Pinball Attacks: Exploiting Channel Allocation in Wireless Networks

Janiece Kelly
Dept. of Computer Science
Texas State University
jek44@txstate.edu

Mina Guirguis
Dept. of Computer Science
Texas State University
msg@txstate.edu

George Atia
Dept. of Electrical Engineering
and Computer Science
University of Central Florida
george.atia@ucf.edu

***Abstract*—As wireless networks continue to grow rapidly denser with the introduction of various wireless-enabled elements, signal interference coupled with limited radio spectrum availability emerges as a significant hindrance to network performance. In order to retain high network throughput, channels must be strategically assigned to nodes in a way that minimizes signal overlap between neighboring nodes. Current static channel assignment techniques are intolerant of network variations and growth, but flexible dynamic techniques are becoming more feasible with the introduction of software defined networks and network function virtualization. As network maintenance tasks are increasingly handled by software, however, network stability becomes susceptible to malicious behavior. In this paper, we adopt an attacker's prespective and expose stealthy attacks – which we coin "pinball" Attacks – that aim to trigger unnecessary channel switching behavior in a network and increase signal interference between neighboring nodes. We develop a Markov Decision Process (MDP) framework and investigate suboptimal attack policies applied to a number of real-world topologies. We derive attack policies as approximate MDP solutions due to the exponentially large state space. Our results show that pinball attack outperforms other attack policies such as Denial of Service, Random, and other heuristic policies.**

I. Introduction

Motivation: Wireless networks are experiencing an explosive growth rate owing to the continuous integration of various wireless-enabled devices, improvements in network capabilities and under-represented popularity of web-based communication that supports mobility. Accompanying this major growth in wireless networks is a dramatic shift in the data traffic profile. Bandwidth-thirsty applications such as video streaming and various cloud services increase the need for faster, higher capacity networks. Mobile traffic grew 81 percent in 2013 and is expected to grow an additional 11-fold by 2018 [1]. As more devices and heavier traffic push the boundaries of network capabilities, it becomes crucial to develop automated methods to monitor, adapt and improve the performance of large scale networks. Due to the direct control such methods would have over wireless networks and the immediate effect any changes would have on network stability, examining their susceptibility to malicious behaviors is very important.

Challenges: Signal interference that is caused by the reuse of overlapping channels in the radio spectrum is one of the main challenges facing wireless networks due to the limited spectrum. Signal interference can be classified into two types: (1) Co-channel interference (CCI), also called crosstalk, occurs when nodes within each others interference radius are using the same channel. CCI increases network delay due to medium contention overhead, as nodes must wait until the channel is clear before transmitting data, and (2) Adjacent channel interference (ACI) occurs when nodes within each others interference radius are using adjacent overlapping channels. ACI causes destructive interference, since any signals sent by one node will be viewed as noise by nearby nodes on adjacent overlapping channels, leading to packet loss and poor quality of service to the network users.

The problem of signal interference can be at least partially alleviated through channel assignment techniques (e.g., Static Channel Assignment (SCA) and Dynamic Channel Assignment (DCA)), which aim to carefully allocate channels to nodes in such a way that adjacent nodes use non-overlapping channels. SCA methods are inefficient for networks with frequent variations, whereas DCA methods are more flexible but require nodes to be interference-aware and able to switch channels quickly. Typically, this is done through Software Defined Networks (SDNs) and Network Function virtualization (NFV) leading to more stable, highly efficient network.

Although abstracting time-consuming network tasks to software does make the SDNs more reactive, this self-resolving behavior also makes the network susceptible to attacks aimed at hijacking and manipulating network functions. An attacker can easily trigger abnormal switching behavior by inducing a conflicting channel causing the victim to switch (and possibly triggering a cascading effect). This type of channel switch attack has been the focus of some previous studies [2]. In currently used switching procedures, an access point must perform a channel availability check for some amount of time prior to switching, then broadcast an 802.11h channel switch announcement, and finally switch channels. This channel switch process takes 224 microseconds in hardware, but at Layer 2 or 3 it can take more than 320ms [3], [4]. The access points' clients lose connection during the switch and must reconnect, so unnecessary switching can severely impair service. Excessive channel switching based on nonexistent conflicts adds to network latency and could potentially prevent the network from ever resolving. In addition to frequent switching, an attacker could further impair the network by intelligently tailoring an attack to trick nodes into switching behavior that actually increases the number of channel conflicts and worsens signal interference.

