THE EFFECT OF DECOY ATTACKS ON

DYNAMIC CHANNEL ASSIGNMENT

by

Janiece Kelly, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a major in Computer Science
December 2014

Committee Members:

Mina Guirguis, Chair

Daniela Ferrero

Qijun Gu

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

ACKNOWLEDGEMENTS

I would like to extend thanks to my advisor, Mina Guirguis, for introducing me to endless fields of research and continually encouraging me to think differently.

To my thesis committee, Qijun Gu and Daniela Ferrero, for their patience and enthusiasm for this thesis.

To Marty Bylander for going the extra mile to assist me with my endeavor into high performance computing.

To Kwok Wu and Avinash Naidu for always pushing me to learn more, learn faster, and "stay hungry".

To my sister, Patience, for your infectious positivity. You mean the world to me.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure **Page**

ABSTRACT

As networks grow rapidly denser with the introduction of wireless-enabled cars, wearables and appliances, 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 techniques for channel assignment are intolerant of network variations and growth, but flexible dynamic assignment techniques are becoming more feasible with the introduction of software defined networks and network function virtualization. Virtualized networks abstract hardware functions to software, making tasks such as channel assignment much more reactive and suitable for automation. As network maintenance tasks are increasingly handled by software, however, network stability becomes susceptible to malicious behavior. In this thesis, we expose and study the effect of stealthy attacks that aim to trigger unnecessary channel switching in a network and increase signal interference. We develop a Markov Decision Problem (MDP) framework and investigate suboptimal attack policies applied to a number of real-world topologies. We derive attack policies as an approximate MDP solution due to the exponentially large state space. Determining vulnerabilities to stealthy attacks is necessary in order to improve the security and stability of software defined networks.

I. INTRODUCTION

Motivation

Wireless networks are experiencing explosive growth worldwide with the increasing use and diversity of wireless-enabled devices such as the Apple Watch, improvements in network capability such as Google Fiber, the growing popularity of web-based communication through applications like Skype and a shift to streaming entertainment through applications like Netflix. Wireless connectivity has become an integral part of not only personal computers, smart phones and tablets, but also vehicles, wearables, home appliances, and even energy meters as we transition to smart cities and homes. A push towards creating an Internet of Everything (IoE) to connect people, data and things, will add an estimated 155 million wireless devices in wearables alone as people begin to use smart glasses, health monitors, fitness trackers and even pet collars.

In addition to *how* we connect, *why* we connect has also changed dramatically. Data traffic has shifted to more bandwidth-heavy content such as video streaming and cloud services, increasing the demand for faster, higher capacity networks. Mobile traffic grew 81 percent in 2013 and is expected to grow an additional 11-fold by 2018 [9]. As more devices and heavier traffic push the boundaries of network capabilities it becomes more important than ever to develop automated methods to monitor and improve performance in 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, we must first examine how susceptible they are to malicious behavior.

Wireless Network Limitations

One of the main limitations of wireless network capacity and performance is signal interference caused by the reuse of overlapping channels in the radio spectrum. The portion of the spectrum reserved for use by wireless devices is limited, and when divided into channels each channel overlaps with adjacent channels. Signal interference occurs when a node in the wireless network uses a channel that is the same as or adjacent to a channel used by another node within radio range. If too many nodes in a network interfere, the network will experience message retransmission and redundancies, congestion, degraded signal quality and poor quality of service for the network's users.

The problem of signal interference can be at least partially alleviated through channel assignment techniques, which aim to carefully allocate channels to nodes in such a way that adjacent nodes use non-overlapping channels and the signal to interference and noise ratio (SINR) is minimized. When the channel assignment scheme is determined and implemented prior to network deployment, this is called static channel assignment (SCA) and when the assignment is able to change over time it is called dynamic channel assignment (DCA). Both methods have limitations since SCA is inefficient for networks with frequent variations, but the more flexible DCA requires the network nodes to be interference-aware and able to switch channels quickly. Rather than relying on a human administrator to manually monitor and switch channels, it is beneficial to carry out channel switching behavior in software. Software Defined Networks (SDNs) allow tasks like channel switching to be performed in software, and thus are a good platform for interference-aware dynamic channel assignment solutions that could potentially lead to

more stable, high performing networks.

Security

Despite the potential for performance improvements due to network function virtualization, it is important to examine possible shortcomings. Abstracting many of the time-consuming tasks to software does make the network more reactive, but this self-resolving nature also makes the network susceptible to attacks aimed at manipulating and controlling network behavior. In a network employing DCA, for example, each node's normal channel switching behavior attempts to reduce channel conflicts with neighboring nodes by switching to a non-overlapping channel. An attacker can easily trigger abnormal switching behavior, however, by tricking a node into believing there is a channel conflict where one does not exist. 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 from 3-20ms [13, 17]. The access point's 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 causing 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 in the network. Understanding the effect of attacks that exploit software-based network activity is crucial to improving the security of

software defined networks in the future.

Thesis Statement

Abstracting network functions such as channel switching from hardware to software can facilitate interference reduction in software defined networks, but the security of network virtualization must be considered because malicious behavior by an attacker may jeopardize network integrity. In this thesis we (1) explore a dynamic channel assignment technique that reduces channel conflicts between adjacent nodes in an interference-aware network of access points, (2) present a decoy attack that aims to increase network instability by projecting false channel information, (3) develop a Markov decision problem framework that can be approximately solved to obtain suboptimal attack policies and (4) compare the performance of suboptimal decoy attack policies to less intelligent attack policies.

II. RELATED WORK

Background

A wireless network is a collection of nodes that communicate by sending and receiving wireless signals. Nodes in the network communicate using a specific radio frequency spectrum, which is divided into one or more specific channels to allow efficient use of the spectrum. A node can switch between channels as needed and the number of channels a node can simultaneously use depends on how many radio interfaces are present on the node. According to IEEE 802.11 standards, wireless-b, wireless-g and wireless-n networks currently operate between the 2.4 and 2.5 GHz radio frequency range, which is divided into 14 channels.

Wireless network performance can be measured in terms of data rate, bandwidth utilization and packet delivery reliability and is influenced by a number of factors such as received signal strength, congestion, number of hops, and radio range of the nodes [6, 10]. Compared to wired networks, wireless networks have higher bit-error rates, limited bandwidth, high latency, and greater potential for disconnection. Since signal quality relies heavily on environmental factors and can easily be degraded by noise, wireless networks are subject to channel fading and non-Wi-Fi interference from microwaves, satellite television, and other devices in the 2.4 GHz spectrum using Bluetooth or ZigBee.

Network performance metrics include throughput, goodput, and delay. Throughput measures data transferred in bits per second and takes into account transmission overhead, system limitations and unsuccessful transmissions. Goodput

measures useable data transferred, which excludes duplicates and packet overhead such as protocol headers and information used solely for routing. Delays can be incurred from processing, congestion, or characteristics of the transfer medium. Performance can be improved using techniques such as noise filtering, frequency modulation, handoffs, medium access control (MAC), and optimized routing.

Signal Interference

Signal interference is one of the main network factors that adversely affects performance, and is difficult to overcome due to the limited availability of channels. In the United States, channels 1-11 are legal to use. These channels are spaced 5 MHz apart in the 2.4 GHz radio spectrum, and since each channel has a width of 20 MHz there is significant overlap with adjacent channels. Network nodes within radio range attempting to communicate simultaneously on the same or overlapping channels will experience signal interference, resulting in difficulties extracting the intended signal. Interference in a network of single-radio nodes can be classified into two types:

1. Co-channel interference (CCI), also called crosstalk, occurs when nodes within each other's 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.
2. Adjacent channel interference (ACI) occurs when nodes within each other's 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.

A successful transmission by a node requires the receiving device to be within a transmission range and the signal to interference and noise ratio to be below a specific threshold. If interference and noise overwhelm the signal, nodes will experience unsuccessful transmissions, prompting them to resend frames to improve chances of a successful communication. Multiple copies of the same transmission reduce network throughput, causing even more congestion and increasing energy utilization. This can be especially detrimental for remotely deployed nodes such as small cells, as physical access to such nodes to replace batteries is limited and power consumption must be kept to a minimum.

Overall, interference decreases network throughput and goodput, and increases signal latency and bandwidth congestion. As networks are increasingly used for streaming services such as video and VoIP, network latencies and delays are unacceptable as they render a network unusable for certain users and applications. Solutions for improving network performance focus on reducing contention, either through communication protocols or changes in the network topology. Throughput can be improved by implementing multi-hop transmission where messages are buffered at intermediate nodes along a routing pathway. Nodes themselves can be equipped with multiple radios that communicate simultaneously. The radio spectrum can be divided and allocated according to frequency, code or time slots to improve transmission scheduling.

Since throughput decreases as the number of users in a network grows, it may be necessary to limit the number of users and avoid bandwidth overload by implementing

smaller wireless networks or "small cells" [15]. The upcoming 802.11ac standard will broaden the frequency spectrum from 2.4 to 5 GHz, reducing congestion by allowing higher data rates and a greater number of non-overlapped channels [20]. Until a solution for overcrowding of the 2.4GHz band becomes available, channel assignment and reuse remains the best way reduce overall interference and improve network performance and user experience. Channel assignment algorithms target the cause of interference directly and apply a mathematical approach to allocate channels based on various factors such as physical location, signal to noise ratios and radio ranges. This thesis focuses on channel assignment and reducing signal interference by minimizing channel frequency overlap.

