TOWARDS MODELING THE BEHAVIOR OF PHYSICAL INTRUDERS IN A REGION MONITORED BY A WIRELESS SENSOR NETWORK

by

PRANAV KRISHNAMOORTHY

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2010

To my parents - without their constant efforts I would never be.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Matthew Wright, his encouragement and insight have been a constant source of inspiration. Without his guidance and persistent help this thesis would not have been possible. I am grateful to Dr. Manfred Huber and Dr. Vassilis Athitsos for their invaluable advice and interest in my research and for taking time to serve on my thesis committee.

I would like to extend a special thanks to my colleagues: Titus Abraham and Kush Kothari. Over endless hours spent brainstorming over the whiteboard, we have tackled many a problems and I attribute a lot of my success to them. I would like to thank my wonderful colleagues at the Information Security Lab for providing an excellent atmosphere to work in. I would also like to thank my room-mates and the wonderful folks at the Graduate School for making these past years thoroughly enjoyable.

Finally, I would like to express gratitude to my mother for her unwavering support and encouragement.

April 14, 2010

ABSTRACT

TOWARDS MODELING THE BEHAVIOR OF PHYSICAL INTRUDERS IN A REGION MONITORED BY A WIRELESS SENSOR NETWORK

Pranav Krishnamoorthy, M.S.

The University of Texas at Arlington, 2010

Supervising Professor: Matthew Wright

A priority task for homeland security is the coverage of large spans of open border that cannot be continuously physically monitored for intrusion. Low-cost monitoring solutions based on wireless sensor networks have been identified as an effective means to perform perimeter monitoring. An ad-hoc wireless sensor network scattered near a border could be used to perform surveillance over a large area with relatively little human intervention.

Determining the effectiveness of such an autonomous network in detecting and thwarting an intelligent intruder is a difficult task. We propose a model for an intelligent attacker that attempts to find a detection-free path in a region with sparse sensing coverage. In particular, we apply reinforcement learning (RL) – a machine learning approach, for our model. RL algorithms are well suited for scenarios in which specifying and finding an optimal solution is difficult. By using RL, our attacker can easily adapt to new scenarios by translating constraints into rewards. We compare our RL-based technique to a reasonable heuristic in simulation. Our results suggest

that our RL-based attacker model is significantly more effective, and therefore more realistic, than the heuristic approach.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Sensors are inexpensive, low-power devices that are capable of limited local processing and wireless communication and are used to facilitate monitoring of physical environments. Wireless sensor networks are large collections of sensor nodes that coordinate to perform some specific action [1]. A dense concentration of sensors forming an ad-hoc network could be used in identifying and tracking movement. Such networks could be used to perform unmanned surveillance of hostile environments, such as borders, for detection of intruders (physical entities) and are therefore of significance to the military [2].

Wireless sensor networks are an increasingly attractive means to bridge the gap between the physical and virtual world [3]. Attack models on sensor networks include attacks on both the physical aspects such as device tampering, jamming — as well as logical aspects such as sybil attacks, attacks on the routing protocols and selective message forwarding [4].

Two parameters are relevant to autonomous intrusion detection - (a)*connectivity*: the sensor network needs to be connected to enable transmission of data to base stations and (b)*coverage*: each location in the physical space of interest should be within the sensing range of at least one active sensor [5, 6]. These parameters are used to determine the topology and density of the distribution and are being actively investigated [2, 7, 8]. Maintaining complete connectivity and full coverage is a special case of this problem and Tannenbaum et. al. discuss the issues in achieving sufficient coverage for a large area such as a border [9, 10]. Node faults and irregularity in sensing

range further affect the coverage. These issues indicate that *gaps* may exist in the coverage areas even if the distribution was calculated to be optimal. The presence of these gaps is further emphasized when sensing coverage is less than 100%. As a consequence, paths may exist that could enable an intruder to evade detection while traversing the perimeter.

During transmission, wireless signals are broadcast in the air. This makes sensor devices particularly vulnerable to eavesdropping [11]. A passive eavesdropper collects the output of a sensor device by eavesdropping on the broadcast medium and can use this to identify what the node is observing. A similar approach can also be used by an intruder to estimate the positions of sensors in a monitored region. Essentially, the intruder enters the region and eavesdrops on the local transmission of wireless messages in response to intrusion. The messages may be encrypted, but the intruder can infer that the sending node is probably reporting a sensor reading that indicates his presence. By estimating the location of the sensor, he can map out its location. Signal strength measurements could be used to enhance the precision of the estimate. We use the term *probing* to denote this action of identifying a sensor node by inducing it to send wireless signals. By repeated probing, an attacker can map the perimeter with sufficient detail to find regions that are not monitored, possibly including complete paths crossing the region without detection. Finding such a path would create a critical breach of the security perimeter, as it could be used repeatedly to enter the secured region or cross the border with much lower chances of being discovered.

The most obvious defense against this form of attack would be to increase the sensor density until the chance of finding such a path became very small. However, increasing the density sufficiently could require a major increase in the number of sensors and a correspondingly large increase in the cost of the system. We look at

alternative defenses that are effective without the added cost. More specifically, we identify a way to reduce the timing correlation between the detection event and the corresponding notification to the base station.

Based on the behavior of the attacker outlined above, we propose an attacker model that uses a combination of probing and path-finding algorithms to identify detection-free paths in the perimeter. To imitate the behavior of the attacker, the model should be flexible, since a real attacker would be adaptive. The model should be capable of finding paths without exposing the attacker to large risks of getting caught.