Contributions: While abstracting network functions – such as dynamic channel switching – from hardware to software can reduce interference in SDNs, the security and integrity of the network, however, can be jeopardized by a malicious attacker. In this paper, we make the following contributions:

(1) We identify pinball attacks that target dynamic channel assignment methods with the goal to induce conflicting channels to lure the network into a cascading channel-switching behavior that increases network instability[1]. (2) We develop a Markov Decision Process (MDP) framework through which an attacker identifies *which* node to attack and *what* channel to induce to maximize the damage inflicted subject to an attack cost metric. This is done through an adversary model that is restricted for practical purposes. (3) We apply approximate dynamic programming methods [5] to identify efficient attack policies. Through varying the attack cost metric, we were able to uncover classes of pinball attacks that adjust their decisions based on risk of exposure. (4) We crafted a set of features that provides effective approximation for a high-dimensional problem that would be computationally prohibitive otherwise. (5) The attacks identified, albeit suboptimal, are shown to outperform other attack policies such as random, Denial of Service, etc..., and thus are appealing to attackers.

The rest of the paper is organized as follows. Section II puts our work in context with other related work. In Section III, we present our Markov Decision Process model along with our proposed approximate solution methods to obtain optimal/suboptimal attack policies. In Section IV, we evaluate the performance of our exposed attack policies against other attack heuristics. We conclude the paper in Section V.

II. Related Work

The work in this paper relates to two areas of research: channel assignment mechanisms and their security.

Channel assignment mechanisms: Channels can be assigned statically (SCA), dynamically (DCA) or using a hybrid strategy. The authors in [6] propose a modified SCA algorithm that uses node placement information and the signal-to-interference ratio of the network to design a channel assignment algorithm. The work in [7] proposes an edge-coloring, distributed DCA algorithm for an 802.11b/g network that aims to eliminate both primary interference caused by using one channel to receive signals from two different nodes and secondary interference caused by unintentionally receiving a broadcast signal from a nearby node. Channel assignment can also be casted as an optimization problem. The authors in [8] and [9] use Integer Programming (IP) to obtain a channel reuse pattern based on co-channel interference constraints. As an alternative to centralized assignment methods, [10] considers channel assignment as a constraint satisfaction problem (CSP) and proposes a distributed DCA model. Through a channel state table and applying channel overlap constraints, the CSP is solved efficiently with low computation time.

Security: Channel assignment algorithms have been the subject of various control-channel attacks [11]–[13] and defense mechanisms [14], [15]. In [11], three types of vulnerabilities in channel assignment algorithms that capitalize on attacking the highest loaded channels are exposed. The attacks, however, were not studied from an optimization-theoretic standpoint. The work in [13] shows a number of vulnerabilities in MAC protocols due to selfish cognitive radio users that seek to gain more than their fair share of resources. Three types of attacks against channel assignment mechanisms are illustrated in [12]. The attacks aim to create utilization-based conflicts, link breakage and Denial of Data attacks. Again, the attacks exposed were not optimized. The attacks exposed above are different from this work since they do not consider an attacker who's decisions are based on a Markov Decision Process (MDP) problem, aiming to maximize their net reward (i.e., difference between damage inflicted and attack cost incurred).

[1]The attack resembles the pinball arcade game in which the player strikes different targets with a ball using flippers and tries to keep the ball in play by having each target induces a cascading effect of striking other targets.

III. System Model and Formulation

The network topologies used in this paper represent a static wireless network where each node is a wireless access point (AP) equipped with one single channel radio and each edge connects access points within each other's interference radius.

An N-node network is represented by a graph, $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i\}, i = 1, \dots, N$, is the vertex set of APs, and $\mathcal{E} = \{e_{ij}\}, i, j \in \{1, \dots, N\}$, is the edge set. An edge e_{ij} between the vertices v_i and v_j signifies that both vertices are within each other's interference radius. We let $\mathbf{c}_k$ denote a $1 \times N$ vector containing the channels assigned to each AP at time k, with entries $c_k(i)$ for vertex $v_i, i = 1, \dots, N$. The network topology is represented by a standard $N \times N$ adjacency matrix, where any APs connected by an edge cannot use the same or adjacent channels without experiencing interference. Channels used in the assignment vector $\mathbf{c}_k$ are limited to the set of all usable channels $\mathcal{C}$ in a country as set by country regulations. In the United States there are 11 usable channels in the 2.4 GHz band for an 802.11n network, so in our system $c_k(i) \in \mathcal{C} = \{1, 2, \dots, 11\}$.

