JAMMER FORENSICS: LOCALIZATION IN PEER TO PEER NETWORKS BASED ON Q-LEARNING

Ying Liu, Wade Trappe

WINLAB, Electrical and Computer Engineering
Rutgers University
Email:$\{yingliu, trappe\}$@winlab.rutgers.edu

ABSTRACT

Jamming attacks are a class of network denial of service attacks that can easily be carried out in wireless networks. In order to be able to repair a network in the presence of such attacks, it is desirable to identify the location of jammed nodes and the congested area that is affected by the jammer. In this paper, we propose the design of a Q-learning based attack-localization algorithm that is integrated with the OLSR routing protocol. Our Q-learning attack-localization algorithm is distributed, asynchronous and can identify the location of the jammer in run-time as the attack takes place. We examine the performance of our approach using NS3 network simulations under two different network topologies, and for both naive and intelligent attack scenarios.

Index Terms— Q-learning, jammer location, distributed, asynchronous, peer-to-peer networks

1. INTRODUCTION

Wireless communications are easily be subjected to interference attacks that reduce the effectiveness of the network and its protocols to reliably deliver data from a source to a destination. Interference attacks can be performed in many different ways, ranging from emitting a high energy interference signal, which degrades physical layer capabilities to decode modulated information, to the intentional use of MAC-layer congestion that prevents nodes from transmitting or receiving [1]. One of the challenges that must be addressed to ensure that networks can survive in the presence of malicious interference or congestion is to locate the jamming source [2, 3]. Once the location of the adversary has been determined, then remedies may be applied to ensure the network can operate reliably. For example, the location of an adversary may be fed to network layer (e.g. routing) functions and used to alter routes in the network, or the location of an adversary may be used to adjust power or channel allocations for network nodes near the adversary.

Localizing a moderately high-power interference source is a problem that has been relatively well-studied [4–9]. It must be realized, however, that an attacker need not apply large amounts of power to adversely impact the network's performance. In fact, a small amount of transmit power being employed by a congestion-style interference source, which continually emits format-compliant packets, can be quite effective at shutting down the MAC-layer functionality of neighboring nodes in a network. The implication of this observation is that simple schemes, such as those that employ received signal strength (RSS), have limited ability to locate an attacker. For such attacks, it is necessary to analyze the collection of network statistics *across* the entire network in order to locate the adversary. These network statistics are dynamic *signals* associated with the network's graph that provide forensic evidence regarding the adversary's presence and must be cleverly leveraged in order to locate the adversary.

In this paper, our goal is to localize the legitimate network node(s) that are closest to the source of congestion, i.e. the attacker. Our approach operates in a distributed fashion, and involves an online learning algorithm that has been tailored to locating a congestion-style adversary in run-time. Specifically, we have modified the well-known Q-learning algorithm, and the advantages of our method are: (1) the algorithm is distributed (Q-learning runs locally at each network node, each node only needs to know network statistics for itself and its one-hop neighbors); and (2) the algorithm is asynchronous and can converge even if some data available is missing or old (as long as the obsolete information eventually vanishes with time).

The paper is organized as follows: In Section 2, we provide an overview of our problem and the network scenario. In Section 3, we briefly overview Q-learning and then describe how Q-learning was integrated with a routing protocol to support network forensics, including detection and isolation of the jamming attack. We next analyze the performance of our jamming localization algorithm in Section 4 through simulations involving different attacker models. In Section 5 we summarize related work, and conclude the paper in Section 6.

2. PROBLEM OVERVIEW

In this section, we describe the basic problem that we are considering by introducing the network scenario and the adversary's characteristics. We consider a network that consists of nodes that communicate via wireless communications with each other. Depending on parameters, such as transmission power and modulation format, each node might only be able to observe communications from a limited amount of the full set of nodes. Our network, thus, will be arranged as a graph where the vertices correspond to wireless nodes, and the edges correspond to wireless links connecting nodes that are within radio range of each other. In our study, we have used two different motivating topologies: a regular grid deployment, and a deployment where the nodes are randomly placed. We illustrate these two topologies in Figure 1.

