# Cross-Layer Game Theoretic Mechanism for Tactical Mobile Networks

William J. Rogers

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Allen B. MacKenzie, Chair
Luiz A. DaSilva
Jeffrey H. Reed

September 20, 2013
Blacksburg, Virginia

Keywords: Cognitive Radio, Game Theory, Topology Control, Link Adaptation, Cross Layer

# Cross-Layer Game Theoretic Mechanism for Tactical Mobile Networks

William J. Rogers

(ABSTRACT)

In recent years, Software Defined and Cognitive Radios (SDRs and CRs) have become popular topics of research. Game theory has proven to be a useful set of tools for analyzing wireless networks, including Cognitive Networks (CNs). This thesis provides a game theoretic cross-layer mechanism that can be used to control SDRs and CRs. We have constructed an upper-layer Topology Control (TC) game, which decides which links each node uses. A TDMA algorithm which we have adapted is then run on these links. The links and the TDMA schedule are then passed to a lower-layer game, the Link Adaptation Game (LAG), where nodes adjust their transmit power and their link parameters, which in this case are modulation scheme and channel coding rate. It is shown that both the TC game and the LAG converge to a Nash Equilibrium (NE). It is also shown that the solution for the TC game approximates the topology that results from maximizing the utility function when appropriate link costs are used. Also seen is the increase in throughput provided by the LAG when compared to the results of Greedy Rate Packing (GRP).

This material is based upon work supported by the US Army CERDEC and the National Science Foundation under Grant No. 0809036. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the Department of Defense, the US Army, or the National Science Foundation.

# Contents

**1 Introduction** **1**

1.1 Software Defined and Cognitive Radio . . . . . 1

1.2 Cross-Layer Design . . . . . 2

1.3 Game Theory . . . . . 4

1.4 Contribution . . . . . 7

1.5 Outline . . . . . 8

**2 Related Work** **10**

2.1 Topology Control . . . . . 10

2.2 Scheduling . . . . . 12

2.3 Power Control and Link Adaptation . . . . . 13

**3 Topology Control Game** **16**

3.1 Model . . . . . 17

3.2 Approach . . . . . 19

3.2.1 Algorithm . . . . . 19

3.2.2 Properties . . . . . 20
3.3 Results . . . . . 22

**4 Scheduling** **28**
4.1 DRAND . . . . . 28
4.2 Link Scheduling . . . . . 30
4.3 Results . . . . . 32

**5 Link Adaptation** **35**
5.1 Model . . . . . 35
5.2 Approach . . . . . 36
5.2.1 Link Adaptation Game . . . . . 36
5.2.2 Algorithm . . . . . 38
5.2.3 Properties . . . . . 40
5.3 Results . . . . . 41
5.3.1 LAG vs. GRP . . . . . 41

**6 Conclusion** **47**
6.1 Future Work . . . . . 48

**Bibliography** **50**

# List of Figures


3.1 Original (top), TC (middle), and GA (bottom) Topologies . . . . . 23

3.2 From top to bottom, the resulting topologies from TC game with $M = 5, 25, 50, 75, 100$ . . . . . 26

3.3 Average number of rounds required to converge, for different values of $\delta$. . . 27

4.1 A successful iteration of the DRAND algorithm. . . . . 29

4.2 A failed iteration of the DRAND algorithm. . . . . 30

4.3 A failed round of DRAND prevented with the queue system. . . . . 32

4.4 Histogram of the resulting number of time slots from the 1000 random topologies. 33

4.5 Histogram of the number of links assigned to each time slot. . . . . 33

5.1 Histograms of the ALP chosen by LAG (left) and GRP (right) for all links in the 1000 iterations. . . . . 43

5.2 Histograms of the ALP chosen by LAG (left) and GRP (right) for links that were alone in a time slot in the 1000 iterations. . . . . 44

5.3 Histograms of the ALP chosen by LAG (left) and GRP (right) for links in a time slot that had at least one other link in the 1000 iterations. . . . . 44

5.4 CDFs of the system throughput for both LAG and GRP. . . . . 45

5.5 CDFs of the throughput for links that were alone in a time slot (left) and for links that had at least one other link in the time slot (right). . . . . 45

# List of Tables


1.1 Lower-Layer Variables . . . . . 4

1.2 Upper-Layer Variables . . . . . 4

1.3 Prisoner's Dilemma . . . . . 6

3.1 Results from TC and GA simulations with $c_{\min} = 0.5$ and $M = 1$. . . . . 24

3.2 Results from TC and GA simulations for $c_{\min} = 0.5$ and $M = PG$. . . . . 24

3.3 Results from TC and GA simulations for $M = 1$ and constant link costs $c_l = 1$. 25

3.4 Results from TC and GA simulations for $M = 1$ and an increased $c_{min} = 1.5$. 25

3.5 Average CPU time required for the TC and GA methods to converge. . . . . 27

4.1 Average values for certain statistics of the scheduling algorithm. . . . . 34

5.1 ALP modulation and coding combinations, with accompanying sigmoid parameters . . . . . 39

5.2 Average number of rounds to convergence, by threshold value and number of links per timeslot. . . . . 46

# Chapter 1

# Introduction

## 1.1 Software Defined and Cognitive Radio

For the past decade, Software Defined Radio (SDR) and Cognitive Radio (CR) have been popular research topics. The term SDR was coined by Dr. Joseph Mitola III, in [1], and represents a hypothetical radio where all RF and baseband signal processing is digital. Obviously, the analog-to-digital converter necessary for such a device cannot currently be made. Instead, SDR has come to mean something slightly different. According to the Wireless Innovation Forum, SDR is defined as "a radio in which some or all of the physical layer functions are software defined," [2]. SDR generally covers the hardware and software technologies where at least some of the radio's functions are controlled through software.

Another term that Dr. Mitola coined was CR, in [3]. Defining CR is difficult, as there are numerous definitions from many different groups, such as the FCC, NTIA, ITU, the Wireless Innovation Forum, and the IEEE. In general, all of these definitions have a few things in common. A radio (generally an SDR) must be able to autonomously sense changes that occur in its environment and adapt its parameters in response to these changes in order to improve its performance. Some definitions require a third capability, one of learning or intelligence,

where the radio understands how changing its parameters impacts its performance and can recall past environments and actions.

## 1.2 Cross-Layer Design

Layered architectures, such as the Open Systems Interconnection (OSI) model, were designed to be used with wired networks. This type of architecture does not allow direct communication between layers that are not next to one another. Layered protocol design offers the ability for protocols to be designed and replaced independently. A protocol change in one layer has little impact on the other layers. In wireless networks however, performance can generally be improved when information can be exchanged between layers which do not normally communicate. Because of this, layered architectures do not always perform well in wireless networks. This has resulted in a great deal of literature relating to cross-layer control mechanisms, each with different approaches and control variables, and often different definitions of what exactly cross-layer design entails.

In the survey [4], the authors provide a definition that unifies the many different interpretations, and attempted to group the different types of protocols. They define cross-layer design with respect to a particular architecture as any protocol that violates the layered architecture [4]. The authors found that these violations of the layered architecture could be grouped into four different types:

**Creation of new interfaces:** This type of violation involves the creation of new interfaces in which information is shared between layers. There are three types of new interfaces: upward (from a lower to higher layer), downward (from a higher to lower-layer), and back and forth (in both directions between two layers).

**Redefining Layer Boundaries:** This violation joins two or more adjacent layers into a "superlayer." The new layer provides the combined services of each individual layer

that it encompasses.

**Coupling without new interfaces:** In this type of violation, a new layer is designed with another layer's processing taken into consideration.

**Vertical calibration:** The last type of violation is the vertical calibration of parameters across multiple layers. In this case, each layer shows its current status and allows changes in parameters to be made so all of the layers can be jointly tuned.

Cross-layer design is particularly useful in ad hoc and cognitive networks for a number of reasons. Often in these networks, nodes make decisions at one layer with the goal of optimizing performance in one or more other layers. Also, as discussed in [5], a great deal of the PHY layer technology used in these types of networks require or benefit from cross-layer design. These include multirate transmission technology, such as the RF front end for SDRs, where different modulation, coding, and power schemes are available; advanced antenna technology, such as directional and smart antennas; and multichannel and multiradio technology. Designers of protocols for these networks are also motivated to use cross-layer design because it allows the ability for opportunistic communication, such as transmission parameters being dynamically adjusted according to variations in channel quality, as well as taking advantage of the different modes of communication in wireless links, such as using the broadcast nature of a channel to have nodes cooperate with each other [4].

After gaining a general understanding of cross-layer design and the motivation behind using it, we required a group of possible control variables on which to base our research. We divided these variables into two categories, lower-layer and upper-layer control variables, which are listed in Tables 1.1 and 1.2 below.

We chose to incorporate a number of control variables, including topology control, TDMA scheduling, modulation scheme, channel coding, and transmit power. These variables are bolded in the tables. We chose to incorporate these variables because of their close relation to each other in ad hoc and cognitive networks. The links in the network can be optimized by

| |
|---|
| **TDMA Schedule** |
| **Modulation** |
| **Channel Coding** |
| **Transmit Power** |
| Symbol Rate |
| Retransmit Behavior |
| Frame Size |

Table 1.1: Lower-Layer Variables

| |
|---|
| **Topology Control** |
| Per-Packet Next-Hop Selection |
| Routing Table Modification |
| TCP Window Size |
| Retransmission Strategy |
| IP Fragmentation |

Table 1.2: Upper-Layer Variables

adjusting their transmit power, modulation schemes, and coding schemes. However, because the links in a multihop network are all related to one another, the link adaptation process is closely related to topology control.

