible nodes become infected with a probability of b if there are infected nodes in its neighborhood. Similarly, if there are updated nodes in its neighborhood it will receive an update with a probability of h. Infected nodes will continuously transmit and drain power if there are at least one non-infected node in its neighborhood. Update source nodes remain dormant until the malware is detected in its neighborhood after which it will become an updated node and begin to propagate the software patch to other nodes. Updated nodes have a fix installed and are immune to the malware.

ISSN 2067 – 4074

Table 1: Cell states

Value	State
0	Empty
1	Susceptible
2	Infected
3	Recovering
4	Dead
5	Update Source
6	Updated

4. Related Parameters and Assumptions

In this section a description of the simulation parameters are presented to explain their purpose and effect on the simulation. The parameters of the simulation must reflect the actual behavior of a WSN and present meaningful results

4.1 Infected rate

The infected rate was selected based on Song and Jiang's work as these values produced good simulation results. The infected rate, b, has a value between 0 and 1.

4.2 Recovery rate

Even without a software update a node has a chance to heal itself with a probability of d, with value between 0 and 1. The chance for automatic recovery should be set with a small value, as this scenario is less likely to occur in comparison to being infected or updated.

4.3 Power consumption rate

Sensor nodes are limited in power reserves; therefore it is relevant that power usage be included in the simulation as it will impact the results. Nodes which are not infected are assumed to use a negligible amount of power in the scope of the simulation. However, infected nodes continuously transmit and therefore consume power at a rate of c for each time interval where it is transmitting, where c is the fraction of power consumed in a single time unit. For the simulation, all nodes are assumed to begin with full power which depletes over time caused by the malware.

4.4 Response time

A software update for the WSN is expected to take a certain amount of time to develop and issue. This lapse of time is denoted by h, where an update will be issued at time $t+h$ after detection of malware at time t.

4.5 Update rate

Nodes will not always be receiving transmissions to save battery, so it is not certain that an update will be immediately issued to neighboring nodes. The update rate is represented by e, where value of e is between 0 and 1.

4.6 Self-propagating software update

For the simulation, it is assumed that the WSN contains a self-propagating update functionality where a given node can be updated and disseminate the same update to other nodes in the network. It is assumed that once the software update is received, the node becomes immune to malware and will no longer risk being infected.

4.7 Routing mechanism

It is assumed that the WSN uses a multi-hop routing protocol which is obeyed by both the malware and the updater. Nodes can only transmit to the nodes in their neighborhood.

4.8 Random node distribution

The sensor density, p, can be used as a variable to determine the probability of a cell containing a node. This provides an evenly random distribution of sensors in the CA space, but does not guarantee a set value of N nodes. An alternative method for distributing nodes is by first determining the value of N, the number of sensors, and randomly place that many sensors in the CA space without overlapping them. This is achieved by first inserting the sensors one by one adjacent to each other, then shuffling the contents of the CA space to create a random distribution.

Besides the density of the node, the connectivity of the nodes is also important so that there are no isolated network segments. The distribution of the sensors was manually inspected to prevent a large number of unconnected nodes. This method could not prevent small numbers of unlinked sensors. However, small numbers of unlinked nodes did not impact the simulation results. Obtaining a good distribution sample became increasingly difficult with lower p and N values.

5. Simulation Results

The malware propagation and update response was simulated in a WSN consisting of N sensors randomly distributed over a cellular automata space of $S^2 = 100 \times 100$ unit^2 cells. The sensor density is denoted by p. The communication range is defined by radius, r. For r=1 a von Neumann neighborhood is used and for r=1.5 a Moore neighborhood is used. Each simulation starts with a single infected sensor and variable number of update sources, w, and the rules used are those described in section 3 and section 4.

The first scenario is a replication of Song and Jiang's model where no updates are available, thus w=0. The power consumption rate was set as c = 0.01. A single infection source is used instead of the three initial infected cells used by Song and Jiang. The simulation was also only run for 700 time units compared to 1000 time units by Song and Jiang. Parameters p, b and r were changed to obtain different results constants d=0.001 and c=0.0035. The results obtained by Song and Jiang, shown in Figure 2, were used as a baseline and the parameters replicated for the simulation. Results shown in Figure 3 closely resemble the in Figure 2. Thus, it can be concluded that the developed model accurately represents WSN behavior through cellular automata.