Contributions: In this thesis, we suggest a *timing correlation based attack* for physical intrusion in a region monitored by wireless sensor networks, that allows the intruder to identify detection free paths. We establish the necessity for identifying alternate defense mechanisms against this model. In Chapter 3, we suggest a heuristic model for the attacker, and based on our model suggest a defense to this attack. We also identify the shortcomings of the heurisitcs model and propose a model that learns from the environment to better resemble the actions of a human intruder. In Chapter 4, we introduce a intruder model based on reinforcement learning. In Chapter 5, we contrast the performance of the RL-based agent against the heurisitc model. Our results suggest that the learning agent performes signficantly better; however the proposed defense continues to work reasonably well. Our results also provide us critical perspective in ways to improve the existing model and we discuss the some possible improvements in Chapter 6.

CHAPTER 2

BACKGROUND

In this Chapter, we discuss some of the more commonly explored forms of attacks on wireless sensor networks and then provide an overview on machine learning and reinforcement learning, the basis of our second intrusion model. We also describe some related work.

2.1 Physical attacks and countermeasures in wireless sensor networks

Adversaries can launch an attack on the physical layer of the wireless sensor network (WSN), thereby obstructing network operations. Physical layer in WSNs consist of sensor devices and the shared wireless medium. The following types of attacks are typically categorized as physical layer attacks [4, 12]:

- Device tampering: Sensor nodes are unmanned and therefore an attacker can simply damage or destroy the nodes. The inherent redundancy in wireless networks makes this a particularly tedious form of attack.
- Eavesdropping: Using relatively modest equipment, an attacker can eavesdrop on transmissions between sensor nodes. Eavesdropping is a passive form of attack and depends on the ability of the attacker to collect raw content from wireless packets and extract useful information from them. Encrypting communication can mitigate the risk of directly leaking sensitive or critical information to a passive adversary.
- Jamming: Jamming is a denial of service attacks that disrupts the availability of the shared medium. A single jamming device can affect several nodes,

making jamming attacks far more damaging as compared to device tampering. Frequency hopping and spread spectrum are some among many techniques that could be applied to counter this attack.

The data in a wireless sensor network is aggregated at centralized base stations with greater processing capabilities. Base stations act as the backbones of the WSN and typically form an interface between the sensor nodes and the user.

As discussed earlier, encryption of the transmission prevents eavesdropping, however, wireless channels are susceptible to *traffic analysis*. In their paper, Deng et al [13] suggest an attack using traffic analysis to identify the position of base stations. Two classes of traffic analysis may be applied. In a *rate correlation attack* an attacker monitors the packet sending rate of nodes. Nodes closer to the base station have a higher transmission rate. In a *time correlation attack*, an adversary observes the correlation in sending time between a node and its neighbour to identify the path as the packet propogates towards the base station.

By identifying the routes and volume of traffic of sensor data, the upstream direction of traffic can be identified, ultimately resulting in identifying the location of the base station. Tampering with or destruction of the base station would result in significantly larger damage to the operation of the network. The defenses proposed against these attacks include introuducing redundancy and randomization in packet routes and using random fake paths to confuse an adversary.

2.2 Reinforcement Learning

An important branch in artificial intelligence is the field of machine learning. Machine learning typically refers to algorithms that cause change in systems that result in either enhancements or synthesis of new systems based on acquisition and integration of knowledge [14]. Machine learning algorithms begin with a hypothesized

function h, that maps the input domain $\mathcal{I}$ to an output domain $\mathcal{O}$ that approximates the output of an idealized function f [15, 16]

In supervised learning, the function f is known over the training set $\mathcal{I}_t$. The algorithms are designed on the basis that a function h that approximates the output for $\mathcal{I}_t$ should perform well for a larger unknown input set $\mathcal{I}$. In unsupervised learning, a training set may exist, but no well-defined function f exists. In this work, we focus on class of algorithms called *reinforcement learning*.

In reinforcement learning (RL), an agent interacts with an unknown environment and attempts to choose actions that maximize its cumulative payoff [17, 18]. The agent receives sensory input from the environment and performs an action based on this input. The results of the action change the environment in some manner, which is returned as a scalar reward value (reinforcement) to the agent. A basic RL model consists of:-

- s: the set of environment states.
- a: the set of agent actions.
- r: the reinforcement receieved for performing an action.

Q-Learning is one such reinforcement learning technique [19]. At every state, the agent choses from one of the actions a_t at a state s_t and receives a reward r_t for it. The result of the action a_t causes the agent to transition to the next state s_{t+1}. The goal of the agent is to maximize the total expected reward. The agent's policy π determines the agent's actions. An optimal policy π^* is a policy that maximizes the rewards for the agent at every action.

In Q-Learning, the optimal policy is discovered by associating with each state-action pair a Q value. $Q(s_t, a_t)^\pi$ is the expected reward at a state s_t when the agent performs an action a_t and continues with a policy π. The set of actions a^* for the optimal set of Q values, would then be the optimal policy function [18]. The

Q-Learning agent performes an infinite set of iterations called episodes, constantly adjusting the Q-values as it visits state-action pairs as:

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha[r_{t+1} + \gamma(\max_{a \in a_t} Q(s_{t+1}, a) - Q(s_t, a_t))]$$

where α is the *learning rate* and γ is the *discount factor* [17].

The discount rate is a measure of the value of rewards that were received from earlier steps. Specifically a reward recieved n steps earlier is multiplied by γ^n. The learning rate determines the effect that the current reward has on future rewards. The values of α and γ range between $(0 - 1]$. In early episodes, the Q values for the agent would be poor approximations of the optimal policy. These approximations would be increasingly refined with each episode.