Channel Assignment Approaches

An optimal channel assignment strategy would assign every node within an interference radius a different non-overlapping channel, but due to the limited radio spectrum the largest set of orthogonal, or mutually non-overlapped, channels in a wireless-n network still only contains three channels: 1, 6, and 11. An assignment scheme that uses only these three channels eliminates all adjacent channel interference, but almost certainly causes a lot of co-channel interference. Channel reuse takes into account the fact that nodes outside of each other's interference radius can freely use the same or adjacent channels without damaging network performance. The interference radius is determined by how far out a signal propagates from a node, since the signal power decays with distance traveled. An efficient assignment strategy may attempt to first eliminate ACI by assigning non-overlapped channels to neighboring nodes and then reduce CCI by allowing channel reuse among non-neighboring nodes.

For a simple implementation of channel assignment in a network, each node experiencing poor performance can randomly switch channels until it finds one with better performance. This random assignment does not require any computation or knowledge of the network, but is inefficient and potentially time consuming. An alternative implementation is static channel assignment (SCA) whereby the topology of the network is analyzed and used to develop a channel assignment scheme that is implemented prior to network deployment. Graph coloring solutions are a popular method used to determine the channel assignment strategy depending on the network topology and degree of interference. Nodes are represented as vertices in a graph where edges represent adjacency and channels are determined using an adaptation of vertex (or edge) coloring where no two adjacent nodes (or edges) may have the same color or channel. SCA can provide a better assignment than random switching, but channels are allocated based on pre-determined estimations of interference radius and expected traffic and thus may be inefficient if location or traffic variations occur. Once deployed it is difficult to modify the system to adapt to changes in traffic since switching one channel may require resolving and reallocation of the entire network.

Recent research has focused on dynamic channel assignment (DCA), which is an adaptive assignment method that overcomes the limitations of SCA by assigning channels “on-the-fly” in reaction to changes in interference or traffic. In a centralized DCA scheme, channel allocation requests are handled by a central controller with complete system-wide visibility. The controller grants or denies channel requests based

on the calculated effect on signal quality. This centralized approach has the advantage of visibility but the disadvantage of lower responsiveness since the controller must communicate with each node, possibly through a series of repeaters for remote nodes. This centralized approach fails as the network grows in size and the controller experiences an overload of requests.

In a distributed DCA scheme, channel switches are decided on by individual nodes based on their local perception of signal quality and awareness of channels used by surrounding nodes or cells. To initiate a channel switch, a node computes signal quality based on signal strength and signal to noise interference ratio. If quality is perceived to be poor based on a pre-decided threshold, the node will then attempt to switch to a higher quality channel. A distributed channel assignment scheme is more robust and scalable than a centralized scheme, but may also be suboptimal in terms of efficiency since it does not consider the state of the entire system. Distributed DCA may also be unable to adapt to rapid fluctuations in channel demand, necessitating development of dynamic allocation schemes that can respond to spatial and temporal changes.

Channel assignment is a popular research area and there are a number of existing methods to assign channels statically or dynamically. [8] proposes a modified SCA algorithm that uses node placement information and the signal-to-interference ratio of the network to create a channel assignment design. [1] 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. The algorithm computes an overlapping channel interference factor and was able to successfully minimize signal interference in a limited number of iteration cycles. [16] proposes a vertex coloring distributed DCA algorithm for an 802.11a/b/g network, that assigns vertex weights in order to address interference chiefly in areas of greatest need. The switching behavior includes a Least Congested Channel Search heuristic. The authors contend that although this heuristic allows a single AP to select a least congested channel, it does not consider channel overlap or re-use. Combining LCCS with a graph coloring scheme resulted in decreased or eliminated interference as well as the use of channels beyond the 3 orthogonal ones.

Since channel assignment is an optimization problem, taking a mathematical approach is common. The authors in [3] and [4] use Integer Programming (IP) to obtain a channel reuse pattern based on co-channel interference constraints. Integer programming is mathematical program model in which some problem is optimized by minimizing or maximizing a cost function. In [3] the authors propose an IP model that takes both channel overlap and traffic demand into account. In addition, the authors acknowledge the computational complexity of IP and apply some properties of the solver to develop a heuristic with performance comparable to the IP solver. Both techniques were able to successfully decrease overall interference, but the strategies are for a centralized channel assignment system. [4] also formulates an IP problem and proposes two heuristic algorithms based on minimum spanning trees. The first heuristic only assigns the 3 orthogonal channels, while the second attempts to use the full spectrum of 11 channels.

The algorithms use an extended form of Prim's algorithm to assign frequencies with the objective of minimizing interference and maximizing throughput and produced an assignment close to optimal for networks of varying density. As an alternative to centralize assignment methods, [14] considers channel assignment as a constraint satisfaction problem (CSP) and proposes a distributed DCA model. Uses a channel state table and applying channel overlap constraints, the authors were able to solve the CSP extremely fast with low computation time. Other suggested mathematical approaches use genetic algorithms, neural networks and simulated annealing [18, 23].

In addition to channel overlap, many other factors that affect network performance can be included in the assignment problem. [7] includes channel acquisition time as a factor in distributed DCA decision-making and proposes a model that takes search and channel acquisition times into account. [22] addresses routing and packet delivery issues caused by interference in wireless mesh networks and presents a routing-information-aware DCA algorithm. The proposed algorithm handles load-balancing, adaptive traffic assignment and fault recovery, but requires nodes to have an additional channel resource for exchanging routing information.

Currently there are many proposed dynamic channel assignment techniques that allow the system to maintain minimal interference even if the topology changes dramatically, but algorithms that work well in simulation are often not applicable in reality. Since nodes vary widely in hardware settings and may be deployed remotely, frequent channel switching will quickly become costly and unfeasible if dependent on

network engineers to carry out switching. For this reason some approaches use variations such as SCA with borrowing or a hybrid strategy (HCA) [11, 21] combining SCA and DCA, but a better solution is to improve the ease of channel switching. Such a solution is soon becoming possible with the introduction of software-defined networks.

Software defined networks (SDNs) decouple wireless network control from the physical hardware, allowing for improvements in service such as seamless roaming and authentication as well as the ability to direct bandwidth heavy traffic to Wi-Fi and light traffic to the more expensive 3G. Future applications could lead to devices that are able to aggregate any Wi-Fi/3G/4G signals in an area and seamlessly switch among them without interrupting the user. Software defined networks can be easily managed due to the use of network function virtualization (NFV) which allows remote updates and provisioning of nodes from a central controller. The virtualized network is programmable and inexpensive to change. For example, instead of using a network engineer to manually assess interference and then deploy someone to change channels on each node in the network, an SDN-enabled network switch or router can run software to monitor the entire network and push channel switch commands to nodes without any human administrator intervention. This paves the way for more stable networks that can maintain high performance without human intervention even if nodes are added or removed from the network.

Network providers are pushing for SDN because the proliferation of network devices in combination with greater demand for data processing is increasing the cost of

deploying a network. Cities have begun using municipal Wi-Fi, which is large-scale city wide wireless access, meaning network providers must be able to deploy and coordinate hundreds of access points. By moving expensive network applications upstream, the cost of intermediate and edge devices and thus overall network cost can be kept low.

Allowing channel allocation in a wireless network to be handled by software provides a valuable architecture in that the entire process can be automated, resulting in a network that can resolve itself into a conflict-free assignment if one exists. A potential drawback to a self-resolving network, however, is the limited visibility of each node and resulting susceptibility to malicious behavior aimed at decreasing the stability of the network. If nodes monitor surrounding signals and switch accordingly, an attacker may select and broadcast channels near certain nodes to create the false impression of an interfering neighbor, forcing the node to switch channels unnecessarily. In this way the attacker may prevent the network from reaching a conflict-free resolution by causing continuous switching, and even create more interference by forcing nodes to switch to more congested channels. In addition to increased interference, this type of "decoy" attack can cause degraded network performance due to delays from accumulating channel acquisition times. This type of channel switch attack has been the focus of some previous studies [12].

Furthermore, since channel switching can be modeled as a discrete-time process and choosing attacks is an optimization problem, the attacker can use mathematical decision making to intelligently determine an attack policy that causes maximum damage

while requiring minimum attack actions on the part of the attacker. One type of decision model is a Markov Decision Process and this framework has been applied to a variety of decision problems [2, 5, 19]. By adapting this framework for an attacker targeting software-driven channel assignment, we can study the robustness of channel-switching techniques under malicious conditions in order to recognize vulnerabilities.

Thesis Focus

This thesis aims to examine the performance of a dynamic channel allocation scheme and expose security weaknesses using a number of different attack policies. We simulate various network topologies to represent real life configurations such as apartment complexes and individual houses in a neighborhood. We address topologies that have known conflict-free assignments as well as those where it is unknown whether such a solution exists. An attacker attempts to create channel conflicts in the topology by launching a decoy attack. In a decoy attack, the attacker selects one of two possible actions- do nothing or create a false conflict at some location in the network to force surrounding nodes to change channels. The state space of a network with channel switching is immense, so even if an attacker wants to predict all future states and decide on an optimally damaging attack, it is not feasible.