A. Channel Assignment Technique

We present a discrete-time, interference-aware dynamic channel assignment technique to reduce channel conflicts in a previously deployed network topology. We assume each AP in the network is aware of all usable channels $\mathcal{C}$ and is able to query neighboring APs to determine which channels are currently in use. We let $\mathcal{N}(v)$ denote the neighbors of node $v \in \mathcal{V}$. Hence,

$$\mathcal{N}(v) = \{u \in \mathcal{V} : e_{uv} \in \mathcal{E}\}.$$

We also define $\delta(v), v \in \mathcal{V}$, as the degree of node v, i.e., the cardinality $|\mathcal{N}(v)|$ of the neighboring set.

In the event of a channel conflict with one or more neighbors, an AP can determine which channels are available for it to switch to so that it is no longer in conflict with its neighbors. An available channel is one that is in the set of usable channels but not in any of the interference sets of the AP's neighbors. An interference set includes all channels that overlap with the assigned channel based on a channel separation constant. For an AP, $v \in \mathcal{V}$, on channel $c \in \mathcal{C}$, the AP's interference set, denoted $\mathcal{I}(v)$ consists of adjacent channels, i.e.,

$$\mathcal{I}(v) \triangleq \{\max(c - \Delta, 1), \dots, c - 1, c, c + 1, \dots, \min(c + \Delta, 11)\} \quad (1)$$

where c is the AP's local channel and Δ is the channel separation constant. For example, for a separation value of 2, a channel c will overlap with channels $c - 2, c - 1, c + 1$ and $c + 2$. For all test cases examined in this paper we used a channel separation constant $\Delta = 2$. The set of channels an AP v can freely switch to is the set difference of usable channels and the union of the interference sets of all its neighbors and is denoted $\mathcal{A}(v)$.

$$\mathcal{A}(v) = \mathcal{C} \setminus \mathcal{B}(v), \tag{2}$$

where, $\mathcal{B}(v)$ is the set of channels that are unavailable to v:

$$\mathcal{B}(v) \triangleq \bigcup_{u \in \mathcal{N}(v)} \mathcal{I}(u). \tag{3}$$

The number of channel conflicts $x(v_i)$ experienced by an AP, v_i, is the number of its neighbors which have interference sets that contain the AP's local channel $c(i)$, i.e.,

$$x(v_i) = \left| \{ u \in \mathcal{N}(v_i) | c(i) \in \mathcal{I}(u) \} \right|. \tag{4}$$

We also define $\mathcal{F}$ as the set of nodes with conflicts in the network, namely,

$$\mathcal{F} = \{ v \in \mathcal{V} : x(v) > 0 \}. \tag{5}$$

During channel switching, a conflicted AP is selected from $\mathcal{F}$ at random and allowed to switch channels. The AP prepares to switch channels by first computing its set of available channels. If more than one channel is available, the AP switches to an available channel at random. If no channels are available, the AP switches to an unavailable channel that is least conflicted, i.e., appears in the smallest number of neighboring interference sets. If more than one channel is "least conflicted" the AP selects one of them at random. The channel switching procedure for an AP is outlined as follows:

```
AP = $v$, where $v$ picked at random from $\mathcal{F}$    (AP to switch)
Find $\mathcal{N}$(AP), neighbors of AP
Find $\mathcal{B}$(AP) = $\bigcup_{u \in \mathcal{N}(\text{AP})} \mathcal{I}(u)$    (Unavailable channels)
Find $\mathcal{A}$(AP) = $\mathcal{C} \setminus \mathcal{B}$(AP)    (Available channels)
if $\mathcal{A}$(AP) = $\emptyset$
 Pick least interfering $c \in \mathcal{B}$(AP)
else
 Pick random $c \in \mathcal{A}$(AP)
```

B. Pinball Attack

An attacker mounting a pinball attack aims to degrade the network's performance by inducing unnecessary channel switching among the nodes causing network instabilities and overhead delays. Since each AP relies on the state of its neighbors to make switching decisions, an attacker can travel to an AP in the network (e.g. by driving to and sitting outside a home in a neighborhood) and broadcast a conflicting channel close to the victim's AP. Neighboring APs aim to resolve the induced conflict through switching to other less-interfering channels, if they are available. Pinball attacks resemble hijacking attacks in which an attacker hijacks an AP and switches its channel to a one that conflicts with neighboring channels. While we do not explicitly consider such hijacking attacks since it is easier to broadcast a channel than to hijack an AP, their attack behavior and outcomes would be similar to pinball attacks.