Complementing the topological configuration of the network, it is necessary to employ an appropriate networking protocol that manages the routing of communication between nodes in the network. In our study, we have chosen to use the OLSR routing protocol [10] at the network layer, while we employed an 802.11 MAC with RTS/CTS turned off.

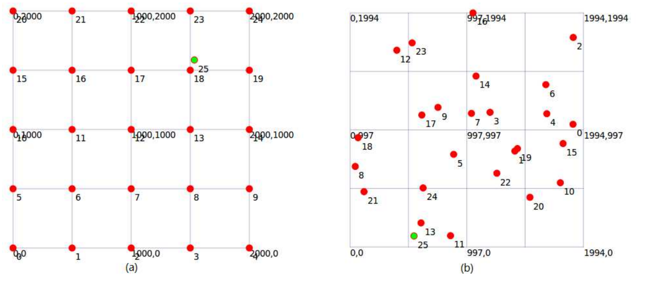


Fig. 1. Grid and random topologies, red nodes are legitimate members. Green is the attacker, which broadcasts valid packets to surroundings.

Our objective is to develop a network forensics tool that locates of a "jamming" node that is attacking the network. Specifically, our attacker is a congestion-style jammer that *continuously* broadcasts format-compliant packets, which results in the loss of legitimate traffic (packets from other nodes). In particular, for victim nodes (i.e. nodes near the jammer), the attack has two immediate consequences: (1) the drop of legitimate packets at the physical and/or MAC layer of a legitimate node; (2) the increased proportion of total control messages to data packets in the network as the routing protocol must send out more control messages to maintain its connectivity. In this study, we assume that the malicious attacker is targeting a specific legitimate node, and hence that node will experience more degradation than other nodes. In this study, we will also consider two variations of this attacker: a naive jammer who simply injects blocking packets, and an intelligent jammer who has infiltrated the network and sends blocking packets as well as false information into the attack localization algorithm. For the intelligent jammer, the attacker is an insider to the network and can announce format-compliant messages that will be interpreted as if he is a legitimate member of the network.

Detecting and isolating the attacker will require forensics upon signals being created at each node that are associated with network traffic statistics. We have adopted the use of Packet Loss Rate (PLR), which captures the proportion of packets at a receiving node to total packets arriving at a receiving node. The calculation of PLR is based on all packets transmitted by the network, including control and data packets. PLR is a readily available network statistic, but we note that other network statistics, such as delay, may also be appropriate.

3. ATTACKER LOCALIZATION IN A NETWORK VIA Q-LEARNING

3.1. Q-learning Preliminaries

Our approach to localizing an adversary in a network uses a modified version of Q-learning. The standard discounted Q-learning was proposed by Watkins in [11], and an asynchronous and distributed version presented in [12]. In Q-learning, the agent learns an optimal policy from past experience by minimizing or maximizing the expected total discounted reward. The agent first randomly chooses an available action from its action set, then it obtains an immediate reward or penalty. This reward/penalty value will factor into calculating the new Q value for the current state and action pair. The optimal action for the current state is collected by finding the action which achieves the minimum/maximum of Q value among all the state-action pairs. We use the Q learning formulas given in [12]:

1. Calculating Q value (policy evaluation)

$$\widetilde{Q}_{t+1}(s,a) = (1-\alpha_{t+1})Q_t(s,a)+\alpha_{t+1}[R(s,a,s')+\gamma_{t+1}V_t(s')] \tag{1}$$

if $\widetilde{Q}_{t+1}(s,a) \leq V_t(s)$

$$Q_{t+1}(s,a) = \widetilde{Q}_{t+1}(s,a) \tag{2}$$

else

$$Q_{t+1}(s,a) = \alpha_{t+1}\widetilde{Q}_{t+1}(s,a) + (1-\alpha_{t+1})V_t(s) \tag{3}$$