The goal of this thesis is to develop a suboptimal attack policy by approximately solving an MDP. We run multiple trajectories in simulation to approximate an attack policy that will maximize the damage done to the network while keeping the attacker's cost low. We examine the behavior of the system when it is not under attack and when it

is under attack by monitoring convergence to a conflict-free assignment if one exists. We compare the derived suboptimal attack policy to less intelligent policies such as random attacks or DoS attacks to determine if the suboptimal policy is able to force the system into worse states.

III. SYSTEM MODEL

Introduction

The network topologies used in this thesis represent a static wireless network where each node is a wireless access point (AP) equipped with one single-channel radio and each edge in the topology connects APs within each other's interference radius. For an N-node network, the state $s_k \in S$ of the system at time k is represented by the vector

$$s_k = [T\ C_k\ v\ b]$$

where T is a standard N x N adjacency matrix capturing the network topology, C_k is an N x 1 vector containing the channels assigned to each AP at time k, and v and b are two describing variables to keep track of the AP most recently attacked (if any) and steps since last attack. The edges in the adjacency matrix T show which APs are within an interference range, meaning any APs connected by an edge cannot use the same or adjacent channels without experiencing interference. The density of T can be controlled during topology generation. Channels used in the assignment vector C_k are limited to the set of all usable channels C 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 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11} and for all $c_i \in C_k$, $c_i \in C$. The system transitions from s_k to a new state s_{k+1} when an AP switches channels, thus modifying the C_k assignment vector.

Network Topologies

We tested a range of network topologies selected to reflect real world topologies. We generate topologies by creating a location matrix depicting the spatial arrangement of all APs in the network. We then convert the location matrix to an adjacency matrix using

a given interference radius value. For all topologies generated for this thesis we considered the interference radius to be 1, meaning an AP only interferes with neighboring APs located one cell away in the location matrix. A description of each topology is as follows:

The 5NAN topology is a diamond shape cluster of 5 APs representing a small section of a suburb, or “neighbor area network”, where houses adjacent to and across from each other interfere. There are maximally 16 conflicts in this topology and it has a known conflict-free solution.

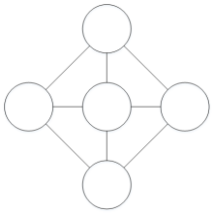


Figure 1. 5nan topology.

The 5chain topology is a chain of 5 APs where the node degree is 2 for center APs and 1 for the two APs on the ends. There are maximally 8 conflicts in this topology. This topology used to see if attacker could create a ripple effect with its attack.

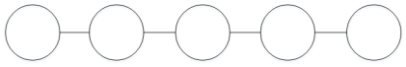


Figure 2. 5chain topology.

The 6ring topology is a ring of 6 APs where each AP has node degree 2. There are maximally 12 conflicts in this topology and it is used to see if attacker could create a cyclical effect with its attack.

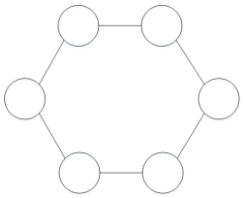


Figure 3. 6ring topology.

The 7tree topology is a binary tree structure of 7 APs, with one root AP, 2 internal APs and 4 leaf APs. There are maximally 12 conflicts in this topology and it is used to see if attacking the root could propagate to leaves.

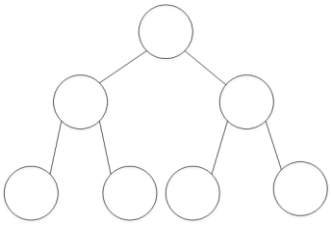


Figure 4. 7tree topology

The 8custom topology is a cluster of 8 APs where network density is skewed towards one end of the network. There are maximally 20 conflicts in this topology, and it is used to examine attack behavior when the network as a range of node degrees. While other topologies have nodes where the degree is either 1 or 2, this topology has nodes with degrees ranging from 1 to 5.

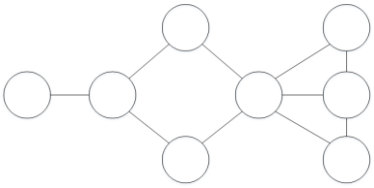


Figure 5. 8custom topology.

The 10barbell topology is a network of 10 APs arranged as two higher density clusters connected by a low density chain. There are maximally 26 conflicts in this topology and it is used to see if an attacker could attack the center and cause propagation either direction.

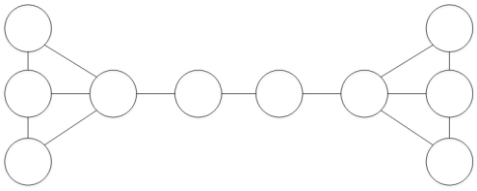


Figure 6. 10barbell topology.

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 C and is able to query and store information about its neighboring APs, which are any APs within an interference radius, to determine which channels are currently in use. The system model contains the following information:

All nodes in the network

$$A = \{AP_1, AP_2, \dots, AP_n\}$$

All neighbors of a node

$$neighbors(AP) = \{n_1,\ n_2, \dots, n_i\}\ where\ i = \delta(AP)$$

All channels used by neighbors of a node

$$neigbhor\_channels(AP) = [n_1.c,\ n_2.c,\ \dots,\ n_i.c]$$

All conflicting nodes in the network

$$A' = \{AP | conflicts(AP) > 0\}$$

In the event of a channel conflict with one or more neighbors, an AP can compute additional information from its stored information to 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 C but not in any of the interference sets of the APs neighbors. An interference set includes all channels that overlap with the assigned channel based on a channel separation constant. For an AP on channel c, the AP's interference set contains a set number of adjacent channels as follows:

$$interference\_set(AP) = \{c_{i-x}, \dots, c_{i-1}, c_i, c_{i+1}, \dots, c_{i+x}\}$$

where c_i is the AP's local channel and x is the channel separation constant. For example, for a separation value of 2 a channel c_i will overlap with channels c_{i-2}, c_{i-1}, c_{i+1} and c_{i+2}. For all test cases examined in this thesis we used a channel separation constant of 2. The set of channels an AP can freely switch to is the set difference of usable channels and the union of the interference sets of all its neighbors.

$$available\_channels(AP) = C - \bigcup interference\_set(neighbors(AP))$$

The number of channel conflicts experienced by an AP is the number of its neighbors' interference sets that contain its local channel.

$$conflicts(AP) = \left|\{n | AP.c \in interference\_set(n)\}\right|$$

If one or more conflicts is detected, the AP will compute its set of available channels. If more than one channel is available, the AP will select one at random. If no channels are available, an AP will switch to an unavailable channel that is "least conflicted", meaning it appears in the fewest number of interference sets. If more than one channel is "least

conflicted" the AP will select one of them at random. The channel switching procedure for an AP is outlined as follows:

$$AP_s = pick\ random\ from\ A'$$

$$N = neighbors(AP_s)$$

$$unavailable\_channels(AP_s) = \bigcup_{N} interference\_set(n)$$

$$available\_channels(AP_s) = C - unavailable\_channels(AP_s)$$

$$if\ available\_channels(AP_s) = \emptyset$$

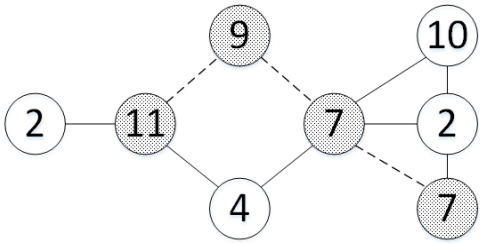
$$pick\ least\ interfering\ c \in unavailable\_channels(AP_s)$$

$$else$$

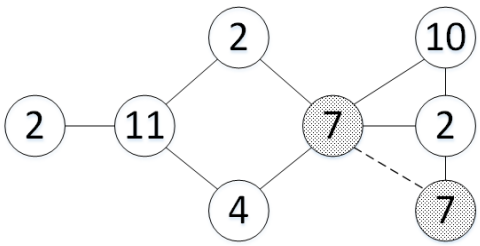
$$pick\ random\ c\ \in available\_channels(AP_s)$$

A typical progression of normal switching behavior is as follows.

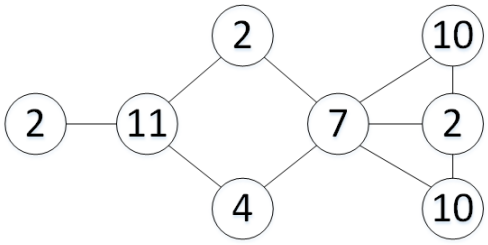
Step 1, system initially has 3 conflicts:



Step 2, system has 1 conflict after a node switches from channel 9 to 2.



Step 3, system has 0 conflicts after a node switches from channel 7 to 10.