Launching an attack does not guarantee the system will enter a worse state. One AP may switch channels per state transition, so even if an attacker creates a fake conflict, an AP involved in a real conflict elsewhere in the network may be selected to switch instead of one of the APs affected by the attack. Additionally, if an affected AP is selected to switch it will not necessarily switch to a conflicting channel. For this reason, the attacker must weigh the potential benefit of successfully causing damage against the definite cost of launching an attack. We consider the attack cost to be risk of exposure, which means attack cost scales with network density as measured by node degree. Attacking an AP with a high node degree could possibly damage a large number of surrounding APs, but an attacker in a densely populated area of the network also runs a higher risk of being caught.

C. MDP Formulation

To maximize the damage caused in comparison to the costs incurred, the attacker must programmatically find a damaging attack pattern by maximizing (resp. minimizing) a reward (resp. cost) function. When attacking a victim, a single attacker must travel to be within an attacking radius from the desired victim AP's location in order to launch the attack. The possible victims available to the attacker at each step depend on the attacker's location, so the travel distance is the limiting factor in possible victims. We use the hop distance between APs to represent the travel distance for the attacker.

In a pinball attack, an attacker has two possible course of actions: (1) do nothing and (2) broadcast channel $c \in \mathcal{C}$ at the location of an AP to make it appear as though the AP is using that channel. In this system model, an attacker knows the entire network state including the topology and currently assigned channels, and can compute additional information such as node degree and the hop distance between any two APs. For an N-node network, the state of the system s_k at time k is represented by the tuple

$$s_k = (\mathbf{c}_k\ , n_k,\ b_k),$$

where $\mathbf{c}_k$ is the $1 \times N$ vector containing the channels assigned to each AP at time k as defined earlier, n_k is the AP most recently attacked (if any), and b_k is the number of steps since the last attack. b_k is part of the state, since it is assumed the attacker can increase its attack radius depending on how many consecutive no-attack actions were used. Also, let $u_k = [v, c]$ denote the action of the attacker at time k, which consists of a victim node v and an attack channel $c \in \mathcal{I}(u)$, for some $u \in \mathcal{N}(v) \cup \{v\}$. The system transitions from s_k to a new state s_{k+1} when an AP switches channels, thus modifying the assignment vector. The transition probability is $p(s_{k+1}|s_k, u_k)$.

As the system transitions from s_k to s_{k+1}, the attacker collects an immediate reward $r(s_k, u_k, s_{k+1})$, which is the sum of conflicts in the reached state s_{k+1},

$$r(s_k, u_k, s_{k+1}) = \sum_{i=1}^{N} x_{k+1}(v_i)$$

where the sum is over all the nodes in the graph and $x_{k+1}(v_i)$ is the number of channel conflicts at node v_i at time $k+1$ as defined in (4). The cost $y(s_k, u_k)$ of an attack action $u_k = [v, c]$ is

$$y(s_k, u_k) = \begin{cases} h \cdot d(n_k, v) + \delta(v) & \text{if } d(n_k, v) \leq b_k + 1 \\ \infty & \text{otherwise} \end{cases} \tag{6}$$

where, $d(n_k, v)$ is distance between the last attacked AP, n_k, and the victim node v, h is a constant cost per unit distance, and $\delta(v)$ is degree of the AP. Thus, mounting an attack incurs a cost that takes into account the distance to the node and risk of being discovered. The distance to the AP must be less than or equal to the number of steps b_k that have passed since the last attack plus 1 in order to satisfy the power budget of the attacker, which can only increase the radius of the attack based on the time spent without attacking. The net reward, denoted $g(s_k, u_k, s_{k+1})$, is the difference between the immediate reward and the cost of the attack. Hence,

$$g(s_k, u_k, s_{k+1}) = r(s_k, u_k, s_{k+1}) - y(s_k, u_k). \quad (7)$$

The expected net reward $g(s_k, u_k)$ when transitioning from state s_k to s_{k+1} is

$$g(s_k, u_k) = \sum_{s_{k+1}} p(s_{k+1}|s_k, u_k) \cdot g(s_k, u_k, s_{k+1}).$$

The more conflicts an attacker causes over the transition, the higher the path reward, which is the sum of rewards as the system transitions along a discrete Markov chain. The reward earned during a transition from s_k to s_{k+1} is weighted by the probability $p(s_{k+1}|s_k, u_k)$ of transitioning from s_k to s_{k+1}.