At any state, the agent can choose to either *exploit* i.e. choose an action based on previous experience, or to *explore*. To guarantee that the results of Q-Learning converge, the agent must select each state-action pair an infinite number of items. The probability p of selecting any action must therefore be non-zero. Random exploration works well for smaller statespaces, but it introduces unnecessary noise for larger statespaces. Boltzmann exploration interweaves the actions of explore and exploit by selecting an action a with the probability

$$p_x(a) = \frac{e^{\frac{Q(x,a)}{T}}}{\sum_{a_t \in A} e^{\frac{Q(x,a_t)}{T}}}$$

The versatility of reinforcement learning allows it to be applied to a wide variety of problems. Reinforcement learning has been applied in several problems where continuous on-line learning is required. In their study of vulnerabilty assessments of peer to peer networks, Dejmal et.al. develop an attacker model based on reinforcement learning [20]. The learned control policies are used to select the actions of a botnet of one or more malicious nodes.

Reinforcement learning techniques have also been used to develop adaptive intrusion detection methods [21] for computer (wired) networks. Complex intrusion behavior is represented as a series of patterns and the learning agent is trained on audit data to identify these patterns.

CHAPTER 3

A RULE-BASED ATTACKER MODEL

In the following Chapter, we propose a model for the world, develop a heuristic model for attack based on this model and propose a defense against this attack.

3.1 Modeling the world

The simulation world represents a large perimeter that is to be monitored for intrusion. For the sake of simplicity, we assume that the world is a rectangular $n * m$ grid with sensors located at integer positions (i, j). Sensors are randomly distributed across this world; each sensor may be located in any one of the blocks of the grid world.

The number of sensor nodes in the world are specified by the coverage which is expressed in terms of the percentage of total area of the grid world that is under coverage. The sensing radius is specified in terms of manhattan units and we set the radius to one for our simulations. This means that each sensor can detect the intruder in any of the one to ten surrounding neighbors of its position. We do not consider connectivity to neighboring sensors when placing sensors. If we assume that wireless connecions go over longer distances, than sensing coverage, the network is very likely to be fully connected in all of our simulations.

$$\text{Number of nodes} = \text{Coverage} * \frac{\text{Area of world}}{9}$$

The world-grid can be perceived as a graph where each block is a node in the graph with edges in the four directions - *up, down, left and right.*

3.2 Modeling the attacker

The attacker is a location-aware intruder attempting to find a detection-free path through the perimeter. The attacker travels at unit speed, one block per unit time. An attacker using a regular path-finding algorithm would get detected and caught. To detect a path, the attacker would have to identify the location of the sensors which in turn means triggering them. By continuing to explore the perimeter once detected, the attacker would certainly be caught. In the event that the attacker realizes that he has been detected, the safest option would be to retreat back to the edge of the perimeter. This retreat action is an indicator of passage of time and allows the attacker to incrementally discover a path. During each iteration, the attacker utilizes any information that was gathered during previous iterations to make decisions.

The attacker applies a depth first traversal to identify paths. Blocks that are under sensor-surveillance are marked as monitored. Blocks that transit to monitored blocks or other deadend blocks are guaranteed to not have a sensor-free path to the far edge and are marked as deadend.

Algorithm 1 A heuristic approach to intrusion

repeat

Choose from *up, left, right* where not monitored or deadend

if no blocks to choose **then**

Mark the block as a dead end. Pop from the stack

else

Push the block onto the stack and continue

end if

until Not path found

Once the attacker probes and identifies that it has been detected, it begins to retreat. Prior to retreating we allow the attacker to explore the neighborhood of the location at which it was detected. This allows the attacker to identify alternate paths (or identify dead-ends) to traverse once it returns to this point. The amount of time that the attacker explores the environment is determined by the restraint index $\kappa = d * exp(-g(t))$, where d is the distance from the far end (exit) of the border and $g(t)$ is a function of the amount of time since detection.

Time in the model is measured in terms of the number of actions performed by the attacker and each movement to a neighboring block takes unit time. As a result $g(t)$ is measured in terms of the number of steps taken by the attacker since detection. We used varying powers of $g(n)$ to vary κ.

3.3 Simulation Results

Our simulations were executed on a world of size $200x200$ with node coverages between $0 - 70\%$. The initial position of the attacker is randomly chosen along any point on one edge of the perimeter.

We define detection rate as the number of actions that results in detection versus the total number of actions that the attacker performs. Figures 3.1 and 3.2 plot the detection rate and the time taken for different values of $g(n)$. At lower concentrations, an attacker using a higher restraint index performs well, detecting paths quicker at equivalent detection rates. Lower restraint indices are better at higher node concentrations. Thus, an algorithm that could adapt to the node concentration should perform well under all conditions.

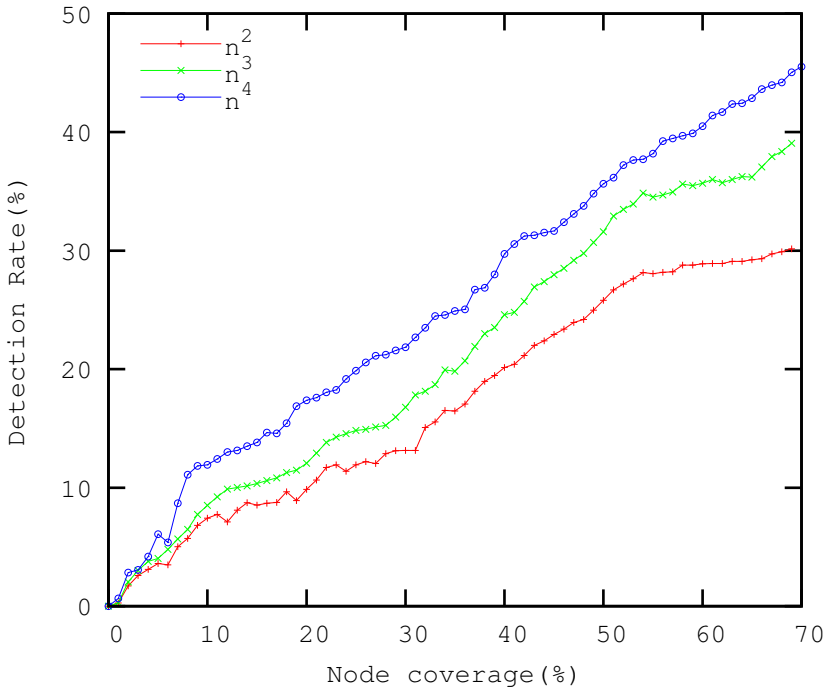

Figure 3.1. Frequency of detection versus node coverage.