## 1.3 Game Theory

In this work, game theory is used to both optimize the topology of a mobile ad hoc network and to optimize the individual links through a TDMA schedule and by varying the modulation, channel coding, and transmit power. Game theory is an important set of tools used to analyze the behavior of autonomous (and often selfish) users who interact with one another. This makes it an appropriate set of tools for analyzing mobile ad hoc and cognitive networks, because there are no forms of centralized infrastructure so all aspects of the network must be distributed. Another benefit to using game theory is that with proper formation of the action space, it can provide insight into cross-layer design approaches. Lastly, game theory offers the ability to design mechanisms that offer incentives to selfish users which help steer them toward more desirable solutions from a network-wide perspective (such as the cost function in the LAG).

A normal form game is the simplest type of game. There are three parts of a normal form game. First, there is a finite set of players $N = \{1, ..., n\}$. Second, for each player $i$, there is a set of available actions $A_i$, where $i \in N$. Lastly, there is a payoff (or utility) function for each player, $u_i$, where all possible outcomes are mapped to a real number. An outcome is

defined as an action profile $A \in \times_{i \in N} A_i$, which is the Cartesian product of all users' action sets. Each player selects an action from her set of actions $a_i \in A_i$ without any knowledge of the other players' choices, in an attempt to maximize her payoff function. An action profile, $a \in A$, represents the selections of all players taken together. Each player $i$ then receives a utility of $u_i(a)$.

A player may choose either "pure" or "mixed" strategies. A pure strategy is one taken directly from $A_i$, while a mixed strategy is an action that represents a probability distribution over $A_i$. We let $\Sigma_i = \Delta(A_i)$ represent the probability distributions over $A_i$. Just as a pure strategy is represented by $a_i \in A_i$, a mixed strategy is represented by $\sigma_i \in \Sigma_i$, and the action profile is $\sigma \in \Sigma = \times_{i \in N} \Sigma_i$. The utilities of players using mixed strategies, known as the expected utility, is the sum of all the individual utilities of the pure actions, multiplied by the probability that that action is chosen, or

$$u_i(\sigma) = \sum_{a \in A} \sigma(a) u_i(a).$$

One of the basic equilibrium concepts in game theory is known as the Nash Equilibrium (NE), and can be either a pure or mixed NE. An action profile $a$ or strategy profile $\sigma$ can also be represented as $(a_i, a_{-i})$ or $(\sigma_i, \sigma_{-i})$, where $a_i$ and $\sigma_i$ is the strategy chosen by player $i$, and $a_{-i}$ and $\sigma_{-i}$ are the actions chosen by all other players. An action profile $a \in A$ is said to be a NE if

$$u_i(a_i, a_{-i}) \geq u_i(a'_i, a_{-i}) \qquad \forall a'_i \in A_i, \qquad \forall i \in N.$$

As above, a mixed strategy profile $\sigma \in \Sigma$ is a NE if

$$u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma'_i, \sigma_{-i}) \qquad \forall \sigma'_i \in \Sigma_i, \qquad \forall i \in N.$$

In other words, a strategy profile is a NE if no player can gain by unilaterally deviating from

the given profile [6].

A basic example of a normal form game that is often used to demonstrate game theory is known as the Prisoner's Dilemma. In the Prisoner's Dilemma, two criminals A and B have been arrested for a crime. Both prisoners are being held in solitary confinement, and have no means of talking to one another. The police do not have enough evidence to convict the prisoners on the main charge, however they do have enough evidence to convict each prisoner on a lesser charge which would result in one year in prison. The police offer each prisoner the option of confessing to the main charge. Each prisoner is told that if he confesses to the crime, but his partner does not, then he will go free while his partner will be given a five year sentence. They are also told that if both end up confessing, they will both receive three year sentences. The game is represented in Table 1.3.

| | B Deny | B Confess |
|---|---|---|
| A Deny | (-1,-1) | (-5,0) |
| A Confess | (0,-5) | (-3,-3) |

Table 1.3: Prisoner's Dilemma

The players in this game are prisoners A and B, and their actions available to them are $a_i = \{\text{Deny}, \text{Confess}\}$, which are represented on the outside of the table. The inside cells of the table represent each players' utilities for the corresponding action profile, with the number on the left representing prisoner A's utility and the number on the right representing prisoner B's utility. Each player's goal is to maximize their utility, which in this case is represented as years in prison.

Both prisoners know that if they deny the charge, they will only receive a short prison sentence. However, each of them also see that there is a benefit of receiving no sentence if they switch from deny to confess, as long as their partner does not also switch to confess. It must also be observed that if one prisoner has confessed, it is in the other prisoner's best interest to also confess. This results in both prisoners confessing, and each receiving a three year sentence. This is NE for this game.

## 1.4 Contribution

In this thesis, we provide a cross-layer game theoretic control mechanism for mobile wireless networks. A main contribution of this thesis is the manner in which we have organized the scheme. Our mechanism is split into three sections, an upper-layer game which handles the topology control, followed by a distributed TDMA scheduling algorithm, finished with a lower-layer game involving the link adaptation via transmit power, modulation scheme, and channel coding rate.

For the upper-layer function, our approach is inspired by the Delta Improvement Algorithm (DIA), as proposed in [7]. In this algorithm, nodes begin with their maximum transmit power to get the best possible connected topology, and then the nodes reduce their power levels by an amount $\delta$ if this reduction improves the node's utility. Once all the nodes choose not to change their transmission power, the algorithm has converged and the NE has been reached.

Our Topology Control (TC) game is the upper-layer game that decides which links the nodes keep active. In the TC game, the nodes begin with the maximum power topology and then begin removing links that are not needed. Each link has a cost that is proportional to the length of the link. In the node's utility function, these costs are weighed versus the average path length to all of the other nodes in the topology. If the node sees that it can increase its utility by removing a link (and its associated cost), then the node removes the link. The links selected by the TC game are then fed to the TDMA scheduling stage. Our contribution in this section consists of a new game-theoretic mechanism for topology control that has the ability to be modified in order to achieve specific mission goals.

Our distributed TDMA scheduling algorithm is based on the Distributed Randomized TDMA scheduling algorithm (DRAND), which was proposed in [8], and the Fair Scheduling Algorithm in [9]. In our algorithm, nodes attempt to schedule a time slot for all of their links. After a node has won a lottery, they attempt to schedule a time slot for their highest priority

link by sending a message to all their one-hop neighbors. If none of the neighbors are busy trying to schedule a slot with another node, then they reply, and the initial node schedules a slot with no conflicting links. If however, a one-hop neighbor is busy, then the node enters the busy node's queue, and is scheduled after the busy node has finished. The algorithm is repeated until all of the nodes have scheduled a time slot for all of their links. The main contributions we have made in the scheduling section is the adaption of DRAND from a node scheduling algorithm to a link scheduling algorithm, as well as the change of lottery probabilities to benefit links with a higher priority in the topology. Following the scheduling algorithm, the TDMA schedule as well as the links from the TC game are then passed to the Link Adaptation Game (LAG), which attempts to optimize the links by using a combination of transmit power, channel coding, and modulation schemes.

The LAG is run on the set of links selected to transmit in each time slot. In this game, each link makes an initial guess at their transmit power and their Adaptable Link Parameter (ALP), which in our case is a combination of modulation scheme and coding scheme. In each round, the players update their actions by first selecting a new ALP that maximizes their utility, based on the previous round's transmit power. Next, using the new ALP, the players select the new transmit power that maximizes their utility. The game continues until all of the links keep the same transmit power and ALP in two consecutive rounds. Our contribution in the link adaptation section is the adaption of the LAG from cellular networks to an ad hoc setting.

## 1.5 Outline

The thesis is organized as follows. Chapter 2 introduces the related work on the topics of topology control, TDMA scheduling, and power control and link adaptation. Chapter 3 discusses the Topology Control game and its model, properties, and results. Chapter 4 continues with the our distributed scheduling algorithm and its results. Chapter 5 describes

the Link Adaptation Game and its model, properties, and results. Finally, the conclusions are presented in Chapter 6.

# Chapter 2

# Related Work

## 2.1 Topology Control

Much of the previous work on topology control has been based on computational geometry. Much of the previous work is based on algorithms that reduce the number of edges in the graph [10], such as the Relative Neighborhood Graph (RNG) and Gabriel Graph (GG) [11], Minimum Spanning Tree (MST) [12], and the Local Minimum Spanning Tree (LMST) [13]. A RNG is a graph that is formed by removing an edge that directly connects two nodes if there is a third node that is closer to both nodes than they are to each other. A GG is one that consists of all edges connecting two nodes that, when using that edge as a disk diameter, the disk contains no other nodes from the system. A spanning tree is a subgraph that connects all of the nodes in the system together. When the links are weighted, the MST is the spanning tree that has the smallest total combined weight. In an LMST, each node independently creates its own local minimum spanning tree, using only locally known information, and only keeps its one-hop neighbors that are on the tree as its neighbors in the final topology.

As well as geometry, a great deal of the literature has been focused around adjusting the

transmission power or transmission range of the nodes to form energy efficient topologies. In [14], a cone-based topology control (CBTC) algorithm was discussed, in which the nodes begin increasing their transmit power until there exists one neighboring node in every cone of angle $\alpha$, or until it reaches its maximum transmit power. If the maximum power is reached before a node is found in all cones, then the node begins decreasing its power until it achieves the minimum power with which it still has as many cones with a neighbor as it did at max power. They conclude that as long as $\alpha \leq \frac{2\pi}{3}$ then network connectivity is guaranteed. CBTC is improved upon in [15] and it is shown that $\alpha \leq \frac{5\pi}{6}$ is sufficient for connectivity. In [16], it is proposed that each node adjusts its transmit power to keep its number of one-hop neighbors bounded in a certain range. If the node has too few neighbors, it increases its power, and when it has too many neighbors, it decreases its power. For surveys on topology control, see [17, 18].

Not many topology control algorithms use game theory. This paper builds on the Delta Improvement Algorithm (DIA) proposed in [7]. In DIA, all of the nodes initially begin transmitting with their maximum transmission power, to get the best connected topology. Then the nodes continuously reduce their power levels if the reduction improves their utility function, until no node can reduce their power without reducing their utility. In DIA, the utility of the node is a function that represents the benefit the node receives from being a part of the topology minus its transmit power. Our paper differs from this because we assume there is a cost to the node associated with keeping a link active, proportional to the length of the link. Instead of continuously decreasing the nodes transmit powers, our game involves the nodes removing some of their links. The change in utility from removing links is the difference between the cost reduction (from removing links) and a penalty associated with the increase in the average distance to all the other nodes in the topology.