In order to find a favorable tradeoff between reward and attack cost, the attacker must identify a sequence of attack policies $\pi = \{\mu_0, \mu_1, \ldots, \}$, where $\mu_k : \mathcal{S} \to \mathcal{U}$, is the attack policy at time k, i.e., a mapping from the state space $\mathcal{S}$ to the control space $\mathcal{U}$. An attack policy describes the sequence of actions the attacker should take over some time frame for all sequences of system states. An optimal policy should optimize the tradeoff between the reward and attack cost so the attacker can achieve maximum damage to the system at little expense to the attacker. Since this is a discrete time system with observable state, the attacker can solve a Markov Decision Problem (MDP) to select an attack policy. Since this channel switching model has an infinite horizon, we use a discount factor $0 < \gamma < 1$ to weight the potential rewards and bias the attacker towards closer rewards. In particular, let

$$J_\pi(s_0) = \sum_{k=0}^{\infty} \gamma^k \ \mathsf{E}\left[g(s_k, \mu_k(s_k), s_{k+1})\right]. \quad (8)$$

where s_0 is the initial state, and $\mathsf{E}[.]$ denotes the expectation w.r.t. the future states, which are unknown at the times of the decisions. The attacker aims to solve for the optimal policy $\pi^* = \{\mu_0^*, \mu_1^*, \ldots\}$ that maximizes the total discounted expected reward, i.e., solve the optimization problem

$$\max_\pi J_\pi(s_0). \quad (9)$$

The optimal solution $J_{\pi^*}(s_0)$ is the optimal value function $J^*(s_0)$.

D. Suboptimal Attack Policy

We have modeled our system as an MDP in which an attacker uses sequential decision-making to navigate discrete system states by defined state transitions. The optimal solution to this problem is a time-invariant policy μ^* [16] defining the optimal action for the attacker to take at each state, and can be obtained as a solution to the Bellman equation,

$$J(s_k) = \max_{u_k \in \mathcal{U}(s_k)} \{g(s_k, u_k) + \gamma \ \mathsf{E}[J(s_{k+1})]\} \quad (10)$$

where $\mathcal{U}(s_k)$ denotes the set of allowable controls given the current state s_k.

In order to obtain the optimal attack policy μ^*, the attacker would need to solve a set of linear equations to evaluate the expected reward J_μ of a policy μ

$$J_\mu(s) = \sum_{s'} p(s'|s, \mu(s)) \cdot [g(s, \mu(s), s') + \gamma J_\mu(s')], \quad (11)$$

for $s = 1, \ldots, \mathcal{S}$. The summation above is over all states s' reachable from s. Then, the attacker would iteratively improve the policy by solving

$$\bar{\mu}(s) = \arg\max_{u \in \mathcal{U}(s)} \sum_{s'} p(s'|s, a) \cdot [g(s, u, s') + \gamma J_\mu(s')], \quad (12)$$

and then evaluate $J_{\bar{\mu}}$ as in (11). The attacker iterates over policy evaluation (11) and policy improvement (12) until convergence to the optimal policy that maximizes the reward[2].

The optimal policy obtained as an exact solution describes the optimal action the attacker should choose given any state. For a very small network, an attacker could possibly enumerate every state and select actions that lead to states with the best reward, but since this paper focuses on a state space similar to a real-world network with N nodes and $|\mathcal{C}|$ channels, the exact solution is intractable. Even if the maximum number of successive no-attack actions is restricted to K, the state space would be of size $|\mathcal{C}|^N \times N \times K$, which grows exponentially with N. Thus, it is computationally intractable for the attacker to evaluate the expected state reward for every possible state and action combination.