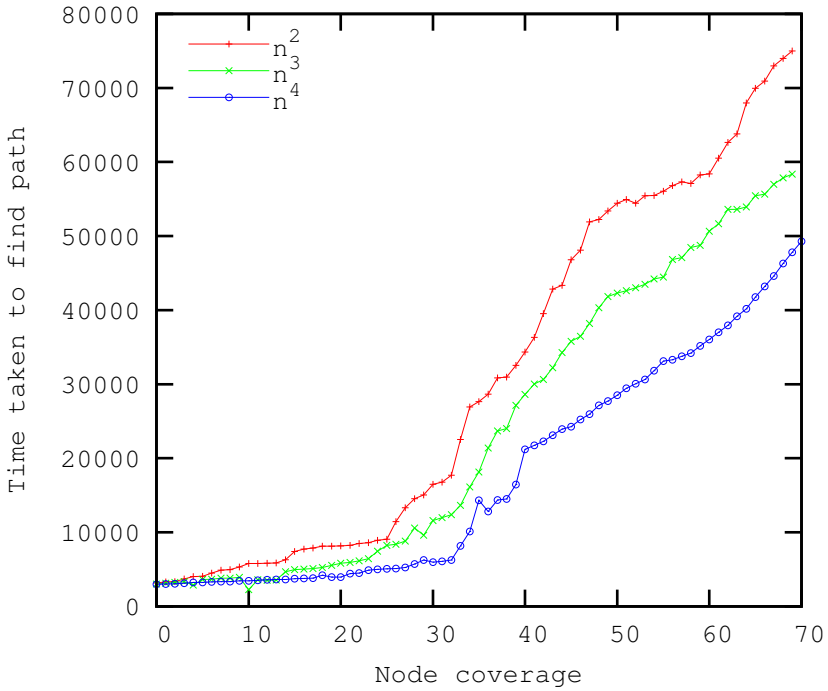

Figure 3.2. Time taken versus node coverage.

3.4 Delayed Trigger

To construct a scenario more fully aligned with real costs to the attacker, rather than arbitrary values, we add a *sentry* to the model. The sentry represents a border guard with the capability to capture the attacker and thereby cost the attacker substantially in time and resources.[1]

Next, the algorithm works under the premise that a probe results in a definitive response that a specific block is under sensor surveillance. Cover traffic is one mechanicm that could be used to make this a difficult task. In this defense mechanism, each sensor node generates additional encrypted traffic (also known as dummy messages) which are ignored by the receiving nodes. The presence of dummy messages makes it difficult to identify regular notification messages. However generating cover traffic is a drain on the network's limited power and computational resources. More significantly, there is remains a strong correlation between the sensor's detection event and the corresponding message that is sent to the base station. This could be exploited by a determined attacker to mount a correlation attack - by effecting events that would cause the sensor to transmit detection messages.

An alternative is to reduce the correlation between the two events. By making the time t since detection at which the sensor begins to transmit the detection message non-deterministic, the fundamental assumption that the attacker makes about the environment is invalidated. For our experiments, t was modeled as as a exponential distributed random variable $X = \lambda e^{-\lambda x}$. The mean delay is determined by the rate parameter as $\frac{1}{\lambda}$.

[1]It is unclear, particularly in the border scenario, that this represents an end to the attacker's activity. Captures may not lead to further detention, and an attacker can hire multiple people to carry out the task without assuming much personal risk.

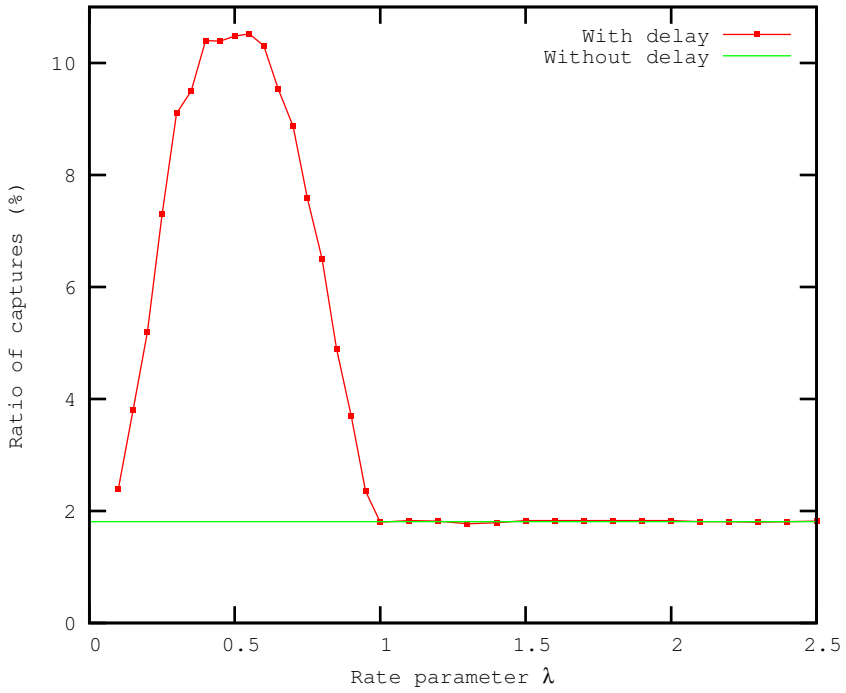

Figure 3.3. Effect of delay.