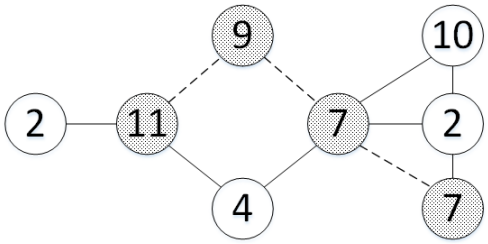


Decoy Attack

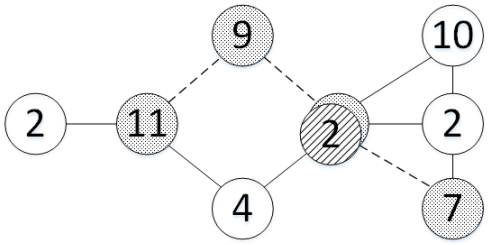
A decoy attack takes advantage of the spatially static nature of the network. An attacker mounting a decoy attack aims to degrade network performance by increasing network instability and delays with unnecessary channel switching, increasing signal interference and congestion by causing more conflicts, and preventing the system from reaching a conflict-free channel assignment if one exists. 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 high power signal that overwhelms that of the AP. Neighboring APs would then see the attacker's broadcasted channel and base their decisions off the fake channel instead of the actual channel assigned to the AP. In this way the attacker can make there appear to be a conflict where there is none and potentially cause a reaction of APs switching channels unnecessarily.

Using the topology shown previously, a typical progression of a system under attack is as follows.

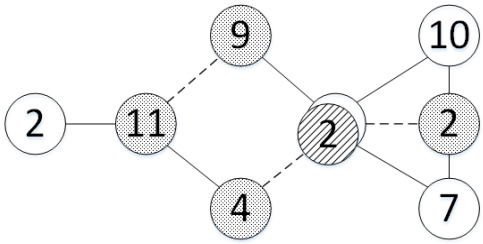
Step 1, system initially has 3 conflicts:



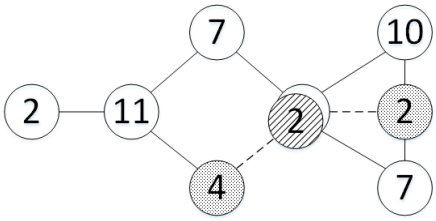
An attacker broadcasts fake channel 2 at its victim node:



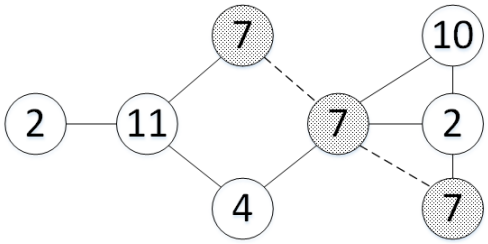
Based on the fake channel, the system now thinks the following 3 conflicts:



Step 2, the system switches a node from channel 9 to 7, thinking the new channel is no longer a conflict:



In reality, however, the channel is still causing a conflict with the victim node and the system has 2 conflicts:



Under no attack conditions as shown previously, the system would only have 1 conflict by this step.

It is not guaranteed that launching an attack will cause a channel switch, because all APs in conflict at a given step have equal probability of being selected to switch. 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 in 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 possible 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. To maximize the damage caused in comparison to the costs incurred, the attacker must programmatically find a damaging attack pattern by maximizing a cost function.

When attacking a victim, a single attacker who must travel to the desired victim AP's location in order to launch the attack. The possible victims available to the attacker

at each step depends on the attacker's location, so travel distance is the limiting factor in possible victims. We use the hop distance between APs to represent the travel distance for the attacker. The hop distance of all APs in the network is computed as follows where the final output is the hop matrix C. The attacker refers to this information when making decisions.

$A = adjacency\ matrix\ of\ the\ N\text{-}node\ network$

$B = NxN\ matrix\ of\ zeros$

$C = A$

$m = 2$

for all node pairs (i,j) from (1,1) to (n,n)

if i == j or C(i,j)>0

continue

else

$for\ k = 1\ to\ N$

$if\ C(i,k) > 0\ \&\&\ A(k,j) > 0$

$B(i,j) = m$

$break$

$if\ B == 0$

$return\ C$

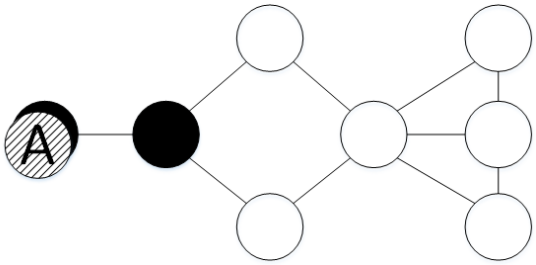
$else$

$C = C + B$

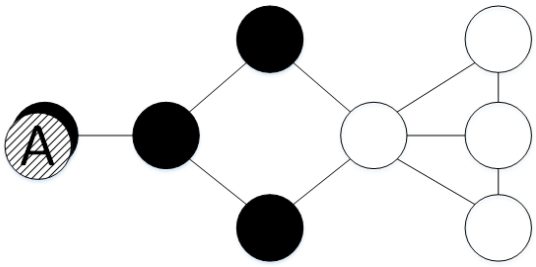
$B = 0$

$m = m + 1$

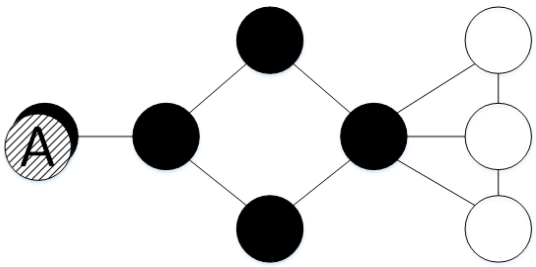
Potential victims after waiting 1 step.



Potential victims after waiting 2 steps.



Potential victims after waiting 3 steps.



For a decoy attack, an attacker has two possible actions: (1) do nothing and (2) make it appear as though an AP n has switched to channel c. These attack actions are represented as <-1,-1> and <n,c>, respectively. 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. As the system transitions from state to state, the attacker collects an immediate

reward r. The reward r for a particular state s_k is the sum of conflicts in the state:

$$r(s_k) = \sum_i conflicts(AP_i)$$

The expected reward g when transitioning from state s_k to s_{k+1} is

$$g(s_k, s_{k+1}) = P(s_{k-1} \mid s_k) * r(s_{k+1})$$

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 $P(s_{k+1} \mid s_k)$ of transitioning from s_k to s_{k+1}. Mounting an attack using action a incurs a cost that takes into account travel to the node and danger of being discovered.

$$attack\_cost(a) = \begin{cases} \infty & if\ d(AP_0, AP) > s_k.b \\ h * d(AP_0, AP) + \delta(AP) & otherwise \end{cases}$$

where action =<AP,c>, $c \in \bigcup interference\_set(neighbors(AP))$, AP is the victim AP to attack, AP_0 is the last AP attacked s_k.b steps ago, d(AP_0,AP) is distance between the APs, h is a scaling constant, and $\delta(AP)$ is degree of the AP. The distance to the AP must be less than or equal to the number of steps b that have passed since the last attack in order for the attacker to have enough time to travel to the AP. The expected immediate reward when transitioning from state s_k to s_{k+1} as a result of action a is

$$g(s_k, a, s_{k+1}) = P(s_{k+1} \mid s_k) * r(s_{k+1}) - attack\_cost(a)$$

In order to find a tradeoff between reward and attack cost, the attacker must develop an optimal attack policy μ that maps $S \rightarrow A$ and specifies the best course of action to take given a particular network state. A policy contains a sequence of actions given a state, so an optimal policy should maximize reward and minimize 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 full visibility, the attacker can solve a Markov Decision

Problem (MDP) to select an attack policy.

IV. POLICY ITERATION

Introduction

A system can be modeled as a Markov decision process if it satisfies the requirement of discrete system states which can be reached by state transitions. The system's state should be defined by a set of variable and the transitions must capture the change in variable values from the current state to another state without relying on any information from previous states. By adding values to the state transitions, a Markov decision process can be made into a Markov decision problem and applied to many types of decision-making tasks to optimize overall performance. For our system, we have a model of states and state transitions as well as a decision-making problem in which an attacker must determine a sequence of attack actions to perform. As an MDP, we aim to minimize the attack cost while maximizing the path reward. An attack policy outlines the sequence of actions the attacker should take over some time frame, but since this channel switching model has an infinite horizon we use a discount factor γ to weight the potential rewards and bias the attacker towards closer rewards. From the current state of the system, an attacker can determine current conflicts and their sizes as well as potential next states and the probabilities that a given action and state will result in a given state.

In order to obtain an optimal attack policy, the attacker would need to exactly evaluate a linear set of equations for the expected state reward J for a policy μ

$$J_\mu(s) = \sum_{s'} P(s'|s) * [g(s,a,s') + \gamma J_\mu(s')]$$

over all states s' reachable from s using an optional discount factor g. Then the attacker would iteratively improve the policy

$$\bar{\mu}(s) = \max_{a \in A(s)} \sum_{s'} P(s'|s,a) * [g(s,a,s') + \gamma J_{\mu}(s')]$$

and recompute $J_{\mu}(s)$ until a policy is found that maximizes the reward. The policy improvement iteration loop terminates when the improved policy$\bar{\mu}$ is the same as input policy μ for that step.