## 2.2 Scheduling

There have been few applications of game theory to TDMA scheduling. To allow for scheduling flexibility, our only requirement for the scheduling algorithm was that it be a distributed algorithm.

In [19], TDMA slot allocation is combined with network discovery. As more nodes are discovered in the topology, the TDMA schedule is adapted to allow for their entry. In [20], the authors develop a two-phase cross-layer scheme to do TDMA scheduling and power control in an ad hoc network, but the proposed scheduling technique was centralized. However, this paper does prove an important property that is discussed further in the power control section below.

The authors of [8] introduce a distributed implementation of RAND (DRAND), a randomized time slot scheduling algorithm. Nodes cannot be scheduled into the same time slot as a conflicting node, which is a node that is within two-hop communication range. Nodes attempt to schedule a time slot when they win a lottery (which separates node channel requests in time and can be used to prioritize some nodes over others) by sending out a request message. If all one-hop neighbors respond with a grant message, a slot is scheduled and the process continues. A node fails to schedule a slot when one of its one-hop neighbors is already in the process of scheduling with one of its other neighbors and sends back a reject message. This is a simple and easily implemented algorithm. We also investigated newer algorithms that had been based on DRAND, in an attempt to reduce the relatively large overhead.

The authors of [21] developed a Localized DRAND (L-DRAND) that adds features for localization to DRAND by using distance information between devices. The nodes are in L-DRAND are given scheduling priorities based on link distances.

In [9], the authors proposed the Fair Scheduling Algorithm, which introduced a queuing system to DRAND. Each node maintains a queue of the request messages that it has received

and sends wait messages to nodes that send requests while another node is being processed. This reduces the overhead caused by sending reject messages and causing nodes to reenter the lottery process. Nodes are then scheduled in the order that their request was received. This algorithm achieves a fairer schedule, with a smaller overhead.

The same authors also proposed an Improved Distributed Scheduling Algorithm (IDSA) in [22]. This method allows for the nodes to decide their own slot according to their local record, instead of sending out request messages. Each node looks at its local record, picks a slot based on previous assignments, and then sends a propose message, which updates the local records of the node's one-hop neighbors. The neighbors respond with an accept message. This approach sounded promising, however the authors do no specify what happens when a node chooses a time slot that was previously unscheduled, but was chosen by another one of its two hop neighbors before it had a chance to propose.

In our scheduling algorithm proposed in this thesis, we have adapted DRAND from [8] from a node scheduling algorithm into a link scheduling algorithm. The lottery probabilities were also changed in order to provide links that have a greater importance to the topology with a scheduling preference. We have also included the overhead reducing queue method that was proposed in [9]

## 2.3 Power Control and Link Adaptation

A great deal of research has been done using game theory for power control and waveform adaptation in wireless networks. Much of this work was surveyed in [23], where the authors discuss a great deal of the literature on game theory in wireless networks. The authors present an interactive PHY-layer adaptation game form, where each node or link $j$ selects a power level $p_j$ and a waveform $w_j$ based on its observations. The works surveyed either take the form of power control games, in which the players are able to select their $p_j$ but are restricted in their choice of waveform, or waveform adaptation games, in which the power

level is restricted, but the $w_j$ is able to be adapted. The authors of [24] also offer a survey of work done on waveform adaptation using game theory. Unfortunately, much of the literature on waveform adaptation, as seen in both [23] and [24], has put a great deal of emphasis on the problem of spreading code optimization of CDMA networks, which is not relevant to our work.

There has been much research on power control in cellular, ad hoc, and cognitive networks, both with and without game theory. Some of this literature is surveyed in [23]. Of particular importance to this thesis is [20]. While the chosen power control technique for ad hoc networks in the paper is a bit simplistic, the authors prove an important property. The authors prove that once a scheduling algorithm has been applied to a wireless ad hoc network, the power control problem becomes similar to the structure of the power control problem in a cellular network. The same is true for other waveform adaptations. This is extremely important, because it means that power control and link adaptation methods proposed for cellular networks can also be applied to ad hoc and cognitive networks, after a scheduling algorithm has been applied.

More recently however, there have been several authors who have attempted to use game theory on power control and link adaptation jointly. One of the first was [25], in which the authors developed the Link Adaptation Game (LAG) for cellular networks, where each link in the network can adapt their transmit power as well as an adaptable link parameter (ALP). In that paper, the ALP was the coding rate, but the ALP can be any combination of parameters as long as they are chosen from a finite set. The authors used a pricing function to penalize the use of excessive power and showed that the game converges to a NE.

Other authors have also come up with games inspired by LAG. In [26], the authors replaced the ALP term in LAG and instead solved for the transmission rate that maximizes the adapted utility function. They proved the existence of a NE, but did not compare their results to those of LAG or any other method. In [27], the authors modified the cost function used in LAG for a cellular radio network, one that makes the cost function dependent on

each user's processing gains. The authors show that their modified version achieves higher throughputs than LAG, however their method does not always converge.

Because of what was proven in [20], we knew that once we used a TDMA scheduling algorithm on the set of links from our TC game, we would be able to use power control and waveform adaptation methods from the cellular literature. We were able to adapt the LAG from [25] from a cellular network into a game in an ad hoc environment, to provide the power control and link adaptation portion of our control mechanism.

# Chapter 3

# Topology Control Game

For our upper-layer control, we have chosen to focus on topology control. We use game theory to model the behavior of nodes in a mobile tactical network. A general overview of game theory was provided in the introduction; we now offer a more in depth look at a specific type of game, called a potential game. A potential game is a normal form game in which there exists a function, called the potential function, which is a global function that maps all players' incentives to change their strategy unilaterally [28]. Two types of potential games are exact and ordinal. For an exact potential game, there must exist a potential function $V$ such that the in the utility of the player making a unilateral deviation and the change in potential function are exactly equal for all unilateral deviations $a_i$ to $b_i$, or

$$u_i(b_i, a_{-i}) - u_i(a_i, a_{-i}) = V(b_i, a_{-i}) - V(a_i, a_{-i}).$$

For a game to be an ordinal potential game, $V$ must change in the same direction as $u_i$ given a unilateral deviation of player $i$, $\forall i \in N$, or

$$u_i(b_i, a_{-i}) - u_i(a_i, a_{-i}) > 0 \leftrightarrow V(b_i, a_{-i}) - V(a_i, a_{-i}) > 0.$$

where $V$ is known as an ordinal potential function [28].

The are a few benefits to being able to model a network as a potential game, the most important is a convergence property. As presented in [29], we have two theorems (one for finite and one for infinite games), presented below, that establish the convergence properties of potential games where players are chosen to change their strategy either at random or in round robin.

**Theorem 1.** *Let $\Gamma$ be a finite ordinal potential game. Then both the best reply dynamic and the better reply dynamic will (almost surely) converge to a NE in a finite number of steps.*

**Theorem 2.** *Let $\Gamma$ be an ordinal potential game with a compact action space $S$ and a continuous potential function $V$. Then the best response dynamic will (almost surely) either converge to a NE or every limit point of the sequence will be a NE.*

Theorems 1 and 2 are proved in [29]. The term "almost sure" results from the zero probability event that the same player is randomly chosen again and again to update their strategy in the case where the player update order is randomly chosen.

We now present our game, modeled in the form of a potential game, and what we later discuss as a "near-potential" game.

## 3.1 Model

The Topology Control (TC) game consists of $N$ nodes, or players. Each node begins with their transmit power at maximum, to identify the maximum power topology. Each node calculates their path lengths to the other nodes and their average path length to all destinations. The nodes then examine the links they currently have active and determine the change in utility that would occur with the removal of each link. Each node's utility function weighs the increase in average path length resulting from the removal of a link versus the cost associated with keeping the link active. A node removes a link if it results in an increase in utility, and the game continues with the other nodes. A node is only able to remove a

link if it still has a path to all other nodes in the network once the link has been removed. This means a node cannot remove a link that would disconnect the network. Also, the links in this model are bidirectional, so if one node decides to remove a link, it is automatically removed from the other node. The action selected by any node $i \in N$ is defined as $a_i$, which is the group of links $i$ has active. The action space $A_i$ of any node is the combination of all possible $a_i$'s. The action space of the game is $A = \times(A_i)_{i \in N}$.

Each node's goal is to maximize its utility function, which is given below.

$$u_i(a_i, a_{-i}) = -M * f_i(a_i, a_{-i}) - \sum_{l \in a_i} c_l \tag{3.1}$$

In the above equations, $a_{-i}$ are the links that the other nodes choose to keep active, $f_i(a_i, a_{-i})$ is the average path length from node $i$ to all of the other nodes in the network, $c_l$ is the cost of link $l$ among the group of links node $i$ currently has active, and $M$ is a weighting factor for the average path length. As $M$ increases, the nodes keep more links active, because the increase in average path length outweighs the gain from removing the link cost. The costs associated with each link are related to the length of the link, but we have a minimum cost, $c_{\min}$, and any link shorter than a certain length is given a cost equal to $c_{\min}$.

It was our goal to make the TC game mission driven. The mission determines the network objective function, which is then optimized. This is achieved by having the ability to replace $f_i$ with any other function that reflects the mission-specific cost of the current network configuration, provided that removing links from the network can never cause $f_i$ to decrease.