We define ratio of captures as the number of detections that result in captures. Figure 3.3 identifies the results of varying rate parameters on the average ratio of detection. The addition of random delay to the sensor nodes significantly increases the probability of getting captured and consequently makes the heuristic approach insufficient. The value of λ needs to be carefully chosen, smaller values result in large mean delays in notifying base stations that afford the attacker longer durations to escape. Larger λ values in turn result in a very small mean delays which degreade to deterministic delays. For our simulations, we set the the rate parameter to 0.5.

As described earlier, *connectivity* of a node ensures that a path exists between a node and the sink i.e. the base station. A signal broadcast by a node may require a multi-hop path with intermediate sensors acting as relays. An intruder would have traveled an additional distance in the interval it takes to route the message. Dousse

et.al. study the tracking of moving intruders in presence of time delays introduced in routing messages from sources(sensors) to sinks [22] and suggest an upper bound on the distance traveled by an intruder. The additional delay introduced by the sensors could be accounted for when tracking the position of the intruder.

In the absence of a reliable way to detect the presence of sensors, an intruder might evolve to take into consideration factors such as sensor coverage and observed response times of the sentry to sensor detections. It is difficult to express these factors as parameters to a simple path-finding algorithm. In Chapter 4, we investigate the effectiveness of a learning algorithm as a means to model the intruder.

CHAPTER 4

RL-BASED APPROACH

In this Chapter, we develop a model for a reinforcement learning-based approach and evaluate its effectiveness against the proposed defense.

4.1 RL-based Agent

The agent's world view is identical to the model discussed in Chapter 3. The world is modeled as grid of blocks of unit size. Associated with each block is confidence value σ ($-100 \leq \sigma \leq 100$) that indicates the agent's confidence that a given block is not under surveillance.

Figure 4.1 is a visualization of how the confidence levels might appear after the agent has explored the world. Low-lying contours indicate areas that are certainly under sensor surveillance. Flat lying regions are unexplored areas and high lying regions indicate areas that the agent has sufficiently explored and is confident of being sensor-free.

An episode consists of the agent starting at a location on the border and terminates when the agent retreats, finds a path to the other end or is captured. We make the assumption that, for a given sensor coverage, the path taken by the agent to arrive at that block and the confidence of the block in which it is currently located are sufficient to determine the action of the agent. Episode confidence is defined as the average confidence of all blocks in the path visited by the agent in that episode. Pairs of discretized block confidence and the average discretized episode confidence

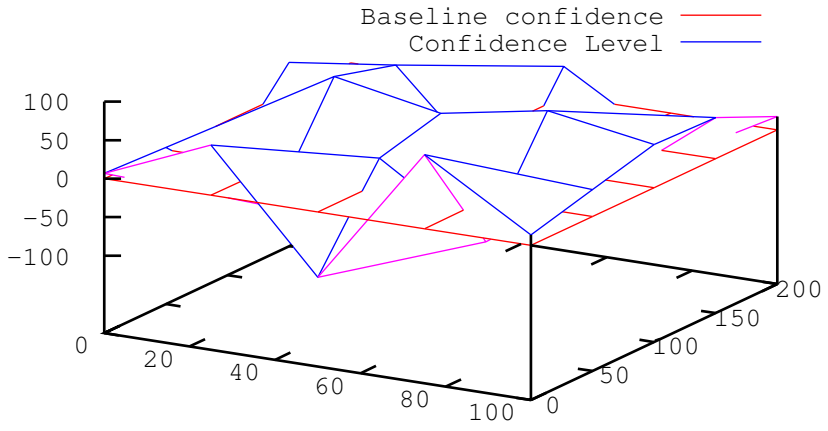

Figure 4.1. Confidence levels after exploration.

form the agent's Q-state value. One additional Q-state represents the goal state and is available when the agent reaches the border.

As the agent moves from one block to another, the rewards the agent receives are

1. The difference in confidence levels of the current block to the next block if none of the other rewards are given. An additional positive reward of 10 is given if the action is *move-up*.
2. A positive reward of 500 for reaching the goal state.
3. A negative reward of -10 for the retreat action.
4. A negative reward of -20 for triggering a sensor.

Figure 4.2 is a visualization of the RL agent's perception of the world. Shades of gray indicate unexplored regions and dark patches indicate regions under sensor surveillance; the sensors are shown only although this is for our benefit and not known to the agent. Various levels of confidence are indicated by shades of green, with increasing confidence indicated by increasing brightness of green.

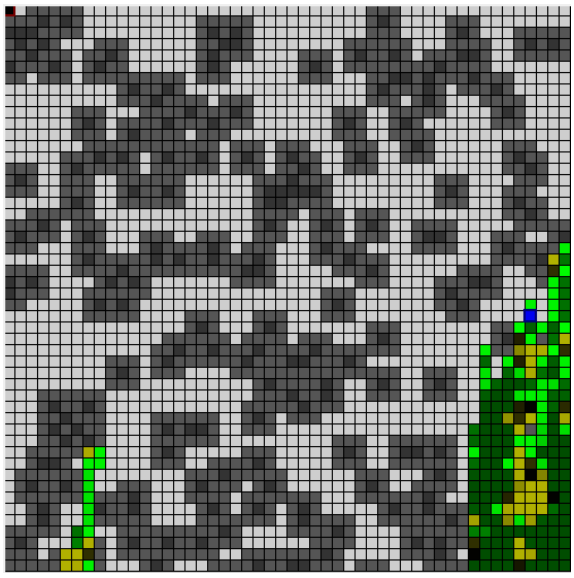

Figure 4.2. A screen capture of our simulation visualization.