The optimal policy obtained as an exact solution would outline the optimal action the attacker should choose given any state. For a very small network, an attacker could possibly enumerate every reachable state and select actions that lead to states with the best reward, but since this thesis focuses on a state space S similar to a real-world network with N nodes and C channels is C^N, it is computationally intractable for the attacker to evaluate the expected state reward for every possible state and action combination. Instead we apply approximate policy iteration (API) to determine a suboptimal attack policy 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 $\delta(AP)$ or 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. The greedy solution was found as follows:

$V = set\ of\ all\ nodes$

$sort(V)$

$for\ each\ node\ v\ in\ V\ starting\ with\ the\ smallest\ degree$

$if\ v\ is\ already\ colored$

$continue$

$else$

$available\_channels = C$

$n = neighbors(v)$

$for\ each\ n$

$i = interference\_set(n)$

$available\_channels = available\_channels - i$

$update\ histogram\ of\ interfering\ channels$

$if\ available\_channels\ \neq \emptyset$

$assign\ to\ v\ the\ first\ available\ channel\ in\ available\_channels$

$else$

$assign\ to\ v\ the\ minimum\ value\ from\ the\ histogram$

From the representative states, a set of representative features is extracted that capture the characteristics of the state and can be weighted and used to estimate its value. In API, the attacker evaluates the approximated expected state rewards using a weight vector r

$\tilde{J}_r(s) = \sum_{j=1}^{M} r_j \varphi_j(s)$

where $\phi(s)$ is the set of features for state s and M is the number of features used. 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.

Feature Selection

In addition to representative states, API also relies on a set of representative features to capture the characteristics of each state and approximate the state value. For topologies with a diverse range of node degrees, we used the following features:

φ_1= Number of APs in conflict with one or more neighbors in the current state

φ_2= Ratio of maximum number of APs involved the same conflict to degree of the network graph

φ_3= Average number of APs involved in the same conflict

φ_4= Average number of channels unavailable to an AP

φ_5= Average conflict size of the highest degree AP(s)

φ_6= Last attacked AP

φ_7= Steps since last attack

φ_8= Flag for whether attacker is at MCN

φ_9= Degree of last attacked location

φ_{10}= Conflicts at last attacked location

φ_{11}= Available channels of last attacked location

φ_{12}= Degree of largest neighbor

φ_{13}= Fraction of APs within hop distance

We adjust the features applied depending on the network topology. For a ring topology, for example, we omitted features 8, 9, 12 since all APs in a ring have the same degree. To evaluate how well the selected features captured state characteristics, we compared the estimated path reward as computed from the features to the actual path reward seen by an attacker over 50 steps and found them to be similar as shown in Figure 7.

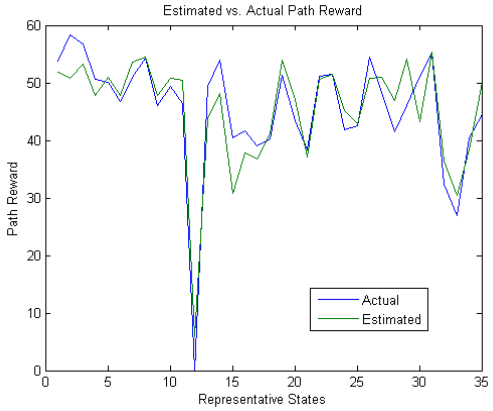


Figure 7. Estimated versus actual path reward.

Model Instantiation

During policy iteration we tested six network topologies using a range of attack costs from a low cost of 0.001 to a high cost of 4, gamma values 0.95 and 0.97, channel separation constant 2 and 35-100 representative states. Representative states were generated by assigning channels to a topology such that the final set of states included some with maximal conflicts, some with minimal as found by a greedy coloring solution, some with conflicts ranging from 2 to $\delta(g)$ and some with random assignments. Each topology was tested for 20 policy iterations and each iteration simulated 20 trajectories of 50 steps. A random attack was used as the 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. We selected 20 cores so that all 20 trajectories could be run simultaneously during each policy iteration. In all test cases we saw fast convergence to a suboptimal policy as evidence by convergence of the path reward values found during policy iteration.

Attack Comparison

To evaluate the performance of an API policy, we compare it to other attack policies by generating another set of representative states and computing the average reward gained by applying each type of attack. We compare the following types of attacks:

1. No attack
2. MCN (most complex node)
3. Random
4. DoS (Denial of service)
5. Myopic
6. API

In a no attack policy, the attacker does nothing at every step. This is the lowest cost policy and serves as a baseline by which all other attacks are compared. Since a network may start with some number of conflicts and resolve over time, the attacker may see a nonzero path reward without expending any effort. By comparing other attack policies to the no attack policy, we hope to see an elevated path reward representative of additional or prolonged conflicts in the system compared to the conflicts that would exist normally without the attacker's intervention. Under a no attack policy we are interested to see

whether the system resolves itself to a state with minimal conflicts, and if so how this state compares to the greedy channel assignment solution.

In an MCN (most complex node) attack, the attacker takes a greedy approach to selecting a victim AP that is within the interference radius of the most other APs. By selecting the most complex (highest degree) AP, the attacker can potentially disrupt the maximum amount of APs with only one attack action. If there is more than one AP with the highest degree, the attacker will pick the one with the shortest hop distance from its current location. Once at the victim AP, the attacker will constantly attack at every step with a guaranteed conflicting channel.

In a random attack, the attacker selects any random AP within one hop and chooses at random whether to broadcast a channel or do nothing. If attacking, the attacker selects any random AP, but broadcasts a guaranteed conflicting channel. 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 much sooner.

In a DoS attack, the attacker selects any random AP and chooses a random channel to broadcast at every step. In a DoS policy, the attacker never chooses to do nothing and instead attacks at every step without considering attack cost. This type of attack incurs the most cost to the attacker in hopes of also causing high amounts of damage with the constant attacks. However, since the victim APs are picked at random the attacker may be expending energy needlessly when there are better victims to attack.

A myopic attack selects an AP to attack without considering any features of the state, thus only considering immediate reward and cost and not the future value of each state. The API attack uses the features of the state and the feature weights computed during policy iteration to intelligently select an attack that should lead to more rewarding states.

For each topology we tested a range of scaling constant values to observe the attacker's behavior when attack cost was low or high. As expected, the path reward for no attack remained constant across all attack costs and we used this path reward as a baseline to compare the performance of other attack policies. When the attack cost is very low, attacker behavior corresponds to a Denial of Service (DoS) attack where constant broadcasting of fake channels forces constant switching in the network. When the attack cost is very high, this corresponds to no attack where the attacker does nothing at every step. A highly connected AP has a high node degree, making it costly to attack due to an increased chance of being exposed.

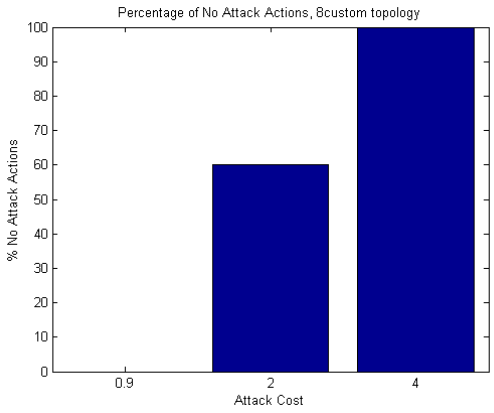


Figure 8. Percentage of "no attack" actions for various attack costs.

As the attack cost increased, the path reward of DoS and rand 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 API remained higher than the path reward of DoS. The reason API 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 API maximizes damage by constantly attacking intelligently, taking future states into account in addition to the potential immediate reward.

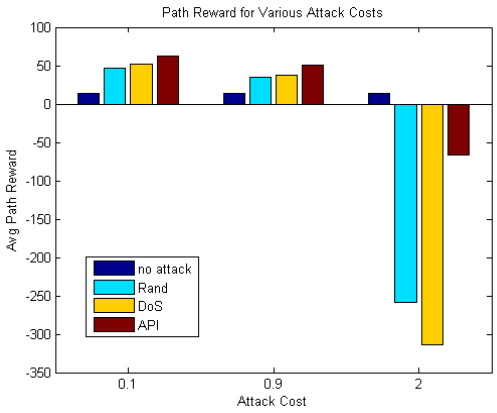


Figure 9. Average path reward for various attack costs.

Over all topologies tested, an attacker following an API policy was able to achieve a path reward greater than any other attack policy. Each of the following comparisons is based on a test case of a single attacker using each attack policy on the same set of representative topologies. The API policy had some defining characteristics that held across all topologies. When selected a channel to project, an attacker following an API policy always picked a channel that overlapped with as many reachable APs as possible. In many cases this meant positioning itself at one AP, but projecting a channel

that did not overlap with the victim AP.

For a 5 chain topology, a single attacker following an API policy achieved an average path reward 1.3 times greater than a DoS attack. Breaking down the cost and reward aspects of the average path reward, API launched much fewer attacks than DoS and caused slightly fewer channel switches, but was able to force the system into more conflicts than DoS. API focused attacks nearly entirely on the 3 internal nodes, while DoS selected its victims without bias. For a chain topology, the maximum conflict size that can be caused by a single attack is 2 so an interesting observation is whether conflicts can be introduced quickly enough to outpace the system resolving conflicts. API was able to introduce more conflicts within 50 steps than DoS.