## 3.2 Approach

### 3.2.1 Algorithm

We begin by assuming all nodes in the network are static (movement will be discussed shortly). As mentioned earlier, each node begins with its maximum transmit power, so the maximum power topology can be identified. The nodes then calculate their path lengths to the other nodes in the network, and each node examines its links to find the one that increases its utility by the most when removed. In our original version of the game, nodes had no restrictions as to when they could remove links during their turn. However, using this algorithm did not achieve desirable results, as the topologies were extremely dependent upon the order that the nodes took their turns. To alleviate this problem, we introduce a counter kept by each node. The counter begins at some value, $\text{count}_{\text{max}}$, which is equal to the maximum link cost, as this is the upper bound on the increase in utility gained by removing a link. In order for a node to remove a link, its counter must be less than or equal to the amount by which its utility is increased when the link is removed. Each round, the counter is decreased by $\delta$. With a small enough $\delta$, we can now guarantee that the links are removed in the same order, resulting in the same topology regardless of node order. The links are removed in the order that provide the maximum network-wide increase in utility.

We now deal with node movement and the ability for nodes to leave and join the network. When a new node tries to join the network, only nodes within range of the new node know that it exists. The game continues regularly for all nodes that are not within range of a new node. When a node within range of the new node has its turn, it will add a link to connect the new node to the network.

When a node moves out of range of its previous links and becomes disconnected, the nodes that were previously connected broadcast that they no longer have a connection. If the moving node is still within range of other nodes in the network, then it will be treated as an incoming node and will follow the same process described in the previous paragraph.

### 3.2.2 Properties

As mentioned earlier, we attempted to model our game as a potential game due to their important convergence properties. We identify this as a possible potential function for our game:

$$V(a) = -M * \sum_{i \in N} f_i(a_i, a_{-i}) - \sum_{i \in N} \sum_{l \in a_i} c_l. \tag{3.2}$$

It can be shown that $V(a)$ is an ordinal potential function, and the TC game is an ordinal potential game when $M$ is below a certain threshold. This leads to Theorem 3.

**Theorem 3.** *The game where the individual utilities are given by (3.1), is an OPG, with the OPF*

$$V(a) = -M * \sum_{i \in N} f_i(a_i, a_{-i}) - \sum_{i \in N} \sum_{l \in a_i} c_l$$

*when*

$$M < \frac{2c_{\min}}{n^2 - 2n}$$

*where* $c_{\min}$ *is the minimum possible link cost.*

*Proof.* In order to be an OPG, $V(a)$ must change in the same direction as a node's utility when an action is made. When a link is removed, $V(a)$ increases by at least $2c_{min}$, and decreases by $M$ times the change in sum of the average path length, or $M \sum_{i \in N} f_i(a_i, a_{-i})$. In order for $V(a)$ to increase, the positive change of $2c_{\min}$ must be greater than the amount of the decrease. By finding an upper bound for the change in average path length, one can find a maximum value of $M$ such that the increase in $2c_{\min}$ is always greater than the amount of the decrease.

To find such an upper bound, we assume a worst case: Initially, suppose that all nodes are one hop from every destination, meaning each node's average path length is equal to 1, and therefore $\sum_{i \in N} f_i(a_i, a_{-i}) = n$. Now, suppose that after a link has been removed, all nodes are now $(n-1)$ hops away from all destinations (again, the worst case). We now have

$\sum_{i \in N} f_i(a_i, a_{-i}) = n(n-1)$. The sum of average hops has increased from $n$ to $n(n-1)$, which is a change of $(n^2 - 2n)$. In order to guarantee that $V(a)$ will increase in this worst case, the value of $2c_{\min}$ must be greater than $M(n^2 - 2n)$. Therefore, $V(a)$ is an OPF when

$$M < \frac{2c_{\min}}{n^2 - 2n}$$

□

Based on the potential game results discussed earlier, proving that the game is a potential game also proves that the game will converge to a NE. It has been seen in many cases however, that the TC game is still an OPG for higher values of $M$. This is unsurprising, as our upper bound on the change in the average path length was very loose. A tighter bound would allow us to guarantee the ordinal potential game property at higher values of $M$. It has also been found that, even when the TC game is not an ordinal potential game, it can still be seen to possess the important characteristic of always converging to a NE, which is shown in Theorem 4.

**Theorem 4.** *The algorithm described in section 4.3.1 with individual utilities given by (3.1) with static node positions will always converge.*

*Proof.* The topology begins with the maximum power topology, therefore all possible links are represented in this starting topology. Each round, nodes can either choose to remove a link that increases its utility, or maintain the current set of links. Nodes also cannot remove a link that disconnects itself or another node from the topology. Therefore, because there are a finite number of links, and the nodes do not choose to add links (because of static positions), the game must terminate after a finite number of steps. □

When nodes are free to move and leave the topology, and new nodes are allowed to enter the topology, the game will still converge. The convergence however depends on the set of nodes and positions eventually remaining constant for a time, as shown in Theorem 5.

**Theorem 5.** *The version of the game allowing for nodes to enter and leave the topology, as well as node movement, will converge once the set of nodes is constant and the nodes stop moving.*

*Proof.* This is similar to the proof for Theorem 2. While nodes are entering, leaving, or moving around the topology, the game will continue to update appropriately, adding and removing links when necessary. Once the nodes stop moving, and no new nodes are joining the topology, the costs will remain constant and the game will continue as the simple version did, where nodes will either choose to remove a link, or they will keep their current set of links. Therefore, because there are a finite number of links, the game must terminate after a finite number of steps. □

## 3.3 Results

In order to assess the results of the TC game, we needed something with which to compare. For this purpose, we used a genetic algorithm (GA) approach to solve for the topology that maximizes the potential function, $V(a)$.

For the following simulations, 1000 topologies of 30 nodes, which were randomly placed inside of a 5x20 unit rectangle. Our original geography of a 10x10 unit square was replaced with the rectangle in order to provide better TDMA schedule reuse, as will be discussed in Chapter 4. In our model, each node was able to form links to any other node within a distance of 3 units. For this game, each node must have a chance to remove a link before one node has the opportunity to remove a second link. The order with which the nodes remove links is chosen at random every round. For the first round of simulations, the $c_l$ for each link was equal to the link distance, $c_{\min} = 0.5$, and $M = 1$. We compared the topologies that resulted from the GA and TC game. A sample instance of an initial topology with the resulting topologies is shown in Fig. 3.1.

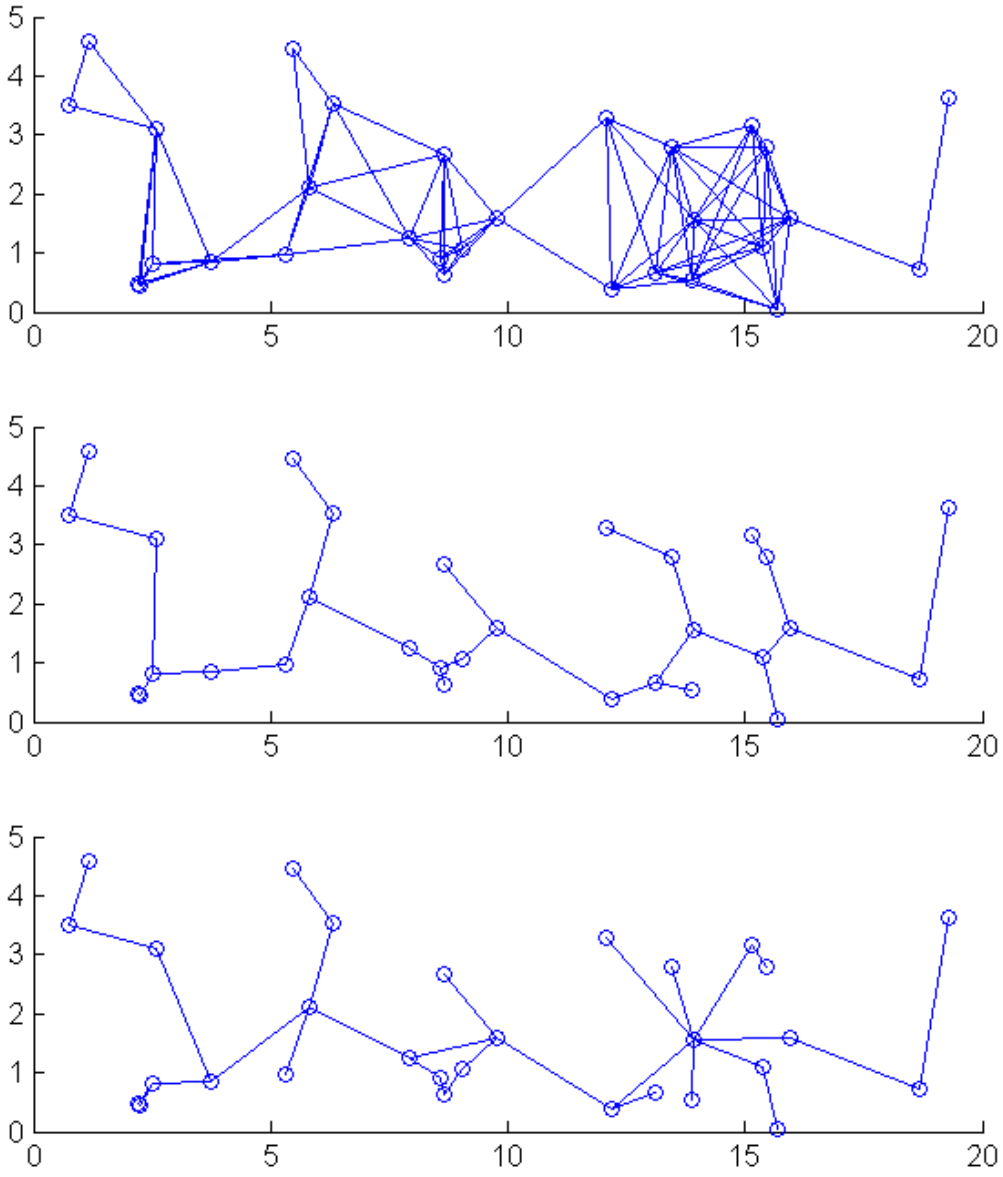


Figure 3.1: Original (top), TC (middle), and GA (bottom) Topologies

Results on the average node order (links per node) and the average path length are also shown in Table 3.1. It can be seen that the GA is able to achieve better results than the TC game, because it removes approximately the same number of links, as shown by the similarity in the average node order, but is able to achieve a lower average path length. This results from the game's preference to remove links with a long distance, as these are the links with greater costs.

| | Average Path Length | Average Node Order | Average V |
|---|---|---|---|
| TC | 6.680 | 1.933 | -285.6 |
| GA | 4.578 | 1.951 | -232.5 |
| %Δ | 45% | 1.0% | 22% |

Table 3.1: Results from TC and GA simulations with $c_{\min} = 0.5$ and $M = 1$.