4.2 Confidence assignments

Each time a block is visited, the agent increments the confidence amount by 10. When an agent identifies a sensor at a block, the confidence of the block is reduced to the minimum block confidence -100. A block that is not under surveillance is more likely to be surrounded by blocks that are also not under sensor surveillance. The converse is also true. To account for this, we cause confidence assignments to ripple. When we assign a confidence c to a block, the immediate four neighbours of the block are assignment confidences of $\frac{c}{5}$

The ripple behavior of assignments enables the agent to avoid blocks that are likely to be under sensor surveillance without having to visit them. However, it may also incorrectly label narrow paths as under surveillance and avoid it. To correct this, we smoothen sharp gradients at the end of each run. This allows the agent to re-explore blocks that may be sensored or find narrow gaps in regions that were

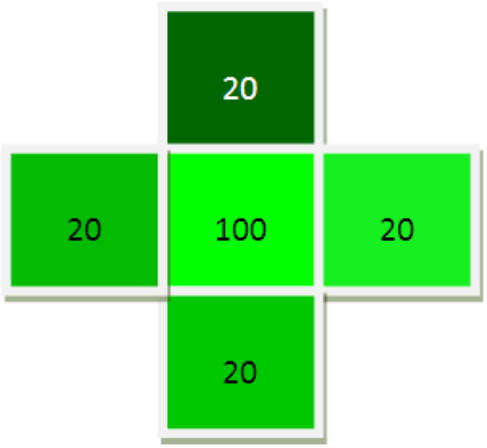

Figure 4.3. Confidence assignment.

previously marked sensored. Our approach to this is similar to an image segmentation problem [23].

Intuitively, a block can be labeled as low confidence if a majority of its neighboring blocks are also low confidence blocks. The reverse also holds true. Unexplored blocks are not considered for this evaluation. For each block $k = (i, j)$ let h_k and l_k be the likelihoods that a block belongs to the high confidence set and low confidence set respectively. We associate a cost p for each block that has a neighbor that belongs to the opposite set.

The problem can be reduced to partitioning blocks into sets $\mathcal{L}$ and $\mathcal{H}$ such that

$$q(\mathcal{L}, \mathcal{H}) = \sum_{m \in \mathcal{L}} h_m + \sum_{n \in \mathcal{H}} l_n - \sum_{\substack{(m,n) \\ m \in \mathcal{H}, n \in \mathcal{L}; |m-n|=1}} p_{mn}$$

The values of h_k and l_k are derived from the discretized confidence assignments. $h_k = |\frac{c_b}{100}|; l_k = 1 - h_k$ if $c_b > 0$. Conversely if $c_b < 0$, $l_k = |\frac{c_b}{100}|; h_k = 1 - l_k$.

For all blocks that the reclassification suggests a value different from the actual assignment, the confidence levels are reinitialized to 0. However, we make exceptions for blocks that are visited and known to be under sensor surveillance.

CHAPTER 5

TESTS AND RESULTS

Two measures are used to gauge the performance of the RL-based algorithm. In both cases, we consider the naive algorithm working in an environment without sensor delay as an oracle model for comparison. All executions were performed using agents that were well-trained over a perimeter of similar node coverage. Values of α and γ for the Q-learning algorithm were chosen to be 0.5 and 1 respectively.

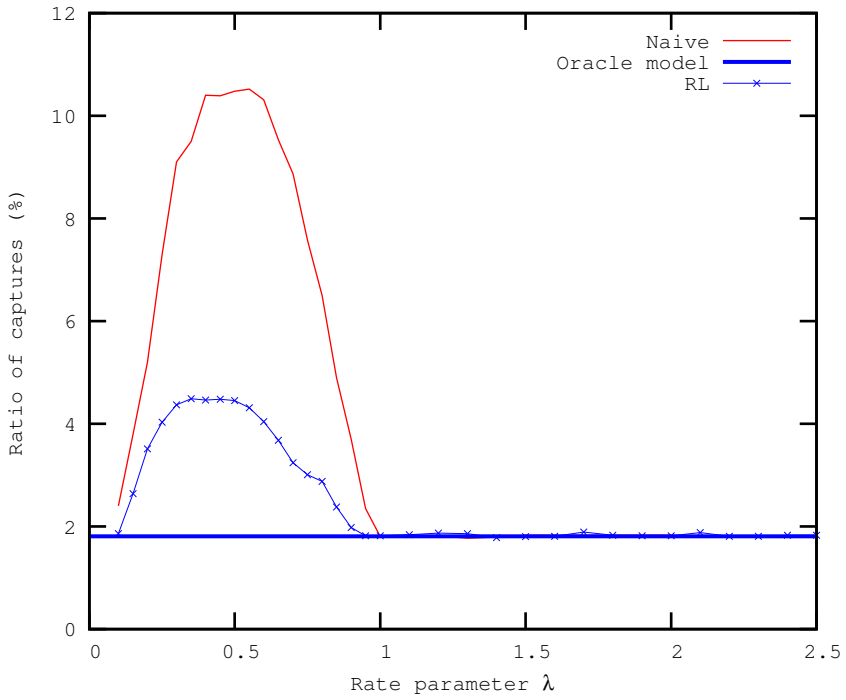

Figure 5.1. Effect of delay on learning agent.

Figure 5.1, measures the effect of delayed trigger on the average capture ratio of the agent. The simulation was performed on a perimeter with 60% node coverage.

The graph is much flatter as compared to the earlier naive results indicating that the percentage of detects that result in captures is largely invariant to sensor delay durations.