2. Calculating V value (policy improvement)

$$V_{t+1}(s) = \min_a Q_{t+1}(s,a) \tag{4}$$

$$a_{optimal} = arg\min_a Q_{t+1}(s,a) \tag{5}$$

In our scheme, each wireless node will run its own Q-learning algorithm, and the Q-values at a particular node will correspond to the packet loss rates associated with the links between that node and each of its neighbors. The internal state within each node's Q-learning algorithm will be an assessment as to *which of its neighbors that node believes to be in the direction of the attacker*. With each node in the network running its own Q-learning algorithm, the collection of assessments can be thought of as a collection of fingers pointing in the direction of the adversary, and thus the final stage of the attacker localization algorithm is to collectively examine these assessments to infer where the adversary is located. Before proceeding to the specifics of our proposed Q-learning algorithm, we note that one of the advantages for using the Q-learning approach is that the Q-value within the algorithm is a discounted, expected value of the PLR, and hence naturally incorporates data smoothing to mitigate bursty PLR fluctuations that naturally arise in the operation of a network.

3.2. Q-learning Integrated with Routing Protocol

Our approach to identifying the jammer location integrates Q-learning with the underlying routing protocol. Routing protocols, such as OLSR, include unused message fields in broadcast control messages that may be utilized to convey additional information between nodes. Specifically, in order to support the dynamic operation of Q-learning at each node, we need the following information to be exchanged: a node's ID; the time slot for a Q value; the time slot for a V value; $V(self\_ID)$; PLR; and the optimal action (i.e. a decision as to which neighbor is in the direction of the jammer).

Using the distributed Q-learning framework of [12], we integrated Q-learning related messages into the routing protocol by attaching them to `hello` messages. These messages are propagated to all one hop neighbors. When its neighbor receives those messages, it detaches the Q-learning messages and stores them in a corresponding V table in reverse time order for the calculation of future Q values. For each time slot, the calculation function of Q learning is called, which searches the old Q value and V table to decide the optimal action (i.e. decision) for next time slot. Pseudocode for the procedure running on each node is as follows:

The receiving node stores the above information into V and PLR lists. The new Q value was calculated iteratively by equations (1), (2) and (3) until it converged in each node. Based on our simulation experience, this process takes $O(N)$ rounds, where N is the total number of nodes in the networks, and we are currently working on a proof for this conjecture. The asynchronous feature of the form of Q-learning we employ is well-suited for congestion

Algorithm 1 function Q-learning

```
initialize $Q_0$, $V_0$
get neighbors N from routing table
calculate new $\gamma_t$, $\alpha_t$
/*policy evaluation*/
for all N in time slot t do
    search $Q_{t-1}(s,a)$ in $selfQNTable$
    repeat
        search available $V$ in $neighborVCTable$
    until find one
    calculate $Q_t(s,a)$
    $selfQNTable.push\_back(Q_t(s,a))$
    if $Q_t(s,a) < minQ$ then
        $minQ = Q_t(s,a)$
    end if
    recalculate cost (packet loss ratio)
end for
/*policy improvement*/
if every k time slots then
    $V_t = minQ$
    $a_{t,optimal} = argmin_a Q$
end if
```

cases as it allows the old neighbor V value in equation (1) to remain (though obsolete) in case there is a loss of information during transmission. Obsolete information can fade out with time due to the discount factor γ. As long as the attacked node can receive and send a little information out, other legitimate nodes can utilize this to find the node nearest to the jammer.