When $M$ is set to the value specified earlier that guarantees the TC game is a potential game, theoretically the TC game and the GA game should provide the same results. Our findings for this case are presented in 3.2. In the simulations, the TC actually achieves a better potential function value than the GA. This is caused by the extremely small weighting of the average path length, which resulted in a huge increase in time required for the GA to converge to the optimal topology.

| | Average Path Length | Average Node Order | Average V |
|---|---|---|---|
| TC | 7.542 | 1.933 | -83.4 |
| GA | 6.951 | 1.936 | -91.6 |
| %Δ | 8.5% | 0.2% | 9.0% |

Table 3.2: Results from TC and GA simulations for $c_{\min} = 0.5$ and $M = PG$.

We also repeated the simulations with different cost values. For the results displayed in Table 3.3, all links were given a constant cost of $c_l = 1$. With constant link costs, the TC game resulted in much closer average path length values. In Table 3.4 we show the results of the simulations when the $c_l$ was set back to the link distance, but $c_{\min}$ was increased to 1.5. In this case, we achieved similar, but slightly closer results than when $c_{\min} = 0.5$.

| | Average Path Length | Average Node Order | Average V |
|---|---|---|---|
| TC | 5.003 | 1.957 | -208.8 |
| GA | 4.131 | 2.002 | -184.0 |
| %Δ | 21.1% | 2.2% | 13.5% |

Table 3.3: Results from TC and GA simulations for $M = 1$ and constant link costs $c_l = 1$.

| | Average Path Length | Average Node Order | Average V |
|---|---|---|---|
| TC | 6.355 | 1.933 | -289.5 |
| GA | 4.450 | 1.945 | -247.9 |
| %Δ | 42.8% | 0.6% | 16.8% |

Table 3.4: Results from TC and GA simulations for $M = 1$ and an increased $c_{min} = 1.5$.

As the value of $M$ is increased, the TC game begins to remove fewer links, leaving the topology less sparse. Unfortunately, as M increases, the game no longer remains a potential game. This is due to the fact that now when a node removes a link to increase its utility, the utilities of the other nodes whose path lengths are affected by this removal decreased by a combined amount that is greater than the increase that was gained by the node that removed the link. This is demonstrated in Fig 3.2. The $M$ value used is displayed on each plot, and the numbers in parentheses represent the starting value of the potential function, and the ending value of the potential function. It can be seen that, while the game is removing fewer links and the topology is becoming less sparse, the ending value of the potential function is actually less than the starting value, meaning the game is no longer a potential game.

One major advantage the TC game has over the GA approach is its convergence rate. As shown in Fig. 3.3, the number of rounds required for the TC game to converge depends on the value of $\delta$ used for the counter. With smaller values of $\delta$, only one node has the ability to remove a link per round, which guarantees that the links are removed in the order that provides the maximum network-wide increase in utility. As $\delta$ increases, the number of nodes that are allowed to remove links in a given round increases, which obviously decreases the number of rounds required to converge. This however means that the final topologies may begin to differ depending on the order the nodes have their turns. It was found through

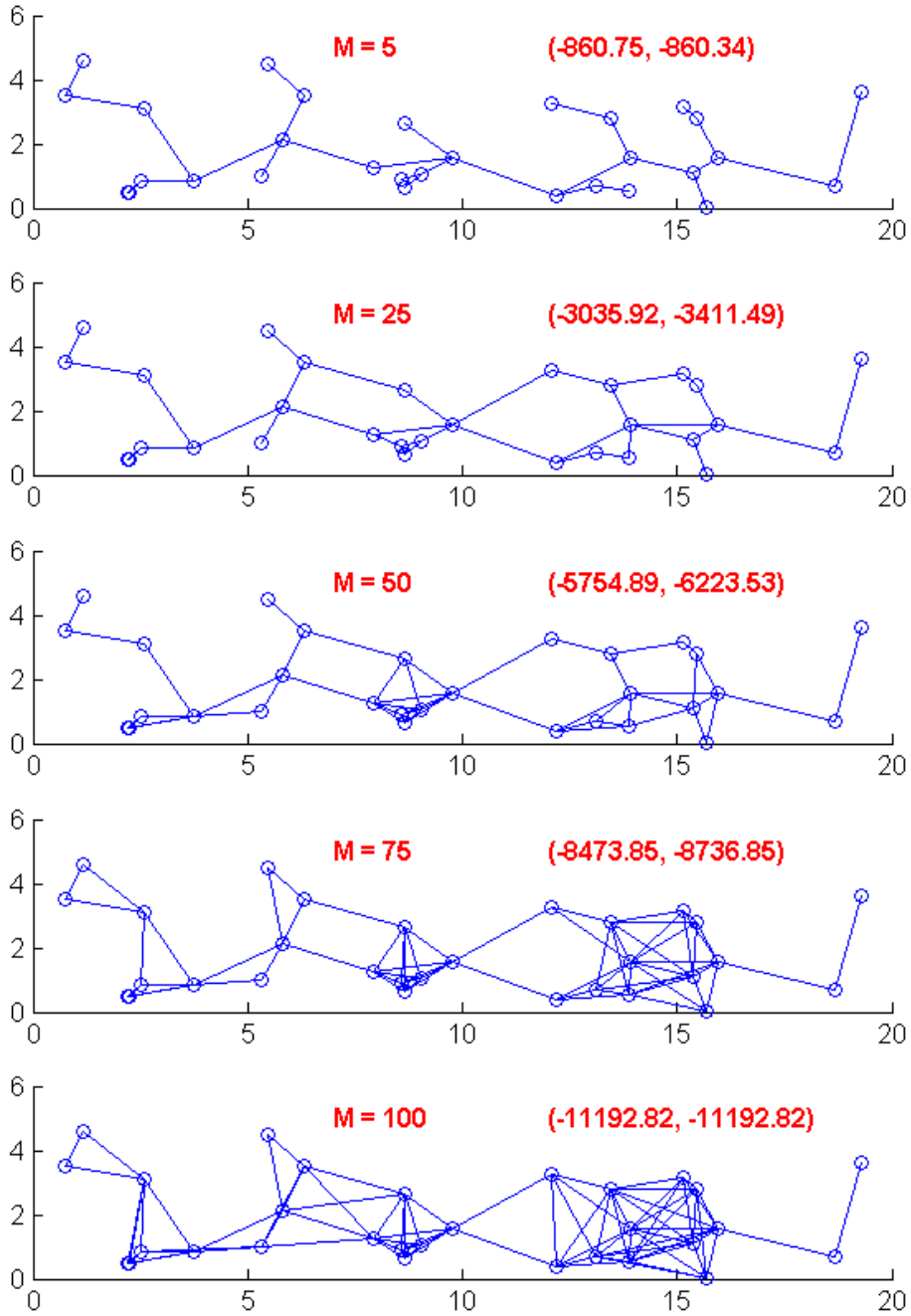


Figure 3.2: From top to bottom, the resulting topologies from TC game with $M = 5, 25, 50, 75, 100$

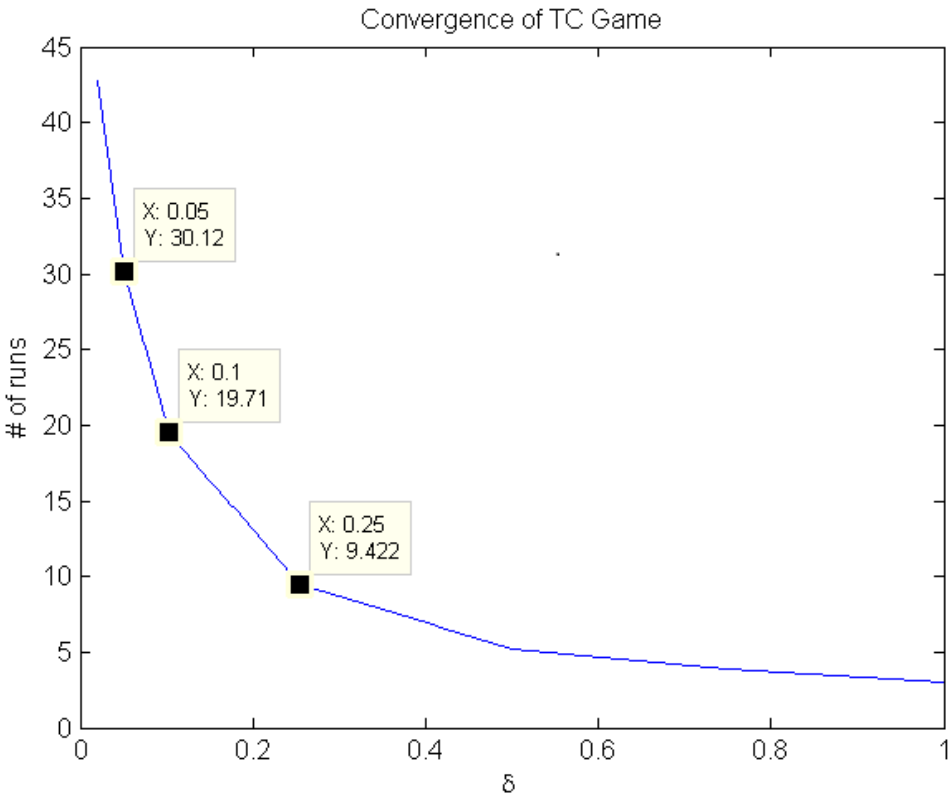


Figure 3.3: Average number of rounds required to converge, for different values of $\delta$.

simulations that for the TC game with the settings discussed at the beginning of this section, the $\delta$ could be as high as 0.25 before the final topologies began to differ. Due to the way GA's work, we can not directly compare the number of rounds required for convergence, so instead we compared the actual CPU time required for both methods to converge. The results are shown in Table 3.5. It can be seen that on average, the GA approach took over ten times as long as the TC game to converge.

| TC | GA |
|---|---|
| 1.23 s | 22.5 s |

Table 3.5: Average CPU time required for the TC and GA methods to converge.