Given ample time, the RL-based algorithm always finds paths if they exist, however merely finding a path would not be sufficient, the algorithm needs to identify the path in reasonable time. A good heuristic for reasonable time is the time taken by the oracle algorithm.

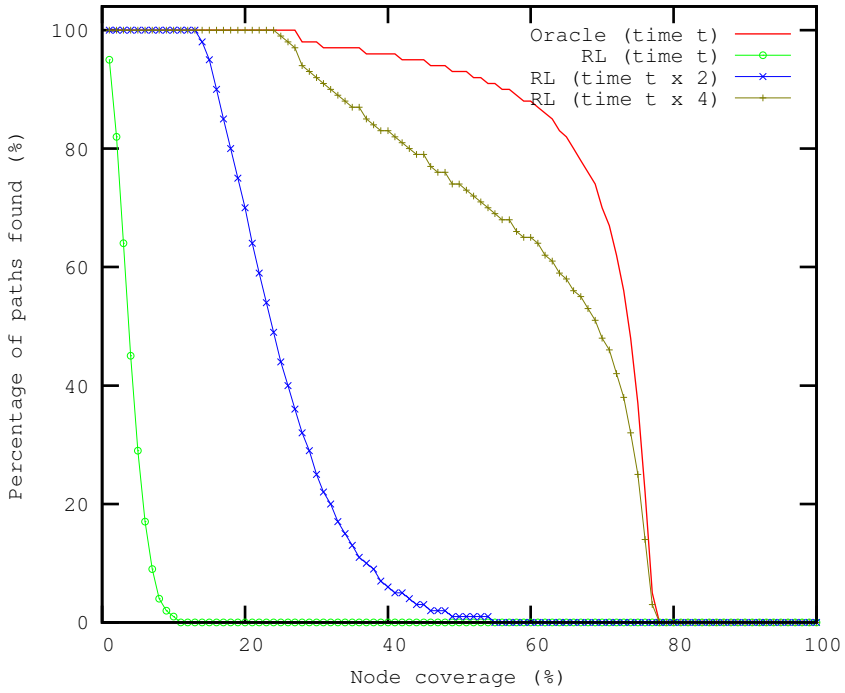

Figure 5.2. Performance of the learning algorithm.

Figure 5.2 indicates the success rate of the RL-based algorithm to determine a path at various multiples of the time taken by the oracle algorithm. At higher node coverages, paths may not exist. This is evident by the failure of the oracle algorithm to identify a path at node coverages greater than 80%. The RL-based

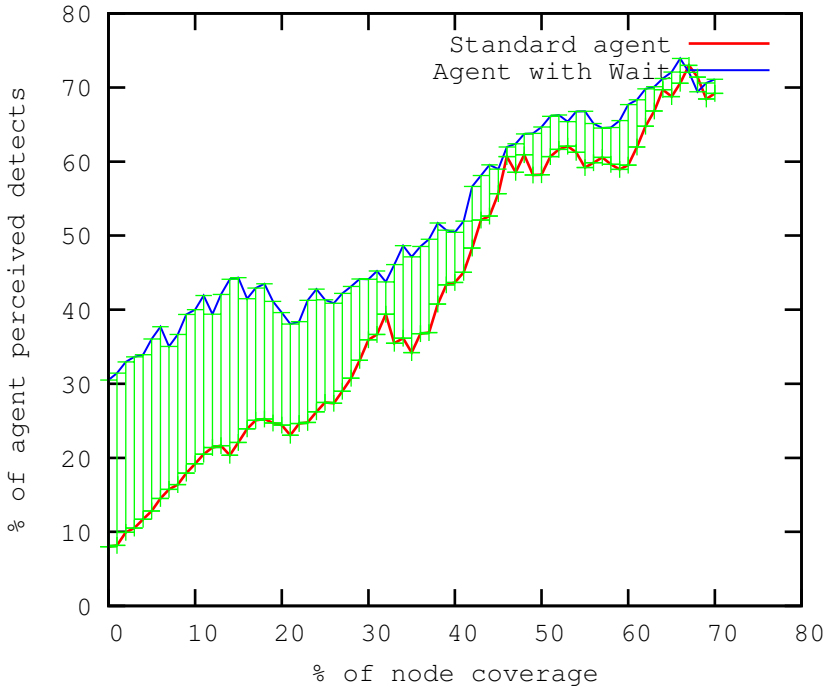

Figure 5.3. Agent with wait.

algorithm fails to identify paths within the same time as the oracle algorithm at all coverages greater than 5% node coverages. Negative reinforcements due to detects and consequently possible captures make the algorithm cautious and prefer retreats when path confidences are low.

At higher coverages, the RL-based algorithm tends to be increasingly cautious, retreating more often and therefore increasing the amount of time it takes to identify paths. It can be observed that allowing the algorithm take longer to detect paths increases the success rate of the algorithm. Allowing it to execute indefinitely results in the algorithm replicating the behavior of the oracle model.

In figure 5.3, we experimented by adding a *wait action* to the agent. The graph compares the percentage of blocks that were identified by the agent as being sensed — i.e. the ratio of the total set of blocks, visited by the agent, that were identified by

it as being under sensor coverage — for the standard agent model described earlier and the agent with the wait action. The new action allows the agent to perform significantly much better in very sparse coverages ($< 30\%$ coverages). However, the difference at higher coverages seems to be insignificant. This could be attributed to the fact that at higher coverage values, a single block is often sensed by more than one node increasing the chances of detecting at least one of them.

CHAPTER 6

FURTHER DISCUSSIONS

As discussed earlier, the agent's understanding of the world solely includes confidence assignments with episode confidence providing it an understanding of the path it took to arrive at a point. This abstraction enables the agent to determine, for a node coverage, optimal values to retreat and to explore the map further. However, the agent's actions are myopic - it makes greedy decisions based on the node coverages of the immediate neighbors without the knowledge or understanding of paths.