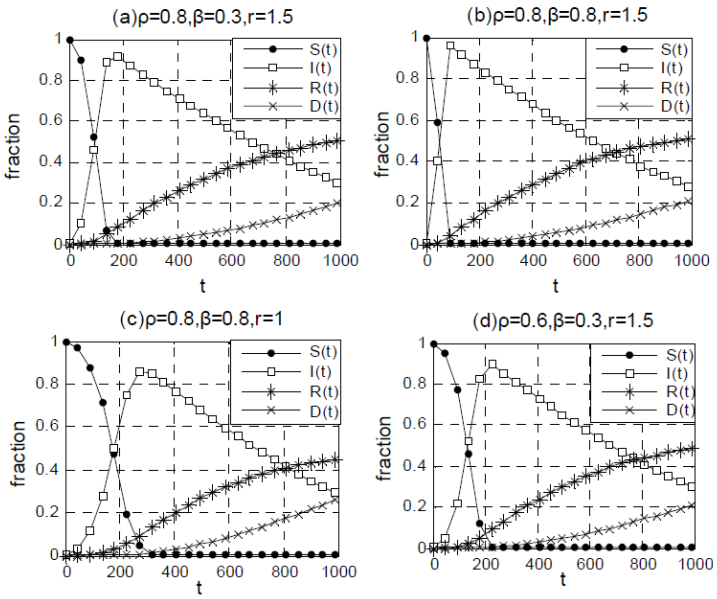

Figure 2. Results of Song and Jiang [11]