Algorithm 2 function Sendpacket

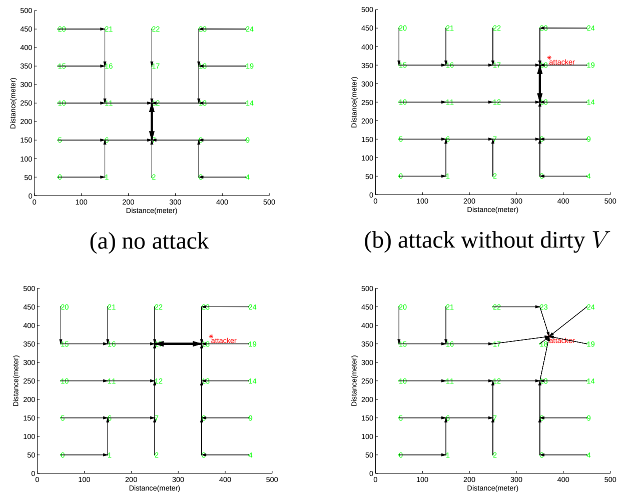
```
if $selfQNTable$ is not empty then
    send $\{selfV_t, PLR_t, ID, time\,stamp\}$
    send $a_{t,optimal}$
end if
```

Algorithm 3 function Receivepacket

```
$neighborVCTable.push\_back(V_t, PLR_t)$
if neighbour's $a_{t,optimal}$ is self id and self's $a_{t,optimal}$ is neighbor id then
    if self's $V_t <$ neighbor's $V_t$ then
        mark self as the target node
    end if
end if
```

4. SIMULATION

We evaluated our algorithm using the NS3 simulator with two 25-node networks as presented in Fig. 1. We employed the OLSR routing protocol with an 802.11 MAC where RTS/CTS was turned off in MAC layer. In our simulations, the nodes were static. The duration for each simulation was 100 time units. In all cases, the attacker begins to broadcast packets starting at time 20 and continuing until the end of the simulation. The attacker's traffic pattern was created to follow a Poisson distribution with inter-arrival time, 0.09 time units, and the length of the attacker's blocking packet was 95 bytes. In each time slot, the PLR was estimated and used to calculate the internal rewards in our Q-function, while the actual Q-values were updated every 2 time units and were estimated in a distributed fashion by each node. Every 6 time units, the V values were updated in a distributed manner. The learning rate was set to be less than 1 and decreased with time. We conducted simulations under different scenarios: (1) there was no attack; (2) there was a naive attacker; (3) there was an intelligent attacker that introduced large V and Q values; and (4) there was an intelligent attacker who introduced small V and Q values. These four scenarios were examined both in the grid and random topologies.



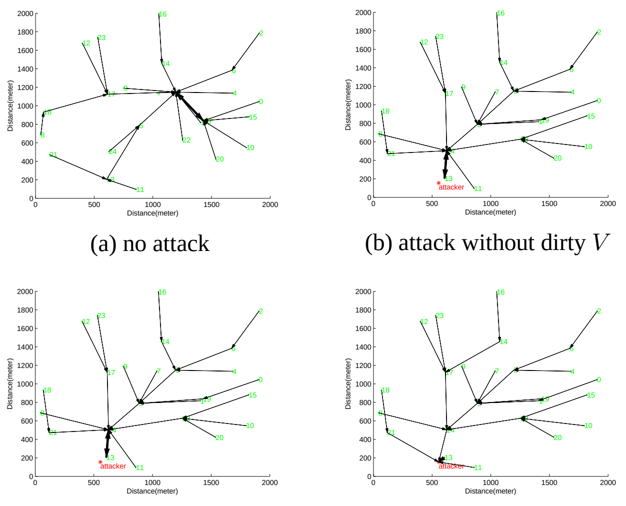
(a) no attack (b) attack without dirty V

(c) attack with large dirty V (d) attack with small dirty V

Fig. 2. Jamming localization in the grid topology. Bold double-arrow link indicates network belief that the jammer is between the arrow endpoint nodes.

Fig. 2 provides the final optimal policies for each node in the grid network under the four scenarios mentioned above. For each node in the figure, there is an arrow that points to the neighbor that node *believes* is in the direction of the attacker. We see that in all cases our Q-learning jammer-locator algorithm was able to find the attacker, or the closest node to the attacker. One interesting phenomena that we observed is that the distribution of V-values exhibits increased variance when the network is under attack, which intuitively occurs because the attack disrupts the behavioral balance associated with nodes in the network. We also examined the case of the random topology. The results in Fig. 3 exhibit similar behavior to the case of the grid topology.