An agent that includes the entire grid-world as a part of the state representation would perform optimally. However, even for a very small grid-world - say 50×50 with 4 confidence levels - the size of the state space would be in excess of 39 trillion. Clearly this solution is not very practical.

A path is a series of neighboring blocks that the agent is confident is not under sensor surveillance. A potential path is the deepest path starting from a given point. By including details of potential paths as a part of the state space and rewarding the agent for selecting *deeper* and *safer* paths, we can correct the agent's myopic behavior. We propose an expansion of the state representation of the agent as $\{c_b, c_e, < |\text{path}|, c_{\text{path}} >\}$ where c_b and c_e are block and episode confidences respectively, $|\text{path}|$ is the length of the potential path from that block and c_{path} is the average confidence of the path.

At every transition, we apply a memoized search for all possible paths starting from a position and select the best. Heuristically, (a) longer paths are preferrable

and (b) the role of the confidence of points in the path is inversely proportional to the distance from the origin.

We use the function

$$\text{Value}(path) = |path| * \sum_{\text{block b} \in \text{path}} c_b$$

to assign values to a path and select the best one. To compensate for horizontal (left or right directions) transitions that do not contribute to depth, we select the tile with the minimum confidence as a representative for that row.

This approach builds on our first model for the world and scales linearly with the size of the grid-world.

CHAPTER 7

CONCLUSION

With the increasing importance of wireless sensor networks, physical aspects of the environment become an important concern in security. We propose one such attack model that could be applied to wireless sensor networks and suggest defense mechanisms against this form of attack. We further investigate a reinforcement learning based intrusion model that is effective against the suggested defense.

Ample aspects of modeling the environment and its constraints, as well as studying the effectiveness of alternate learning algorithms could be explored to further the study of this intrusion model.

REFERENCES

[1] A. Bharathidasan, V. An, and S. Ponduru, "Sensor networks: An overview," Department of Computer Science, University of California, Davis, Tech. Rep., 2002.

[2] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, and L. Gu, "Energy-efficient surveillance system using wireless sensor networks," in *In Mobisys*. ACM Press, 2004, pp. 270–283.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comput. Netw.*, vol. 38, no. 4, pp. 393–422, 2002.

[4] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293 – 315, 2003, sensor Network Protocols and Applications. [Online]. Available: http://www.sciencedirect.com/science/article/B7576-499CSFN-7/2/ad3f92c2d573d82839cdb5ae91272fd7

[5] H. Zhang and J. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *Ad Hoc & Sensor Wireless Networks*, vol. 1, no. 1-2, 2005. [Online]. Available: http://www.oldcitypublishing.com/AHSWN/AHSWN 1.1-2 abstracts/Zhang abs.html

[6] Y. Liu and W. Liang, "Approximate coverage in wireless sensor networks," in *LCN '05: Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 68–75.

[7] M. Cardei and J. Wu, "Energy-efficient coverage problems in wireless ad-hoc sensor networks," 2006.

[8] J. H. Li and M. Yu, "Sensor coverage in wireless ad hoc sensor networks," *Int. J. Sen. Netw.*, vol. 2, no. 3/4, pp. 218–229, 2007.

[9] A. S. Tanenbaum, C. Gamage, and B. Crispo, "Taking sensor networks from the lab to the jungle," *Computer*, vol. 39, no. 8, pp. 98–100, 2006.

[10] C. Gamage, K. Bicakci, B. Crispo, and A. S. Tanenbaum, "Security for the mythical air-dropped sensor network," in *Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC 2006)*, 2006.

[11] M. Abadi and J. Jrjens, "Formal eavesdropping and its computational interpretation," 2000.

[12] K. Xing, S. S. R. Srinivasan, M. Rivera, J. Li, and X. Cheng, *Network Security*, 2007, ch. Attacks and Countermeasures in Sensor Networks: A Survey.

[13] *Countermeasures Against Traffic Analysis Attacks in Wireless Sensor Networks*, 2005. [Online]. Available: http://dx.doi.org/10.1109/SECURECOMM.2005.16

[14] T. Mitchell, *Machine Learning.* McGraw Hill, 1997.

[15] P. Langley, *Elements of machine learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995.

[16] N. J. Nilsson, "Introduction to machine learning," draft of Incomplete Notes.

[17] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning.* Cambridge, MA, USA: MIT Press, 1998.

[18] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: http://dx.doi.org/10.1007/BF00992698

[19] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge,England, 1989.

[20] S. Dejmal, A. Fern, and T. Nguyen, "Reinforcement learning for vulnerability assessment in peer-to-peer networks," in *IAAI'08: Proceedings of the 20th national conference on Innovative applications of artificial intelligence*. AAAI Press, 2008, pp. 1655–1662.

[21] X. Xu and T. X. 0005, "A reinforcement learning approach for host-based intrusion detection using sequences of system calls," in *ICIC (1)*, 2005, pp. 995–1003.

[22] O. Dousse, C. Tavoularis, and P. Thiran, "Delay of intrusion detection in wireless sensor networks," in *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2006, pp. 155–165.

[23] J. Kleinberg and E. Tardos, *Algorithm Design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

BIOGRAPHICAL STATEMENT

Pranav Krishnamoorthy was born in Mumbai, India, in 1984. He received his Bachelors of Engineering degree from Mumbai University, India in 2006. Between 2006 to 2008, he was with the .NET Competency Center at Infosys Technology Limited as a Software Engineer. He has been a part of the iSec research lab at University of Texas at Arlington from 2009. His research interests include anonymity systems and programming languages.