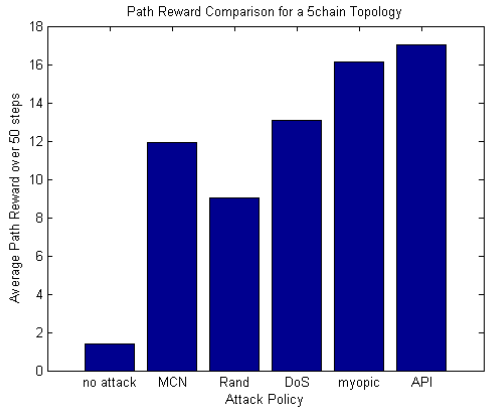


Figure 10. Average path reward for various attack policies with a 5chain topology.

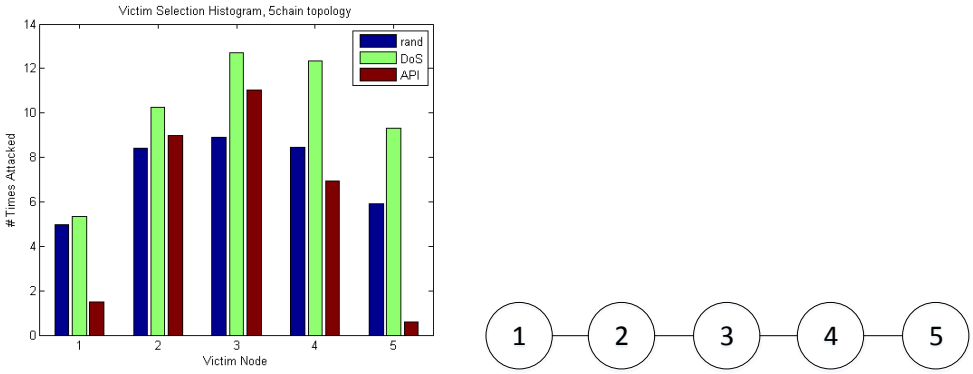


Figure 11. Histogram of victim selection for various attack policies with a 5chain topology.

For a 5 node cluster topology, a single attacker following an API policy achieved an average path reward 1.3 times greater than a DoS attack. Compared to the myopic policy, the API policy had fewer "do nothing" actions. Compared to the DoS policy, the API policy focused attacks more heavily on AP 3, the most complex AP. One possible reason the attacker did not avoid this higher cost AP is due to the clustered nature of the APs, meaning that the difference between the smallest and largest node degrees was only 1. An interesting case is one that begins as a resolved state. Myopic, Dos and API can all create conflicts in the system, but the created conflicts differ. DoS creates small conflicts in the resolved system, but conflicts do not compound and the system generally resolves again within a few steps. The no conflict states persist for several steps before the attacker successfully creates another conflict. Myopic consistently creates larger conflicts than DoS and even when the system is able to resolve the attacker can quickly create new conflicts. API creates large conflicts as well, but the system does not resolve as often as it does for myopic.

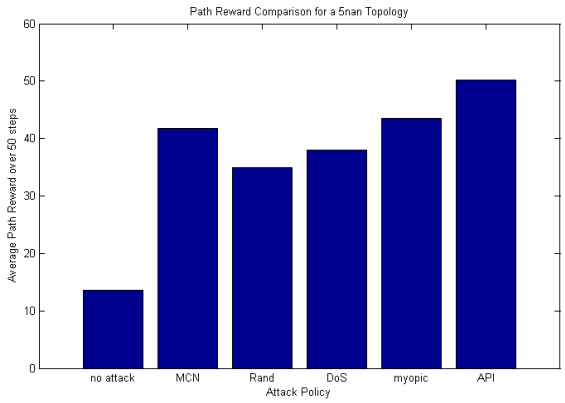


Figure 12. Average path reward for various attack policies with a 5nan topology.

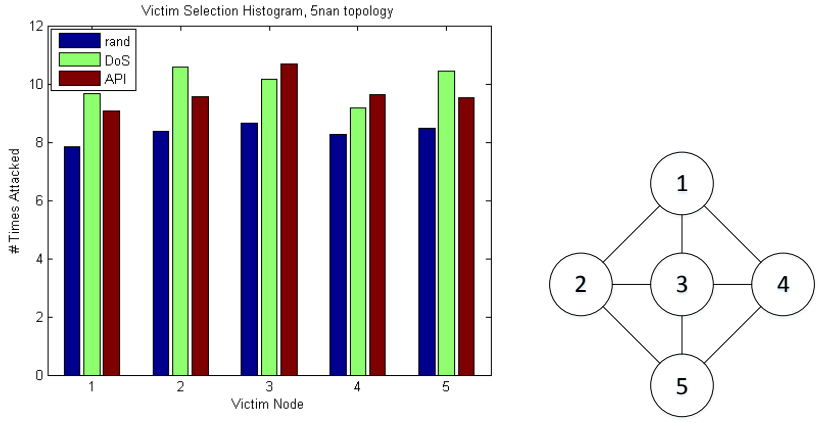
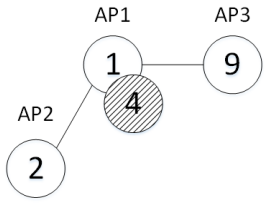


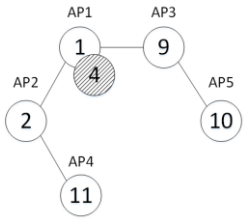
Figure 13. Histogram of victim selection for various attack policies with a 5nan topology.

For a 6 node ring topology, a single attacker following an API policy achieved an average path reward 2.7 times greater than a DoS attack. The ring configuration was the most challenging topology for every attack, with many attacks performing worse than not attacking at all. As soon as the topology converged, the attacker had great difficulty reintroducing conflicts. This is because each AP only has two neighbors, so the number of conflicts an attacker can introduce using only one attack is very limited. Since the victim AP's location will not give an API attacker any benefit, only the choice of channel

will distinguish the API policy from less intelligent policies. For this topology it is easier to see the effect of API's channel selection and how API makes sure to broadcast a channel that overlaps as many APs as possible. For example, in one test case a 3-node section of the ring had been assigned channels 2-1-9 and API chose to attack the middle AP1:



A myopic policy would see that projecting channel 1, 2 or 3 at AP1 would overlap with 2 out of 3 APs, but instead the attacker projected channel 4 at AP1. At first this choice seems less optimal since it only overlaps with one AP, but looking at the larger picture helps explain the choice:



The surrounding ring section is 11-2-1-9-10, so prior to the attack AP2 is already restricted from switching to channels 9, 10, or 11 due to its left neighbor. By projecting channel 4 on the other side of AP2, the attacker has less of an immediate chance of conflict, but it broadens AP2's unavailable set to 2, 3, 4, 5, 6, 9, 10, 11. Not only is channel 4 conflicting with AP2 to prompt it to change, but the attacker also strategically left channel 1 as a viable switch option when in truth it will cause a conflict with AP1.

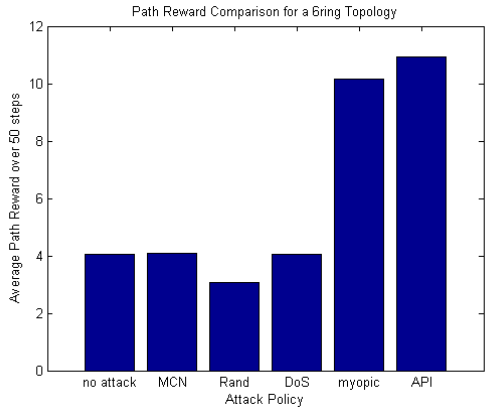


Figure 14. Average path reward for various attack policies with a 5ring topology.

For a 7 node tree topology, a single attacker following an API policy achieved an average path reward 6.3 times greater than a DoS attack. DoS performed exceptionally poorly for this topology, while MCN performed fairly well. Compared to DoS, MCN focused attacks solely on one internal AP since internal APs in the tree have the highest degree. This resulted in more frequent reintroductions of conflicts into the resolved system compared to DoS. Both MCN and DoS were unable to introduce conflicts that compounded. API was able to both introduce conflicts much more frequently than Dos or MCN but also cause compounding conflicts.

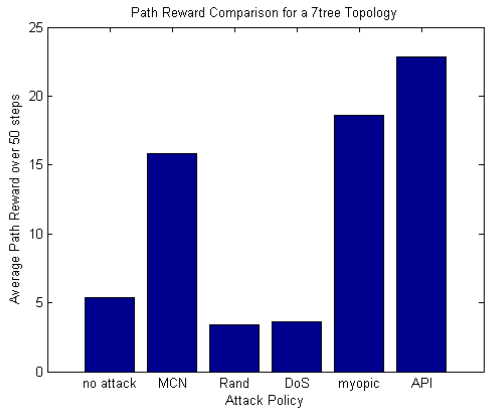


Figure 15. Average path reward for various attack policies with a 7tree topology.