# Chapter 4

# Scheduling

The goal of the scheduling portion of the mechanism is to create a TDMA schedule where links that cause the most interference with each other are separated into different slots. For simplicity, the only requirement of the scheduling procedure is to provide a TDMA schedule that meets that goal. We adapt a previous distributed node slot assignment algorithm DRAND into a distributed link slot assignment algorithm. Both of these algorithms are discussed below. One benefit of having the scheduling algorithm take place between the Topology Control game and the Link Adaptation game is that the proposed scheduling algorithm can be replaced with any other scheduling algorithm, if another is deemed better for the mission. The rest of the chapter is comprised of a summary of DRAND, followed by a description of our scheduling algorithm and the final results.

## 4.1 DRAND

We selected DRAND [8] as the algorithm to base our scheduling algorithm on because it is quite simple, and easy to implement in real systems. It also does not require any time synchronization to run. The slot assignment problem as defined in [8] is as follows. The

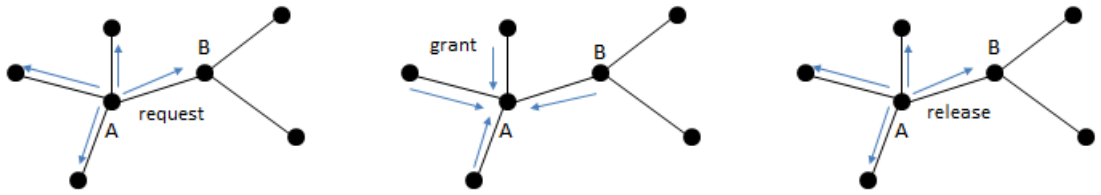


Figure 4.1: A successful iteration of the DRAND algorithm.

network is represented by a graph $G = (V, L)$, $V$ being the set of nodes and $L$ being the set of edges, or links. The edges, $l = (u, v)$, only exist if $u, v \in V$ and $u$ and $v$ can both communicate with one another. Time is divided into frames of equal length, which are then divided into time slots, also of equal length. The TDMA schedule is determined by conflict relations between nodes (in DRAND) or between links (in our scheduling algorithm described later). In DRAND, nodes are in conflict when they cause significant interference with one another, which is assumed to occur when nodes are within two hops of one another. A summary of the DRAND algorithm from [8] is discussed below.

DRAND is run in rounds, and the nodes shift between four states: IDLE, REQUEST, GRANT, and RELEASE. Let $C_j$ be node $j$'s estimate of the number of its one- and two-hop neighbors that do not yet have channel assignments. To begin the algorithm, a node A starts in the IDLE state, in which it tosses a fair coin. If the coin results in a head, the node runs a lottery. A's probability of winning the lottery is $p_A = 1/k$ where $k$ is set to the maximum $C_j$ over all one- and two-hop neighbors of A. If the node loses the lottery, it remains in the IDLE state and waits a determined amount of time before attempting the lottery again. However, if A wins the lottery, it begins negotiating a time slot with its neighbors by entering the REQUEST state and broadcasting the *request* message to its one-hop neighbors. If a neighbor is in the IDLE or RELEASE state, then it changes to the GRANT state and sends a *grant* message back to A, which includes the time slots that are already chosen by its one-hop neighbors. If a neighbor however, is in the REQUEST or GRANT state, it sends a *reject* message back to A. If A receives a *reject* message from any of its neighbors, then it sends a *fail* message out to all its one-hop neighbors and switches back to the IDLE state.

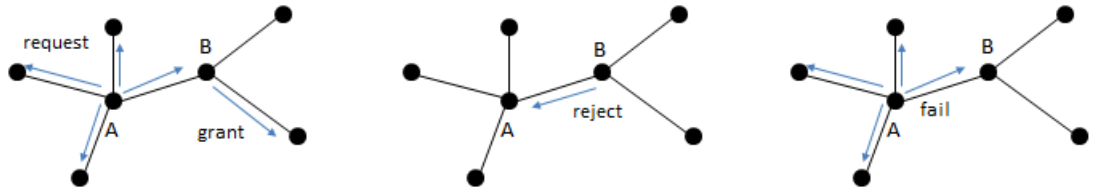


Figure 4.2: A failed iteration of the DRAND algorithm.

The neighbors whose state was changed to GRANT because of A's *request* message, upon receiving A's *fail* message, change back to state IDLE if it has yet to decide on a time slot, or to state RELEASE if it has decided on a slot. If A receives a *grant* message from all of its one-hop neighbors, then it chooses the minimum time slot that has not been chosen by any of its two-hop neighbors. A then enters the RELEASE state, and sends a *release* message to its one-hop neighbors. Successful and failed rounds of DRAND are illustrated in figures 4.1 and 4.2 respectively.

## 4.2 Link Scheduling

The DRAND algorithm provides a TDMA schedule where each node is assigned a timeslot. This does not match up well with our goal of optimizing individual links. For this reason, we have adapted the DRAND algorithm into a link slot assignment algorithm. The links discussed below differ from those discussed in Chapter 4, as they are unidirectional. Let $m$ be the number of source-destination pairs in the network ($m = n(n-1)$). Associated with each source destination pair is a path through the network, which is determined by a routing algorithm. For each link, $l$, let $x_l$ be the number of source-destination pairs for which link $l$ lies on the selected path, and let $w_l = x_l/m$ be the link's weight. We use the link weight to prioritize the links that are more important to the topology.

Much of the algorithm remains the same as DRAND, with a few important changes, as well as the addition of a concept introduced in the Fair Scheduling Algorithm [9]. First, $C_j$ is no longer the number of one- and two-hop neighbors of a node that have not selected time

slots, but it is now the number of of one- and two-hop links, or links that have a one- or two-hop neighbor as the transmitter node, that have not been assigned time slots. Second, the probability of nodes winning the lottery is changed. Each node calculates its total link weight, $w_{j,total}$. The new probability of a node A winning its lottery is $p_A = (w_{A,total})\frac{1}{L_A}$ , where $L_A$ is the number of links in $L$ that have node A as the transmitter. This new probability awards nodes that have a large $w_{j,total}$. We have also changed the conditions resulting in two links being restricted from the same time slot. Assume a slot is being assigned to link $l$. Another link, $k$, is a restricted link (cannot be scheduled in the same time slot), to link $l$ if:

1. The receiver of link $k$ is a one- or two-hop neighbor of the transmitter of link $l$. These links cause the greatest amounts of interference.

2. Link $k$ has the same transmitter as link $l$. A node cannot be a transmitter for two different links in the same slot.

3. The receiver of link $k$ is the transmitter of link $l$. A node cannot both transmit and receive in the same slot.

4. The transmitter of link $k$ is a one- or two-hop neighbor of the receiver of link $l$. This is the reverse of the first case, and is required so two restricted links both see each other as restricted.

When identifying restricted links, the graph $G$ of the original topology from before the TC game is used. This is necessary as only the original graph captures all the potential interferers. Only links selected by the TC game will be scheduled, but we must look for link conflicts reflecting all potential physical interferers.

The last change to the scheduling algorithm is the introduction of a queue for each node, which was added to reduce the overhead of the algorithm, as in [9]. In situations where one node, B, has already sent out a *grant* message to a neighbor node, C, and then receives

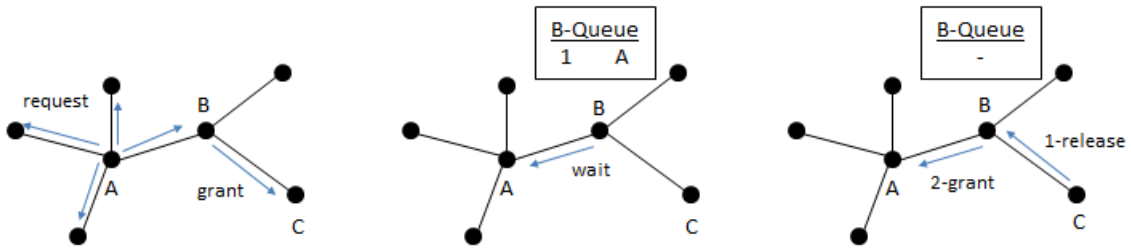


Figure 4.3: A failed round of DRAND prevented with the queue system.

another *request* message from a different neighbor, A, the DRAND algorithm requires B to send a *reject* message to A, and A to send out *fail* messages to its other neighbors. Using the queue process, instead of node B sending a *reject* message to A, it sends A a *wait* message. Node B now retains a queue of the other *request* messages that it has received. When it returns to the IDLE or RELEASE state (when it has received a *release* message from node C), it will automatically reenter the GRANT state and send a *grant* message to the first node in its queue, in this case, to node A. This process is shown in figure 4.3. The process repeats until its queue is empty. Because of the queue system, node A does not end up failing and having to attempt the lottery again, resulting in fewer messages being exchanged.

The overall process of the DRAND algorithm remains the same, with the exception of the queue system discussed above. Now however, once a node has won its lottery and received *grant* messages back from all of its neighbors, it no longer selects a time slot for itself, but it selects a time slot for its unassigned link with the greatest $w_l$. If, after assigning the link a time slot, the node still has unassigned links, it returns to the IDLE state and continues attempting to schedule time slots. Once all of the node's links have been assigned a slot, the node enters the RELEASE state.