ISSN 2067 – 4074

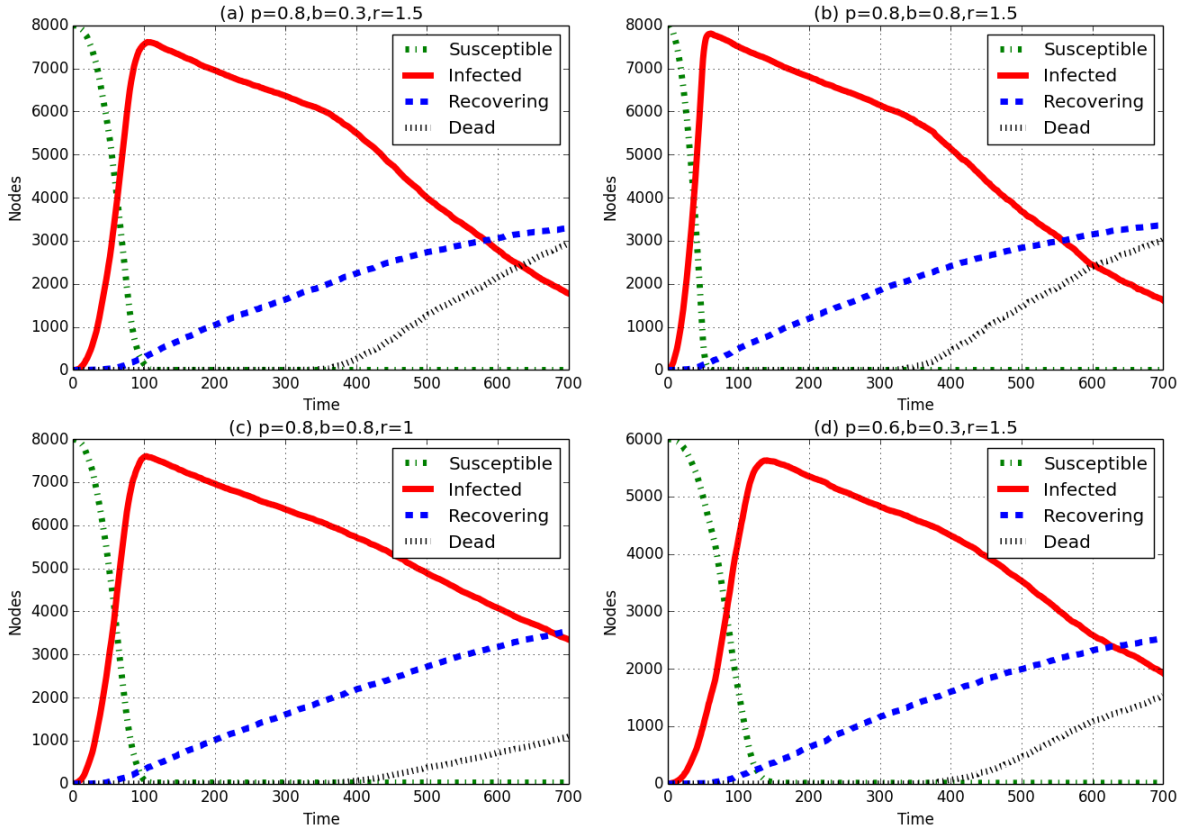

Figure 3: Simulation results for developed CA model

The next scenario adds a variable number of update sources. The update sources are placed a fixed distance from the attack source at 45, 135, 225, and 315 degrees relative to the attack source location at the center of the map. Thus for an attack node at (50, 50), updater nodes were placed at (25, 25), (25, 75), (75, 25) and (75, 75) respectively, given a space of 100*100. Once one updater node detects the malware, the response timer is started. After the set response period, all updater nodes transform their state into updated nodes and begin propagating the update to other nodes. Figure 4 shows the results for the simulation where w=1 in 4(a), w=2 in 4(b), and w=3 in 4(c). The rest of the parameters were kept constant with p=0.8, b=0.3, d=0.01, c=0.01 and h = 20. So after detection, an update is issued after 20 time periods. Power consumption, c, was increased to simulate a longer lapsed time period for each simulation step. It also increases the impact of the malware and produces more contrasting results within a shorter simulation period.

In Figure 4(a), it can be seen that using only a single updater node for the whole network is not very effective in managing malware propagation. Nodes stop becoming updated after around t=230 because the number of dead nodes has become too high and the network is poorly connected, thus the update can no longer propagate, whereas infected nodes continue to drain power leading to the majority of nodes dying. In Figure 4(b) 2 nodes were used, which reduced the spread of malware, but still resulted in a high death count. However, with 4 updater nodes surrounding the attack source, was quickly eliminated in 200 time units and less than 10% of the nodes suffered death.