Instead, we develop suboptimal pinball attack policies using Approximate Policy Iteration (API) based on estimations of the state reward. Rather than examining every possible state, API uses representative states and features. Representative states are used as training states during policy improvement, so they must be selected in such a way that they capture most possibilities for the system and give wide coverage of useful regions in the state space. We selected representative states containing a spectrum of possible conflict states, from minimal conflicts or a conflict-free assignment to maximum conflicts with all APs assigned the same channel. All APs may start on any default channel, so an experimental run may begin with any number of conflicts. The minimal size of a single conflict is 2 and the maximal size is $\max_{v \in \mathcal{V}} \delta(v)$, the largest degree of an AP in the network. Since graph-coloring is an NP hard problem, we did not attempt to generate a guaranteed minimal conflict state and instead approximated it using a greedy graph coloring solution.

From the representative states, a set of representative features is extracted that captures the characteristics of the state and can be weighted and used to estimate its value. In a pinball attack, the attacker evaluates an approximated parametric expected state rewards using a weight vector $\mathbf{r}$

$$\tilde{J}_{\mathbf{r}}(s) = \sum_{j=1}^{M} r_j \phi_j(s) \quad (13)$$

[2]Convergence is guaranteed in this case since the transformation defined through policy iteration is a contraction mapping, hence the iterations converge to a unique fixed point.

where M is the total number of features, $\phi_j(s)$ denotes the j-th feature for state s, and $r_j, j = 1, \ldots, M$ (the j-th entry of vector $\mathbf{r}$) the weight of the j-th feature. Instead of iteratively evaluating and improving the policy until the iteration loop naturally terminates, API uses Monte Carlo simulations to evaluate the feature weights over a number of independent trajectories from these representative states. More specifically, the weights vector $\mathbf{r}$ is obtained as a solution to a least-squares problem minimizing the square error between the empirical average reward of these trajectories and the parametric approximation $\tilde{J}_{\mathbf{r}}$ in (13). Then, to compute an improved action for state s, we solve

$$\tilde{\mu}(s) = \arg\max_u \sum_{s'} p(s'|s,u) \left[g(s,u,s') + \gamma \ \tilde{J}_{\mathbf{r}}(s') \right] \quad (14)$$

where the value function is replaced by the approximate parametric form $\tilde{J}_{\mathbf{r}}$.

E. Feature Selection

Pinball attacks rely on a set of representative features to capture the characteristics of each state and approximate the state value. We used 13 features: number of APs in conflict with one or more neighbors in the current state, ratio of maximum number of APs involved the same conflict to degree of the network graph, average number of APs involved in the same conflict, average number of channels unavailable to an AP, average conflict size of the highest degree AP(s), last attacked AP, steps since last attack, flag for whether attacker is at most complex node (MCN), i.e., node with highest degree, degree of last attacked location, conflicts at last attacked location, available channels of last attacked location, degree of largest neighbor, and fraction of APs within hop distance.

IV. Performance Evaluation

A. Model Instantiation

During policy iteration we tested a range of attack costs and values of γ, a channel separation constant of 2 and 35 representative states. We tested a range of network topologies selected to reflect real world topologies. For all topologies we considered the interference radius to be 1, meaning an AP only interferes with other APs located one hop away. Each topology was tested for 20 policy iterations and each iteration simulated 20 trajectories of 50 steps. A random attack was used as a rollout policy. The Monte Carlo simulations were written in MATLAB using the parallel computing toolbox and run using 20 cores on a research cluster with Univa Grid engine.

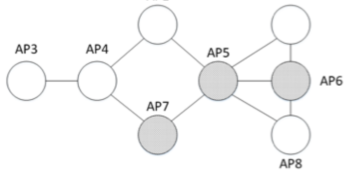


Fig. 1: 8-node topology with a range of node degrees.