## 4.3 Results

As mentioned in section 3.4, the simulation area is a rectangle of size 5x20 units (a rectangle allows for better simulation of spatial reuse in a small simulated area than a square of the

same total area).

The results for this section flow down from the same 1000 random topologies that were tested in the TC section above. The number of timeslots assigned in each case is presented as a histogram in Fig. 4.4. Also shown is a histogram of the number of links per time slot, as Fig. 4.5. The average number of links, timeslots per run, and number of links assigned per slot are presented in Table 4.1.

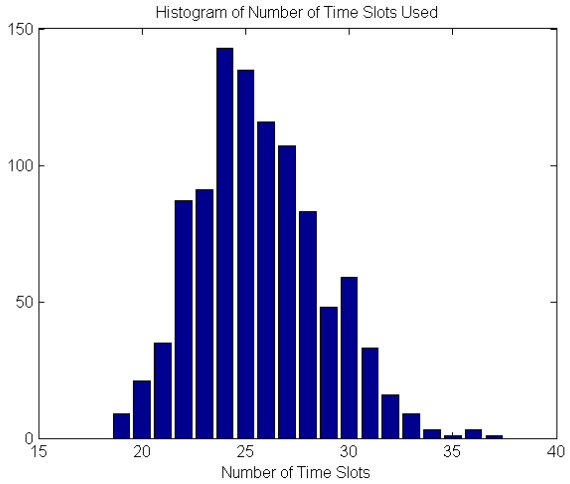


Figure 4.4: Histogram of the resulting number of time slots from the 1000 random topologies.

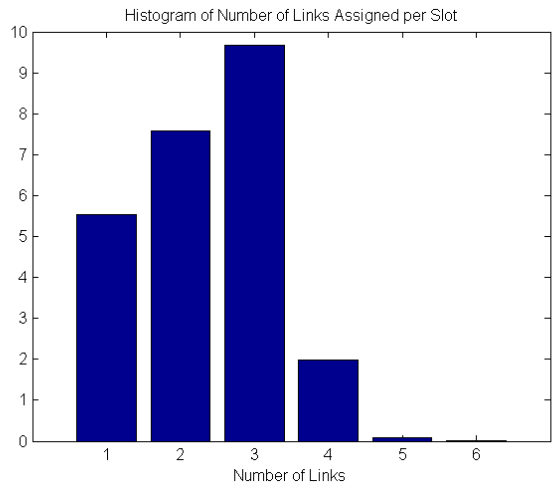


Figure 4.5: Histogram of the number of links assigned to each time slot.

| # of Links | # of Time Slots | # of Links per Slot | # of Messages | # of Rounds |
|---|---|---|---|---|
| 58 | 25.6 | 2.34 | 464 | 350 |

Table 4.1: Average values for certain statistics of the scheduling algorithm.

It can be seen that due to the conditions DRAND uses to determine when links cannot transmit in the same timeslot, many links require their own timeslot. These links central locations in the topology result in their either causing or receiving a large amount of interference to every other link in the network. However, the average number of timeslots required of 25.6, while high, is not out of the realm of possibility. These results may be able to be improved by relaxing some of the constraints when determining which links are restricted from being in the same timeslot.

Also shown in Table 4.1 are the averages for the number of overhead messages sent and the number of rounds played until all of the links had been scheduled. It can be seen that the number of rounds required to schedule all of the links, at 350, is quite high seeing as on average there were only 58 links requiring scheduling. This means that our probabilities for a node winning the lottery were probably a bit too low, seeing how approximately 85% of the rounds resulted in zero nodes scheduling a time slot for one of its links.

# Chapter 5

# Link Adaptation

Much of the work in this section has been adapted from the work of Samir Ginde, from [30] and [25]. In Ginde's work, the LAG that will be discussed in this chapter was originally designed for a cellular network. We however have adapted LAG for use with wireless ad hoc networks. As mentioned previously, we were able to do this because it was proven in [20] that once a scheduling algorithm has been applied to a wireless ad hoc network, the power control problems of the ad hoc network and a cellular network become very similar. In the rest of this chapter, we present the model and approach of LAG, followed by the special properties of the game and the final results.

## 5.1 Model

We have a set $T$ of $N$ radio links. The links in this game differ from those in Chapter 3, because in this game the links are not bidirectional. Therefore, for each link that was passed down from the TC game, we have two links, one for each direction. For simulating this game, we assume the link gain matrix remains constant. The link gain matrix is found by

computing

$$G_{ab} = k * \left(\frac{d_{ab}}{d_0}\right)^{-\alpha} \tag{5.1}$$

for all pairs of nodes $a$ and $b$ in the network. In the above equation, $G_{ab}$ is the link gain between transmitter $a$ and receiver $b$, $k$ is a normalizing constant, $d_{ab}$ is the distance between nodes $a$ and $b$, $d_0$ is a normalizing distance (in our use $d_0$ is equal to the smallest distance between two nodes in the network divided by two), and $\alpha$ is the attenuation exponent.

Each link $i \in T$ is able to adapt $r_i$ the adaptable link parameter (ALP), which must be chosen from a finite set. In the context of this game, the ALP is a choice of a combination of modulation scheme and channel coding rate. Each link can also adjust its power, $P_i$, between $P_{min}$ and $P_{max}$.

The SINR at the receiver of a link is given by

$$\gamma_l = \frac{G_{ab} * P_i}{\sum_{j \neq i} G_{cb} * P_j + n_i} \tag{5.2}$$

where $j$ consists of all the other links in the same time slot as link $i$, with transmitter node $c$ and $n_i$ is the channel noise experienced by link $i$. Depending on the SINR a link is currently experiencing, the link can use its ALP to adjust the data rate while keeping the transmit power constant.

## 5.2 Approach

### 5.2.1 Link Adaptation Game

The goal of each link is to maximize their effective link throughput $L$, given by

$$L_i = R_b \left(1 - \mathrm{FER}(\gamma_i, r_i)\right) \tag{5.3}$$

where $R_b$ is the transmission rate in bits per second on link $i$ and FER is the frame error rate as a function of SINR $\gamma_i$. Knowing that the FER decreases with an increasing SINR, it can be seen from Eq. 5.2 and Eq 5.3 that link throughput increases with a link's transmit power. Unfortunately, as one link increases its power in an attempt to maximize its throughput, it causes interference on all the other links which degrades their SINRs and in turn reduces their throughput. It is because of these properties that the situation can be modeled by a noncooperative game, the Link Adaptation Game (LAG) [30]. As mentioned earlier, a strategic-form game is made up of a group of players, their actions, and a payoff or utility function based on the actions of the players.

The LAG has a set of players, the links in set $T$ from above. The transmitters of each link adapt their transmit powers and their ALPs based on the receivers' estimates of their interference. This pair of transmit power and ALP make up the action of each player $i \in T$, and is represented as $(P_i, r_i)$, where $P_i \in P$ is the transmit power of link $i$, and $r_i \in R$ is the ALP of link $i$. The individual power and ALP selections of each players form the power and ALP vectors, $\mathbf{P}$ and $\mathbf{r}$ respectively, where $\mathbf{P} = (P_i)_{i \in T}$ where $P_i \in [P_{min}, P_{max}]$ and $\mathbf{r} = (r_i)_{i \in T}$. The action space $A_i$ of link $i$ is defined as $A_i = P \times R$, and the action space of the game is the cross product of all $A_i$'s, $A = \times(A_i)_{i \in T}$.

The last required component of the game is the utility function of the links. As discussed, maximizing the effective per-user throughput $L_i$ is the goal of each player. It therefore makes sense that this would make up a portion of the utility function. However, because this is a noncooperative game and the throughput monotonically increases with power but monotonically decreases with interference, any attempt to increase ones throughput by increasing transmit power results in performance degradation because of the interference caused by the other links also increasing their power. It is because of this that a pricing component is added to the utility function, to penalize excessive power use. The utility function of player $i$ is given by

$$U_i(\mathbf{P}, r_i) = L_i(\gamma_i, r_i) - C_i(P_i) \tag{5.4}$$

where $L_i$ is given by Eq. 5.3 and $C_i(P_i)$ is the cost function given below. Note that the cost function for player $i$ is only a function of player $i$'s power, $P_i$.

$$C_i(P_i) = KP_i^q \tag{5.5}$$

$K$ and $q$ are both positive constants. In our case, we have used values of $K = 100$ and $q = 2$.

The sigmoid function of SINR is used to model the throughput versus SINR of a wireless link and is given below. More information on the sigmoid function can be found in [30].

$$L_i(\gamma_i, r_i) = \frac{\alpha(r_i)}{1 + e^{-\lambda(r_i)[\gamma_i - \delta(r_i)]}} \tag{5.6}$$

In this equation, $\alpha$ is the peak value of the sigmoid function, $\delta$ is the abscissa of the point where the sigmoid attains maximum slope, and $\lambda$ is the steepness factor of the sigmoid function. As shown in Eq. 5.6, $\alpha$, $\lambda$, and $\delta$ can all be functions of the ALP.

In our simulations, we have used LTE physical layer parameters to demonstrate the results. In our game, the players' ALP choice is a combination of modulation scheme and coding rate, which were taken from [31]. The combinations, as well as the sigmoid parameters used are given in Table 5.1 below.
Some of the modulation and coding combinations from [31] were removed because they were never selected by any links in our model.