Lastly, we note, as with any such detection scheme, our approach will experience difficulty in differentiating between benign causes of congestion (such as the convergence of many high-traffic flows) and the congestion introduced by an intentional jammer. In particular, it is possible to construct high-traffic flow cases converging at a specific node that will lead to significant packet loss, in which case our algorithm identifies the network node closest to the worst region of benign congestion. Further, the ability of our scheme to identify and locate a jammer is tied to the inter-arrival time between the jammer's emitted packets, and a jammer can avoid detection by increasing its inter-arrival time. Nonetheless, in our experiments, we have witnessed that even so, our Q-learning approach locates the *region* around the jammer. One approach we are exploring



(a) no attack (b) attack without dirty V

(c) attack with large dirty V (d) attack with small dirty V

Fig. 3. Jamming localization in the random topology.

integrating into our algorithm is a traffic-control method whereby nodes in the network adjust their outgoing traffic rate when the network suspects there is a jammer present. Such adjustment would reduce the likelihood of congestion being falsely declared as jamming by our forensics algorithms.

5. RELATED WORK

To find the location of an attacker, or the node under attack, is important and can serve as the basis for repairing the network [13–15]. Previous work on finding the area that is jammed [16–18] can be classified into three categories. The first category involves measuring the received signal strength and using propagation modeling to estimate distance. For example, [19] utilized the variation of the hearing range of a node under attack and its affected neighbors to formulate equations from which the jammer can be localized. The second category obtain the jammer's location by utilizing information about the geometric location of affected nodes. The typical examples for this category are the conventional Centroid Location (CL) [20], Weighted Centroid Localization (WCL) [21] and its improved versions, Virtual Force Iterative Localization (VFIL) [3] and Double Circle Localization (DCL) [22]. Unfortunately, both the first and second categories require the exact locations of neighbor nodes. The third category, however, uses the *relative* position of nodes in the network, and one example of this category is [23], which uses gradient descent to find the node with minimum Packet Delivery Ratio (PDR). One unfortunate behavior of such an approach is that it may be trapped in local minima during the searching process. The approach presented in this paper belongs to the third category. Compared to [23], our scheme is distributed and can be executed in runtime during an attack. Our algorithm can adapt to environmental changes because nodes with historically large PLR can be revisited when their PLR becomes small. In particular, our Q-learning approach bases its decision on the expected PLR and sum of its discounted histories instead of instantaneous values, thus the optimal decision is not easily subjected to the bursty nature of network statistics, and consequently exhibits algorithmic stability.

6. CONCLUSION

In this paper, we examined the problem of locating the source of a jamming or congestion attack against a wireless network. We proposed integrating reinforcement learning (specifically, the popular Q-learning algorithm), with a network routing protocol to arrive at a distributed forensics algorithm that processes signals associated with network statistics to infer the location of a jamming event. The network signals we utilized were local estimates of packet loss rates as made by each node in a network. Our algorithm was validated through NS3 simulations under two different network topologies, and for both naive and intelligent attack scenarios. In particular, we found that our algorithm could reliably identify the location of an intelligent insider-attacker who both emits blocking packets and introduces false Q-learning information into the network. As part of our ongoing work, we intend to explore adaptively tuning our attack localization algorithm according to dynamic conditions within the network, such as node mobility (both the network and attacker's mobility) as well as under varying network traffic conditions.

7. REFERENCES

[1] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2005, pp. 46–57.

[2] A Wood, John A Stankovic, and Sang H Son, "Jam: A jammed-area mapping service for sensor networks," in *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*. IEEE, 2003, pp. 286–297.

[3] Hongbo Liu, X Wenyuan, Yingying Chen, and Zhenhua Liu, "Localizing jammers in wireless networks," in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–6.

[4] Guolin Sun and Jaap van de Beek, "Simple distributed interference source localization for radio environment mapping," in *Wireless Days (WD), 2010 IFIP*. IEEE, 2010, pp. 1–5.