B. Attack Comparison

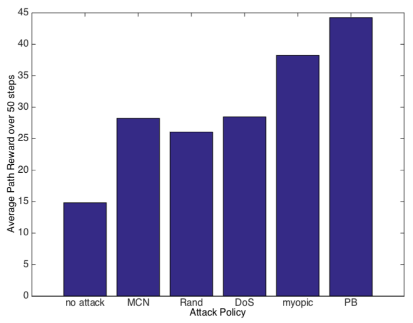
To study the potency of pinball attacks, we compare them to other attack policies by generating another set of representative states and computing the average reward gained under each attack. We compare between the following attack policies: (1) No attack; (2) MCN (most complex node); (3) Random (rand); (4) DoS (Denial of service); (5) Myopic; (6) Pinball (PB). In a no-attack policy, the attacker takes no action at every step. This is the lowest cost policy and serves as a baseline. Since a network may start with conflicts and resolves over time, the attacker may obtain a non-zero path reward without exerting any effort. In a MCN attack, the attacker takes a greedy approach and targets the victim AP with the largest node degree (if multiples exist, the attacker picks the one with the shortest hop distance from its current location). Once at the victim AP, the attacker will ignore any costs incurred and constantly attack at every step with a conflicting channel. In a random attack, the attacker selects any random reachable AP and chooses at random whether to broadcast a channel or do nothing. Since the random policy allows the attacker to withhold attacks at random, the attacker may miss critical attack opportunities and allow the system to resolve. In a DoS attack, the attacker selects any random reachable AP and broadcasts an interfering channel at every step. However, since the victim APs are picked at random the attacker may be expending energy needlessly when there are better victims to attack. In a myopic attack, the attacker selects a victim AP with the highest immediate reward (i.e., does not consider the next state and future rewards).

For each topology, we tested a range of scaling constant values h to observe the attacker's behavior as the attack cost is varied. As expected, the path reward under a no-attack policy remained constant and is used as a baseline. When the attack cost is very low, the attacker's behavior resembles that of a DoS attack. When the attack cost is very high, it resembles a no-attack policy.

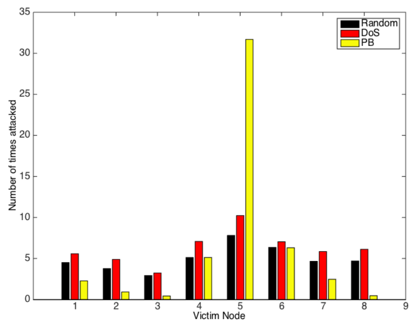
As the attack cost increased, the path reward of DoS and random policies decreased, dropping well below that of no attack for very high attack costs. For all values, even very low attack costs, the path reward of pinball remained higher than the path reward of DoS. The reason pinball attacks could outperform DoS at low attack costs despite launching the same number of attacks is because DoS attempts to maximize damage simply by constantly attacking while pinball maximizes damage by constantly attacking intelligently, taking future states into account in addition to the potential immediate reward.

Over all topologies tested, an attacker following a pinball attack policy was able to achieve a path reward greater than any other attack policy. Due to lack of space, we only present the detailed results for one 8-node topology (shown in Fig. 1).

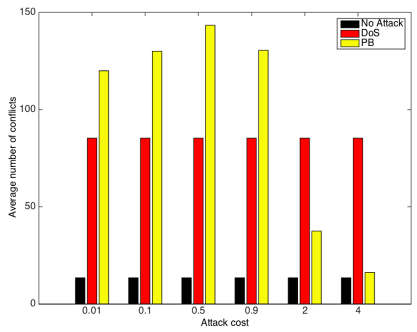
In this case, a single attacker following a pinball attack policy achieved an average path reward 1.5 times greater than a DoS attack (see Fig. 2a). Nearly all myopic attacks were aimed at APs 4 and 5. Under a pinball attack policy, the attacker was more active in the denser side of the network, focusing attacks on APs 4, 5 and 6 as shown in Fig. 2b. The pinball attack approach is smarter because although AP 4 and 6 have the same node degree, AP 6 has more complex neighbors, one of which is the most complex AP in the whole network. Rather than constantly attacking AP 5, the most complex AP, directly by broadcasting the same channel at AP 5, the attacker often targeted lower cost neighbors and broadcasted a channel that overlapped with both AP 5 and



(a) Average path reward for various attack policies with the 8-node topology.



(b) Histogram of victim selection for various attack policies with the 8-node topology.



(c) Total conflicts seen over 50 steps for various attack policies and costs.