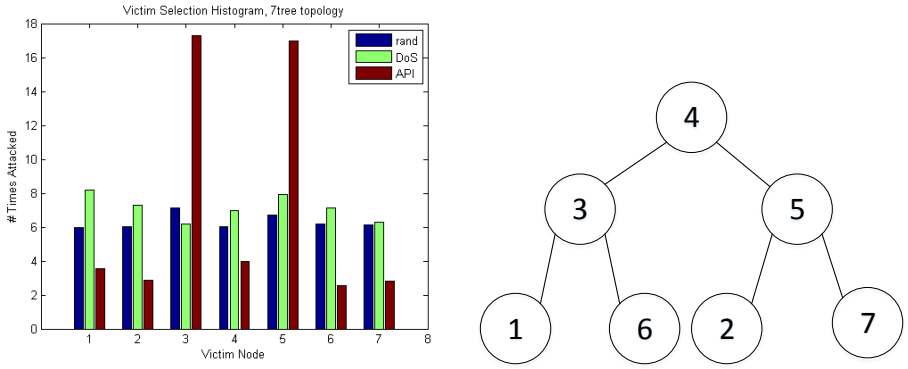
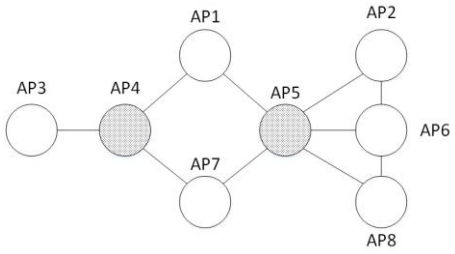
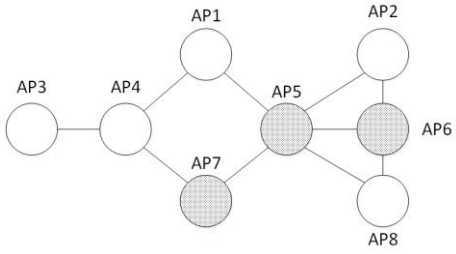


Figure 16. Victim selection histogram for various attack policies with a 7tree topology.

For an 8 node topology with a range of node degrees, a single attacker following an API policy achieved an average path reward 1.5 times greater than a DoS attack. In a multiple attacker system, the API policy achieved an average path reward 2.2 times greater than DoS. For a single attacker, nearly all myopic attacks were aimed at APs 4 and 5 located in the center of the AP cluster:



For an API attack, the attacker was more active in the denser side of the network, focusing attacks on APs 5, 6 and 7:



The API 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 other surrounding APs, increasing the potential number of conflicts that one attack could cause. The API attack outperformed myopic, showing that it is beneficial for an attacker to make decisions based on the features of a state. Although the API 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.

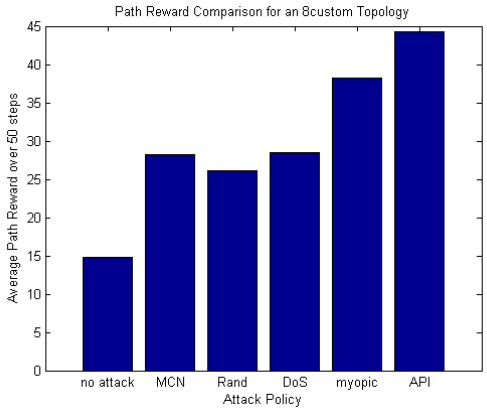


Figure 17. Average path reward for various attack policies with an 8custom topology.

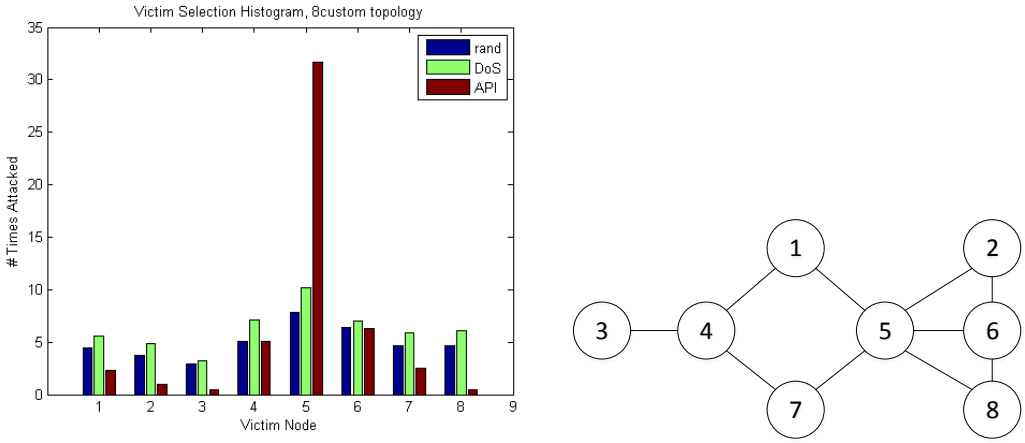


Figure 18. Histogram of victim selection for various attack policies with an 8custom topology.

For a 10 node barbell topology, a single attacker following an API policy achieved an average path reward 1.7 times greater than a DoS attack. For this topology, myopic and API policies and nearly the same performance. This indicates that selecting victims according to the selected features does not provide any advantage over selecting victims based on immediate reward.

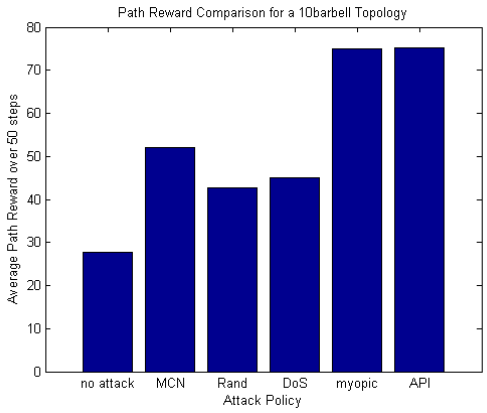


Figure 19. Average path reward for various attack policies with a 10barbell topology.

A possible explanation can be drawn from observation of the victim histogram. Most attacks are launched at nodes 4 and 7, which are the most complex nodes in the topology.

However, moving to the chain of 2-degree nodes between these complex nodes is never as beneficial to the attacker as attacking its immediate surroundings. Any potential benefit from traveling across the center chain is overwhelmed by the options in the denser areas of the topology. It is possible that this drastic difference in immediate reward compared to potential future rewards results in API behaving like a myopic policy.

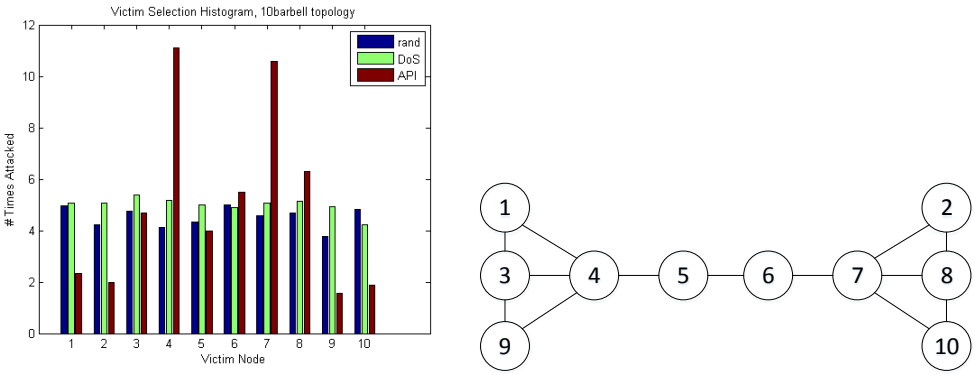


Figure 20. Histogram of victim selection for various attack policies with a 10barbell topology.

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 the both the number of channel switches and total conflicts observed in the system at high attack costs, API causes fewer conflicts and switches because the attacker is much more conservative about launching attacks.

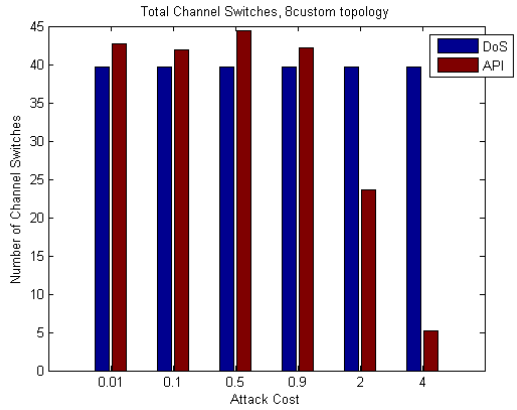


Figure 21. Total channels switched observed on average for various attack policies and costs.

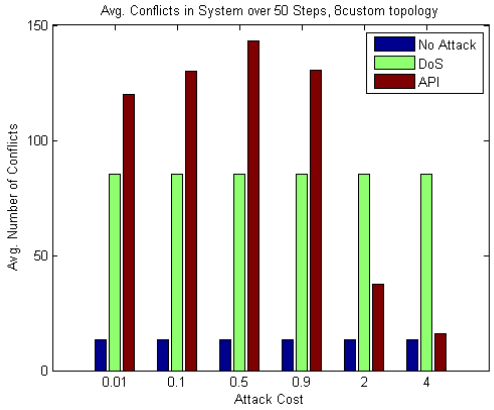


Figure 22. Total conflicts seen over 50 steps for various attack policies and costs.