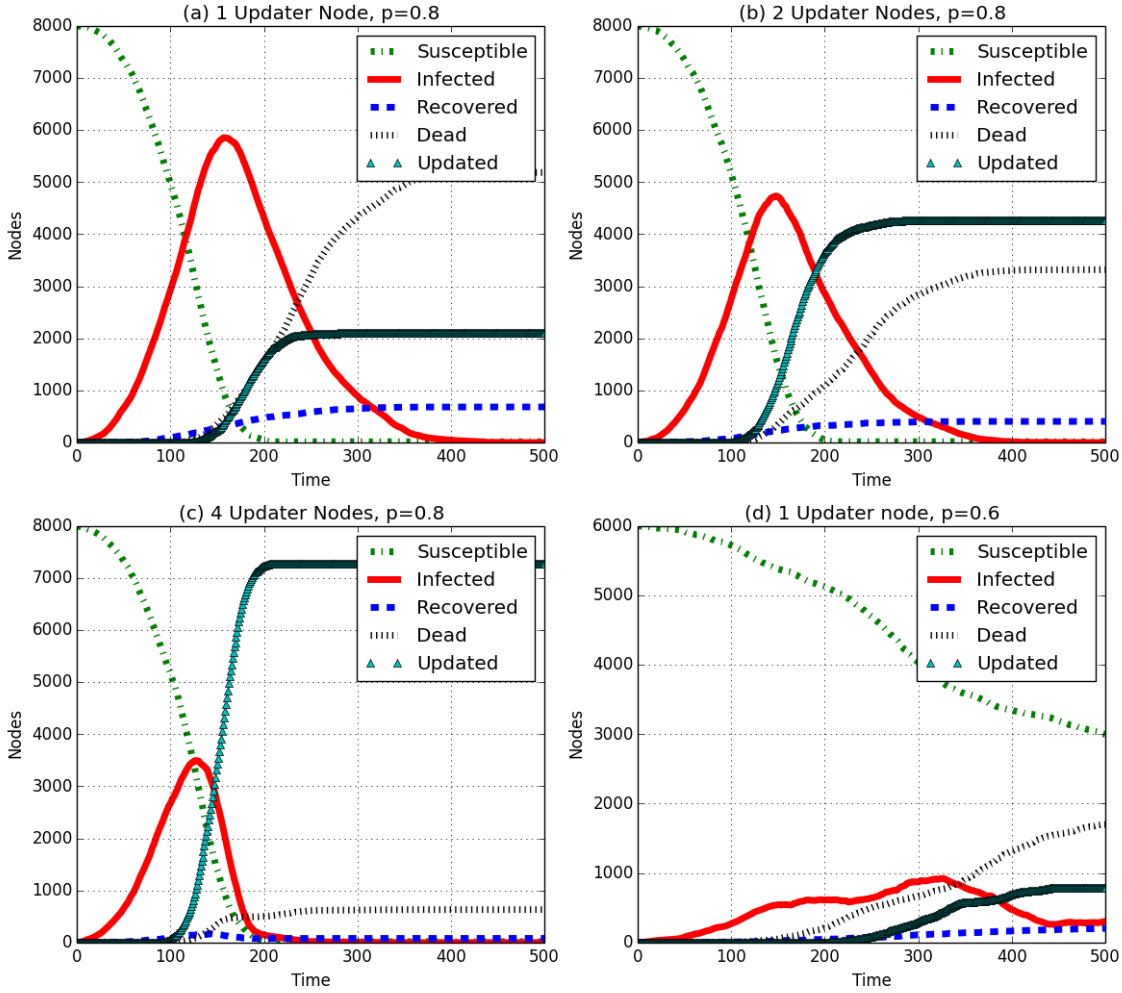

Figure 4. Attack response using self-propagating updates

In Figure 4(d), w=1 but the density of the network was changed to p=0.6. Connectivity is poor, so while both malware and updates fail to propagate well, the rate of death is higher than rate of update propagation, which suggests that most of the network will not receive the update before power levels are depleted.

The last experiment carried out was changing the distance between the attack source and the updater node. In this experiment w=1 located 20, 40, and 60 units away from the attack source. The state of the CA space was captured at intervals of t=3, t=100, t=200, and t=300. Using these four captured frames is sufficient for analyzing the results of the simulation. Other constants remained the same as the ones used in the simulation shown in Figure 4(a) with p=0.8, b=0.3, d=0.01, c=0.01 and h = 20.

In Figure 5, it can be seen that having the updater node very close to the source of infection means that the infection can be detected quickly and healed before any serious damage is done to the network. At t=300, the malware has been almost completely eradicated with the majority of nodes receiving updates and becoming immune.

In Figure 6, the malware manages to reach infect most of the network, but recovery from the malware is still largely successful. There are more dead nodes than in Figure 5, but still only a small fraction of the total number of sensors are lost.

ISSN 2067 – 4074

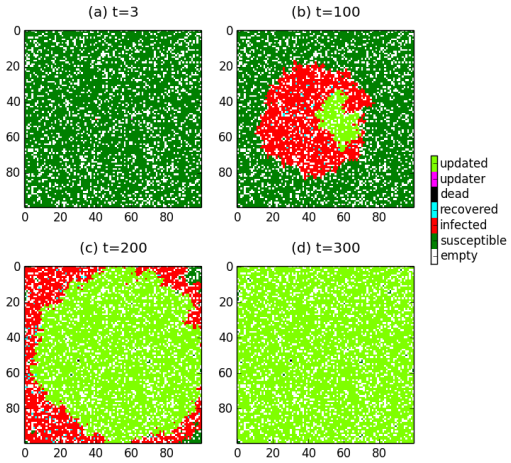

Figure 5. 20 units distance between infection source and updater

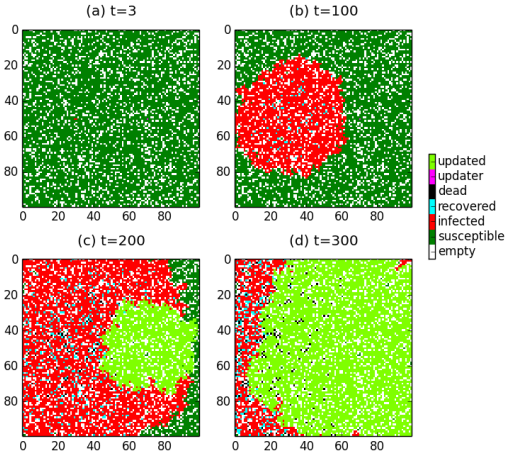

Figure 6. 40 units distance between infection source and updater

In Figure 7, the malware is detected too late and the updates struggle to propagate through the network because of the increasing number of dead nodes disconnecting the network.