[5] Reza Monir Vaghefi, Mohammad Reza Gholami, R Michael Buehrer, and Erik G Strom, "Cooperative received signal strength-based sensor localization with unknown transmit powers," *Signal Processing, IEEE Transactions on*, vol. 61, no. 6, pp. 1389–1403, 2013.

[6] Lanxin Lin, Hing-Cheung So, and YT Chan, "Received signal strength based positioning for multiple nodes in wireless sensor networks," *Digital Signal Processing*, vol. 25, pp. 41–50, 2014.

[7] Hannan Lohrasbipeydeh, A Gulliver, and Hamidreza Amindavar, "Blind received signal strength difference based source localization with system parameter errors," 2014.

[8] Robin Wentao Ouyang, AK-S Wong, and Chin-Tau Lea, "Received signal strength-based wireless localization via semidefinite programming: noncooperative and cooperative schemes," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 3, pp. 1307–1318, 2010.

[9] Patrik Moravek, Dan Komosny, Milan Simek, David Girbau, and Antonio Lazaro, "Energy analysis of received signal strength localization in wireless sensor networks," *Radioengineering*, vol. 10, no. 4, pp. 937–945, 2011.

[10] Thomas Clausen, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, Laurent Viennot, et al., “Optimized link state routing protocol (olsr),” 2003.

[11] Christopher John Cornish Hellaby Watkins, *Learning from delayed rewards.*, Ph.D. thesis, University of Cambridge, 1989.

[12] Dimitri P Bertsekas and Huizhen Yu, “Distributed asynchronous policy iteration in dynamic programming,” in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*. IEEE, 2010, pp. 1368–1375.

[13] Wenyuan Xu, Ke Ma, Wade Trappe, and Yanyong Zhang, “Jamming sensor networks: attack and defense strategies,” *Network, IEEE*, vol. 20, no. 3, pp. 41–47, 2006.

[14] Wenyuan Xu, Timothy Wood, Wade Trappe, and Yanyong Zhang, “Channel surfing and spatial retreats: defenses against wireless denial of service,” in *Proceedings of the 3rd ACM workshop on Wireless security*. ACM, 2004, pp. 80–89.

[15] Guevara Noubir, “On connectivity in ad hoc networks under jamming using directional antennas and mobility,” in *Wired/Wireless Internet Communications*, pp. 186–200. Springer, 2004.

[16] Yu Seung Kim, Frank Mokaya, Eric Chen, and Patrick Tague, “All your jammers belong to uslocalization of wireless sensors under jamming attack,” in *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 949–954.

[17] Tianzhen Cheng, Ping Li, and Sencun Zhu, “An algorithm for jammer localization in wireless sensor networks,” in *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*. IEEE, 2012, pp. 724–731.

[18] Donggang Liu, Peng Ning, An Liu, Cliff Wang, and Wenliang Kevin Du, “Attack-resistant location estimation in wireless sensor networks,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 11, no. 4, pp. 22, 2008.

[19] Zhenhua Liu, Hongbo Liu, Wenyuan Xu, and Yingying Chen, “Exploiting jamming-caused neighbor changes for jammer localization,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 3, pp. 547–555, 2012.

[20] Nirupama Bulusu, John Heidemann, and Deborah Estrin, “Gps-less low-cost outdoor localization for very small devices,” *Personal Communications, IEEE*, vol. 7, no. 5, pp. 28–34, 2000.

[21] Jan Blumenthal, Ralf Grossmann, Frank Golatowski, and Dirk Timmermann, “Weighted centroid localization in zigbee-based sensor networks,” in *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*. IEEE, 2007, pp. 1–6.

[22] Tianzhen Cheng, Ping Li, and Sencun Zhu, “An algorithm for jammer localization in wireless sensor networks,” in *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*. IEEE, 2012, pp. 724–731.

[23] Konstantinos Pelechrinis, Iordanis Koutsopoulos, Ioannis Broustis, and Srikanth V Krishnamurthy, “Lightweight jammer localization in wireless networks: System design and implementation,” in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. IEEE, 2009, pp. 1–6.