Even though fewer conflicts are created, the overall path reward for API is still higher because the attacker is not constantly incurring massive penalties.

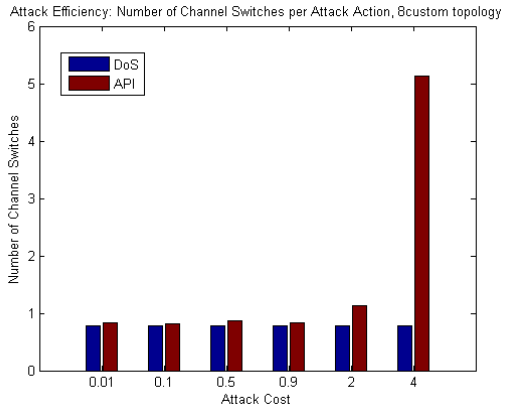


Figure 23. Attack efficiency comparison for DoS attack and API attack.

API is able to prolong the time the system remains in a conflicted state for low and medium attack costs. For high attack costs, API launches fewer attacks, so the system is able to resolve sooner similar to what would happen during no attack.

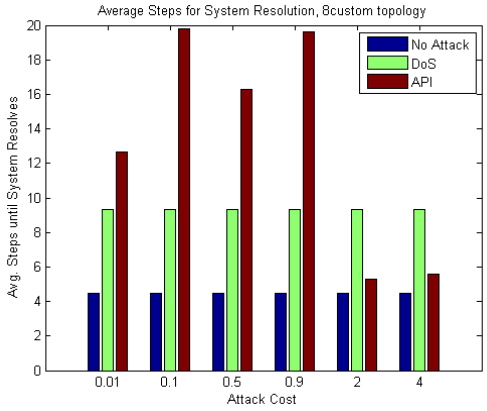


Figure 24. Average number of steps until system resolution for no attack, DoS and API attack policies.

V. CONCLUSION AND FUTURE WORK

Conclusion

Decoy attacks can significantly affect the stability of a network using dynamic channel switching by prolonging convergence to a no-conflict state, provoking unnecessary channel switching in APs, and introducing conflicts into a resolved system. An attacker can successfully estimate the value of a state using a set of predefined features and can approximately solve an MDP to obtain a suboptimal attack policy that outperforms standard attack such as DoS. The suboptimal policy performed well for a number of different topologies and in especially hostile systems with high attack costs.

Future Work

In this thesis we developed a set of 13 features to capture the characteristics of a network state. In the future we may examine additional features and the effect of feature selection on attack performance. For example, a potential feature could indicate whether a state is susceptible to a chain reaction damage effect.

This work focuses on a system with simple single-channel APs, but our proposed system model can be expanded to include multi-radio systems and mobile networks where cellular devices connect to a base station that services a particular geographical area. In this case, devices would not only interfere with other devices using the same or adjacent channels in adjacent cells but also devices in the same cell (co-site interference). Channels could be reused in geographically separated cells, however. In addition to channel switching latency, a mobile network will also have handoff delays as mobile

devices move between ranges of APs. In future work with mobile nodes, it will be important to consider the limited power supply of mobile devices and incorporate prolonged node activity as a reward for the attacker.

Our system model can also be adapted to a mesh network, where instead of assigning channels to nodes in a non-overlapping pattern the channels must be assigned to pairs of nodes. In a mesh network nodes must be on the same channel in order to communicate. This can be seen as edge coloring where the model covered in this thesis is vertex coloring.

In the future, this work can be expanded to incorporate a traffic model and develop an adaptive, load balancing channel selection algorithm and attack recovery mechanism. In addition to reducing channel overlap, such a system should aim to make efficient use of bandwidth based on traffic demands. Interference determination can be expanded to include radio sensitivity and SINR in addition to spatial separation.

REFERENCES

1. Akl, R., & Arepally, A. (2007, May). Dynamic channel assignment in IEEE 802.11 networks. In *IEEE International Conference on Portable Information Devices,* 1-5. doi:10.1109/PORTABLE.2007.63
2. Bertsekas, D. P. (2011). Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications, 9*(3), 310-335. doi:10.1007/s11768-011-1005-3
3. Daniels, K., Chandra, K., Liu, S., & Widhani, S. (2004). Dynamic channel assignment with cumulative co-channel interference. *ACM SIGMOBILE Mobile Computing and Communications Review*, *8*(4), 4-18. doi:10.1145/1052871.1052872
4. El-Hajj, W., & Alazemi, H. (2009). Optimal frequency assignment for IEEE 802.11 wireless networks. *Wireless Communications and Mobile Computing, 9*(1), 131-141. doi:10.1002/wcm.609
5. Guirguis, M., & Atia, G. (2013, June). Stuck in Traffic (SiT) Attacks: A Framework for Identifying Stealthy Attacks that Cause Traffic Congestion. In *Vehicular Technology Conference,* 1-5. doi:10.1109/VTCSpring.2013.6692769
6. Gupta, P., & Kumar, P. R. (2000). The capacity of wireless networks. *IEEE Transactions on Information Theory, 46*(2), 388-404. doi:10.1109/18.825799
7. Gupta, S. K., & Srimani, P. K. (1999, February). Distributed dynamic channel allocation in mobile networks: Combining search and update. In *IEEE International Performance, Computing and Communications Conference,* 120-126. doi:10.1109/PCCC.1999.749429

8. Haidar, M., Ghimire, R., Al-Rizzo, H., Akl, R., & Chan, Y. (2008, May). Channel Assignment in an IEEE 802.11 WLAN based on Signal-to-Interference Ratio. In *Canadian Conference on Electrical and Computer Engineering,* 1169-1174. doi:10.1109/CCECE.2008.4564722
9. Index, C. V. N. (2014). Global mobile data traffic forecast update, 2014–2018 *http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.*
10. Jain, K., Padhye, J., Padmanabhan, V. N., & Qiu, L. (2005). Impact of interference on multi-hop wireless network performance. *Wireless networks, 11*(4), 471-487. doi:10.1007/s11276-005-1769-9
11. Katzela, I., & Naghshineh, M. (1996). Channel assignment schemes for cellular mobile telecommunication systems: A comprehensive survey. *Personal Communications, 3*(3), 10-31. doi:10.1109/98.511762
12. Konings, B., Schaub, F., Kargl, F., & Dietzel, S. (2009, October). Channel switch and quiet attack: New DoS attacks exploiting the 802.11 standard. In *IEEE 34th Conference on Local Computer Networks,* 14-21. doi:10.1109/LCN.2009.5355149
13. LAN/MAN standards Committee. (2003). Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE-SA Standards Board.*

14. Lin, S. Y., Horng, S. C., & Chan, T. Y. (2011, June). Interference avoidance distributed dynamic channel assignment for cellular network. In *International Conference on System Science and Engineering*, 588-592. doi:10.1109/ICSSE.2011.5961971

15. Lopez-Perez, D., Guvenc, I., De La Roche, G., Kountouris, M., Quek, T. Q., & Zhang, J. (2011). Enhanced intercell interference coordination challenges in heterogeneous networks. *Wireless Communications, IEEE, 18*(3), 22-30. doi:10.1109/MWC.2011.5876497

16. Mishra, A., Banerjee, S., & Arbaugh, W. (2005). Weighted coloring based channel assignment for WLANs. *ACM SIGMOBILE Mobile Computing and Communications Review*, *9*(3), 19-31. doi:10.1145/1094549.1094551

17. Murray, D., Dixon, M., & Koziniec, T. (2007, September). Scanning delays in 802.11 networks. In *The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies*, 255-260. doi:10.1109/NGMAST.2007.4343430

18. Ngo, C. Y., & Li, V. O. (1998). Fixed channel assignment in cellular radio networks using a modified genetic algorithm. *Vehicular Technology, IEEE Transactions on*, *47*(1), 163-172. doi:10.1109/25.661043

19. Nguyen, V., Guirguis, M., & Atia, G. (2014, October) A Unifying Approach for the Identification of Application-driven Stealthy Attacks on Mobile CPS. In proceedings of the 52nd Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, October 2014.

20. ONF Market Education Committee. (2012). Software-Defined Networking: The New Norm for Networks. *ONF White Paper. Palo Alto, US: Open Networking Foundation*.
21. Papazoglou, P. M., Karras, D. A., & Papademetriou, R. C. (2006). On new dynamic channel assignment schemes and their efficient evaluation through a generic simulation system for large-scale cellular telecommunications. *HERMIS, An International Journal of Computer Mathematics and its Applications*, 1108-7609.
22. Sun, W., Fu, T., Xia, F., Qin, Z., & Cong, R. (2012). A dynamic channel assignment strategy based on cross-layer design for wireless mesh networks. *International Journal of Communication Systems, 25*(9), 1122-1138. doi:10.1002/dac.2385
23. Wang, L., Lee, S. N., & Hing, W. Y. (2011, May). Solving channel assignment problems using local search methods and simulated annealing. *SPIE Defense, Security, and Sensing*, 80581K. doi:10.1117/12.884492