6. Conclusions and Future Work

A model based on cellular automata has been enhanced to simulate self-propagating updates in response to discovery of malware propagation in a WSN. The model was found to retain the characteristics of Song and Jiang's model while adding two extra states and rules to the CA. Self-propagating updates certainly reduce the impact of malware propagation. However, factors such as number of updater nodes, node density, node placement, and network connectivity play important parts in determining the outcome of the simulation.

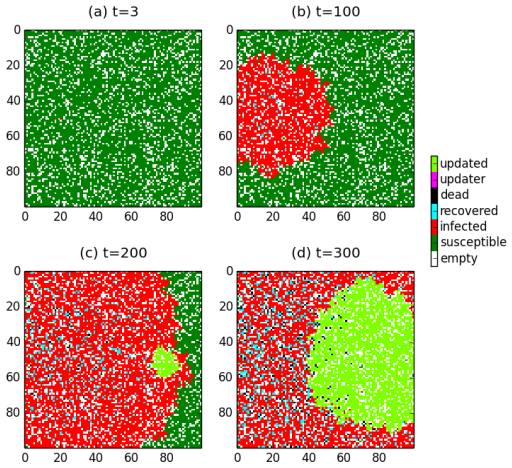

Figure 7. 60 units distance between infection source and updater

For future works, it is important to create a method that can guarantee full network connectivity of a randomly generated CA space, specifically in the case of low density values of p less than 0.8. More rules can also be added with more cell states to simulate more complex events, behavior and interactions between cells.

References

[1] Akyildiz, Ian F., et al. "Wireless sensor networks: a survey." Computer networks 38.4 (2002): 393-422.

[2] Werner-Allen, Geoffrey, et al. "Monitoring volcanic eruptions with a wireless sensor network." Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on. IEEE, 2005.

[3] Mainwaring, Alan, et al. "Wireless sensor networks for habitat monitoring." Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications. ACM, 2002.

[4] Lee, Sang Hyuk, et al. "Wireless sensor network design for tactical military applications: remote large-scale environments." Military Communications Conference, 2009. MILCOM 2009. IEEE. IEEE, 2009.

[5] Baggio, Aline. "Wireless sensor networks in precision agriculture." ACM Workshop on Real-World Wireless Sensor Networks (REALWSN 2005), Stockholm, Sweden. 2005.

[6] Lo, Benny, et al. "Body sensor network-a wireless sensor platform for pervasive healthcare monitoring." The 3rd International Conference on Pervasive Computing. Vol. 13. 2005.
[7] Niu, Ruixin, and Pramod K. Varshney. "Distributed detection and fusion in a large wireless sensor network of random size." EURASIP Journal on Wireless Communications and Networking 2005.4 (2005): 462-472.
[8] Nicol, David M. "The impact of stochastic variance on worm propagation and detection." Proceedings of the 4th ACM workshop on Recurring malcode. ACM, 2006.
[9] M. Nekovee, "Worm epidemics in wireless ad hoc networks," New Journal of Physics, vol. 9, pp. 189, 2007.
[10] S. A. Khayam and H. Radha, "Using signal processing techniques to model worm propagation over wireless sensor networks," Signal Processing Magazine, IEEE, vol. 23, pp. 164-169, 2006.
[11] Song, Yurong, and Guo-Ping Jiang. "Modeling malware propagation in wireless sensor networks using cellular automata." Neural Networks and Signal Processing, 2008 International Conference on. IEEE, 2008.
[12] Stathopoulos, Thanos, John Heidemann, and Deborah Estrin. A remote code update mechanism for wireless sensor networks. No. CENS-TR-30. CALIFORNIA UNIV LOS ANGELES CENTER FOR EMBEDDED NETWORKED SENSING, 2003.
[13] Patel, Neil, David Culler, and Scott Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. Computer Science Division, University of California, 2003.
[14] Brown, Stephen, and Cormac J. Sreenan. "A new model for updating software in wireless sensor networks." Network, IEEE 20.6 (2006): 42-47.
[15] Wolfram, Stephen. "Universality and complexity in cellular automata." Physica D: Nonlinear Phenomena 10.1 (1984): 1-35.
[16] https://www.cs.purdue.edu/homes/park/interest-ca.html
[17] http://scipy.org/