Fig. 2: Results from 8-node topology.

other surrounding APs, increasing the potential number of conflicts. The pinball attack outperformed myopic, showing that it is beneficial for an attacker to make decisions based on the features of a state. Although the pinball attack policy was not always able to prevent the system from resolving, it was always able to continually reintroduce conflicts into the resolved system and, in some cases, create even more conflicts than the system originally had. In addition to overall path reward, policy performance can also be evaluated in terms of the total conflicts, number of channel switches, or number of steps until the system resolves. For both the number of channel switches and total conflicts observed in the system *at high attack costs*, pinball attack causes fewer conflicts and switches because the attacker is much more conservative about launching attacks as illustrated in Fig. 2c. Even though fewer conflicts are created, the overall path reward for pinball attack is still higher because the attacker is not constantly incurring massive penalties.

In a 6-node ring toplogy, a pinball attack achieved an average path reward 2.7 times greater than DoS and in a 7-node tree topology, a pinball attack achieved 6.3 times greater average reward path than DoS. These results are not presented in detail here due to lack of space.

V. Conclusion

In this paper, we exposed pinball attacks that target wireless networks employing dynamic channel switching methods. We have shown that optimized pinball attacks can significantly impact the stability of the network by introducing unnecessary channel switching between network nodes in a resolved system and prolonging convergence to a non-conflicting sate. Using an MDP formulation, an attacker can solve an optimization problem to identify the most vulnerable nodes to attack, given different attack costs. Through approximate policy iteration, the attacker can estimate the value of being in a state using a set of features and can obtain an optimal/suboptimal attack policies. Through various topologies that resemble realistic real-world settings, we have shown that pinball attacks can outperform other attack policies such as Denial of Service, random and even other heuristics that may seem appealing such as the most complex node. Furthermore, pinball attacks can adapt their aggressiveness based on the attack, enabling us to identify classes of attacks based on their exposure risk.

Acknowledgement

This work was supported in part by NSF awards CNS-1149397, CNS-1156712, CCF-1320547 and CCF-1552497.

References

[1] Cisco, "Global mobile data traffic forecast update, 2014-2018." [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

[2] B. Konings, F. Schaub, F. Kargl, and S. Dietzel, "Channel Switch and Quiet Attack: New DoS Attacks Exploiting the 802.11 Standard," in *LCN*, 2009.

[3] D. Murray, M. Dixon, and T. Koziniec, "Scanning Delays in 802.11 Networks," in *NGMAST*, 2007.

[4] "Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," IEEE-SA Standards Board, 2003.

[5] D. P. Bertsekas, "Approximate policy iteration: A survey and some new methods," *Journal of Control Theory and Applications*, 2011.

[6] M. Haidar, R. Ghimire, H. Al-Rizzo, R. Akl, and Y. Chan, "Channel Assignment in an IEEE 802.11 WLAN based on Signal-to-Interference Ratio," in *CCECE*, 2008.

[7] R. Akl and A. Arepally, "Dynamic Channel Assignment in IEEE 802.11 Networks," in *IEEE PORTABLE*, 2007.

[8] K. Daniels, K. Chandra, S. Liu, and S. Widhani, "Dynamic channel assignment with cumulative co-channel interference," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 4, 2004.

[9] W. El-Hajj and H. Alazemi, "Optimal Frequency Assignment for IEEE 802.11 Wireless Networks," *Wireless communications and mobile computing*, vol. 9, no. 1, 2009.

[10] S.-Y. Lin, S.-C. Horng, and T.-Y. Chan, "Interference avoidance distributed dynamic channel assignment for cellular network," in *ICSSE*. IEEE, 2011.

[11] A. Naveed and S. Kanhere, "Security vulnerabilities in channel assignment of multi-radio multi-channel wireless mesh networks," in *IEEE Global Telecommunications Conference*. IEEE, 2006, pp. 1–5.

[12] Q. Gu, M. Yu, W. Zang, and P. Liu, "Lightweight Attacks Against Channel Assignment Protocols in MIMC Wireless Networks," in *IEEE ICC*, 2011.

[13] Y. Zhang and L. Lazos, "Vulnerabilities of Cognitive Radio MAC Protocols and Countermeasures," *IEEE Networks*, vol. 27, no. 3, 2013.

[14] L. Lazos, S. Liu, and M. Krunz, "Mitigating Control-channel Jamming Attacks in Multi-channel Ad hoc Networks," in *WiSec*, 2009.

[15] S. Liu, L. Lazos, and M. Krunz, "Thwarting Control-channel Jamming Attacks from Inside Jammers," *IEEE Transactions on Mobile Computing*, vol. 11, no. 9, 2012.

[16] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.