### 5.2.2 Algorithm

This section discusses a distributed algorithm used to find a NE in the LAG. The links do not have access to the complete link gain matrix $\mathbf{G}$ so nodes cannot discover a NE immediately. Each player $i$ must first make an initial guess $(P_i, r_i)$ at its NE power and ALP, denoted by $(P_i*, r_i*)$. Each round, the players update their guesses by selecting new $(P_i, r_i)$ that maximize their utility. Each of these updates is know as a best response, as every turn, the

| ALP | Modulation | Code Rate | $\alpha$ kbps | $\lambda$ | $\delta$ dB |
|---|---|---|---|---|---|
| 1 | 4QAM | 0.19 | 67.9 | 0.25 | -4.80 |
| 2 | 4QAM | 0.30 | 108.3 | 0.25 | -2.73 |
| 3 | 4QAM | 0.44 | 157.9 | 0.25 | -0.67 |
| 4 | 4QAM | 0.59 | 211.6 | 0.25 | 1.40 |
| 5 | 16QAM | 0.37 | 265.8 | 0.25 | 3.47 |
| 6 | 16QAM | 0.48 | 344.5 | 0.25 | 5.53 |
| 7 | 16QAM | 0.60 | 433.2 | 0.25 | 7.60 |
| 8 | 64QAM | 0.55 | 598.0 | 0.25 | 11.73 |
| 9 | 64QAM | 0.65 | 702.4 | 0.25 | 13.80 |
| 10 | 64QAM | 0.75 | 814.2 | 0.25 | 15.87 |
| 11 | 64QAM | 0.85 | 920.7 | 0.25 | 17.93 |
| 12 | 64QAM | 0.93 | 999.9 | 0.25 | 20 |

Table 5.1: ALP modulation and coding combinations, with accompanying sigmoid parameters

users update their $(P_i, r_i)$ to the "best" current options which maximize their current utility. The algorithm is finished when no player decides to change from their previous action, and the algorithm has converged to a NE solution. The algorithm is formally described below, where $k$ represents the current iteration of the algorithm.

**Algorithm LAG**

1. Initialize $(\mathbf{P}(0), \mathbf{r}(0))$. k=0.

2. At each iteration, for all $i \in T$

$$r_i(k+1) = \arg\max_{r_i \in R} \{U_i(\mathbf{P}(k), r_i)\} \tag{5.7}$$

   It should be noted that the cost function $C_i$ is not a function of the ALP, so an ALP $r_j$ is only selected if its throughput for the given link SINR $\gamma$ is higher.

3. At iteration $k$, for each $i \in T$

$$P_i(k+1) = \arg\max_{P_i \in P} U_i(P_i, P_{-i}(k), r_i(k+1)) \tag{5.8}$$

We know that $U_i$ has a single global maximum given that $r_i$ is fixed, from Proposition 1 (discussed in the Properties section below).

4. If $r_i(k+1) = r_i(k)$ and $P_i(k+1) = P_i(k)$ $\forall i \in T$, then the algorithm has converged to a NE, and $\mathbf{P}* = \mathbf{P}(k+1), \mathbf{r}* = \mathbf{r}(k+1)$.

5. Else, $k = k+1$, return to 2.

Now in our model, the scheduling algorithm has already run, and each link is assigned a time slot in which to transmit. The LAG algorithm is played separately for each time slot in the system, so as to maximize each links utility.

### 5.2.3 Properties

We now discuss some important properties of the LAG. The following definitions, lemmas, and propositions are taken from [30] and [25], and they show that a NE (formally defined below) exists for the LAG.

**Definition 1.** *Nash Equilibrium*: The NE of a strategic game $G = \langle N, A, U \rangle$ is an action profile $a'$ that satisfies the following property for all $i \in N$ [32].

$$U_i(a'_i, a'_{-i}) \geq U_i(a_i, a'_{-i}) \qquad \forall a_i \in A_i. \tag{5.9}$$

Here, $U_i$ is the utility function of player $i$.

**Lemma 1.** *The action space $A_i$ of player $i \in T$ is nonempty and compact [30].*

**Lemma 2.** *$A_i$ is not a convex set. However, under the condition that $r_i$ is fixed, $A_i$ is a convex set [30].*

**Lemma 3.** *$U_i$ is continuously differentiable on $A$ when $r$ is fixed. Hence, $U_i$ is continuous on $A$ [30].*

**Proposition 1.** *$U_i$ has a unique global maximum in $P_i$, assuming $r_i$ is fixed. $U_i$ is also strictly quasi-concave over $P_i$ [25].*

**Proposition 2.** *A NE exists in game LAG [25].*

Next, we examine the convergence of game LAG. We know that if LAG converges, then it converges to a NE, because each update by a player in algorithm LAG is a best response, maximizing their $U_i$, and the algorithm only terminates when all players do not deviate from their previously selected action.

**Theorem 6.** *Algorithm LAG converges to a NE [25].*

## 5.3 Results

### 5.3.1 LAG vs. GRP

For our results, we use the set of links that was passed down from the TC game and the TDMA schedule that resulted from our scheduling algorithm. The LAG algorithm is run concurrently on each time slot from the TDMA schedule. In our simulations, we use values of $k = 1$ and $\alpha = 4$ for the link gain calculations. Noise is referenced to the maximum received power for the link in the network with the shortest distance (maximum $G_{ab}$), as shown in Eq. 5.10, and is identical for all $i \in T$.

$$n_i = \frac{\max\limits_{i \in T}(G_{ab})P_{max}}{SNR_{ref}} \qquad \forall i \in T \tag{5.10}$$

where the $SNR_{ref}$ is the reference signal-to-noise ratio. For our simulations, we look at the interference-limited case, so the SNR is set to 100 dB. The maximum transmit power in our simulations is $P_{max} = 1$ W, and the minimum transmit power is $P_{min} = 0$ W. Also, we set the initial guess for each node to $(P_i, r_i) = (0.2 * P_{max}, 1)$. If significant movement takes place within the network, then the TC and scheduling algorithm are run again.

In this section, we compare the results of LAG to Greedy Rate Packing (GRP), which we now describe. In GRP, users are assigned SINR targets (or rates) in the order of the link gains, from highest to lowest. In the original version of GRP presented in [33], and the one adapted and used in [30], the maximum achievable target assuming worst-case interference from higher priority links is calculated and assigned in order. The problem with this algorithm for our purposes, is that the algorithm assigns large SINR targets (and hence rates) to the users with the highest link gain, and then assigns the remaining capacity to links with lower gains, which often results in links being assigned rates of zero because there is no remaining capacity available. Also presented in [33], is another version of GRP that is used when minimum rates are required on a certain number of links. This is appropriate for our studies, because once the TC and scheduling algorithm has been run, we require all links to transmit at some rate. Once the links have been assigned TDMA slots, all of the links in the slot are required to at least achieve the lowest data rate available, or ALP = 1. The SINR target assigned to each user is given by

$$\gamma_i = \{\max \gamma_i : \sum_{j=1}^{i} \frac{\gamma_j}{1+\gamma_j} + (|T|-i)\frac{\gamma_1}{1+\gamma_1} \leq 1 - \frac{n_i}{\min_{1\leq j\leq i}\left[(P_{max}G_j)\left(\frac{\gamma_j+1}{\gamma_j}\right)\right]}, \gamma_i \in \{\gamma^1, ..., \gamma^{|R|}\}\} \tag{5.11}$$

In the above equation, $G_j$ is the link gain for link $j$ and, $\{\gamma^1, ..., \gamma^{|R|}\}$ are equal to the SINR thresholds required for a link to choose that ALP, and $\gamma_j$ is the SINR target assigned to link $j$. $|T|$ is equal to the number of links in the time slot. This version of GRP is discussed in greater detail in [33].

As before, the results from this section flow down from the TC and the scheduling algorithm, and are run on the previously described scenarios. We first look at the ALP choices (and therefore the approximate rate choices) made by each link, running the LAG and the GRP algorithms. Histograms for each are presented in figure 5.1. It can clearly be seen that the LAG does an excellent job of maximizing the number of links with the highest ALP choice, while GRP fails to achieve nearly as many links with that choice. The ALP choice can

be further analyzed by viewing them in terms of the number of links in the time slot. We present two more sets of histograms, shown in figure 5.2 and figure 5.3, which show the ALP choice of the links that are alone in the time slot and the ALP choice of the links that are in time slots with more than one link, respectively. As seen in figure 5.2, LAG and GRP achieve the exact same results when it comes to ALP choice of links in their own time slot. The real improvement can be seen in figure 5.3, where the ALP choice of links that were assigned time slots with other links is shown. LAG is able to achieve the maximum ALP choice for the majority of the links, where GRP is only able to achieve lower than maximum ALP choices, with the majority of the links being assigned near minimum ALPs.

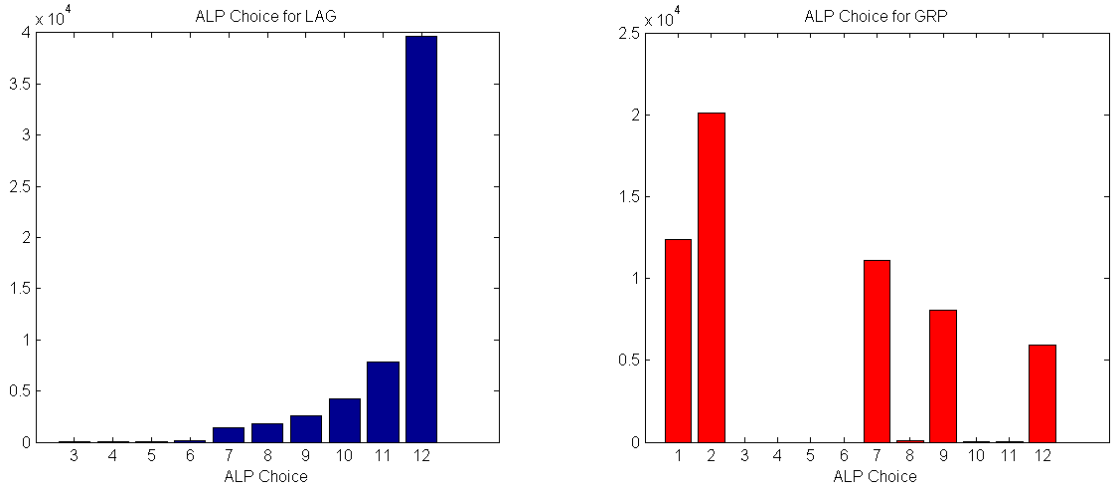


Figure 5.1: Histograms of the ALP chosen by LAG (left) and GRP (right) for all links in the 1000 iterations.

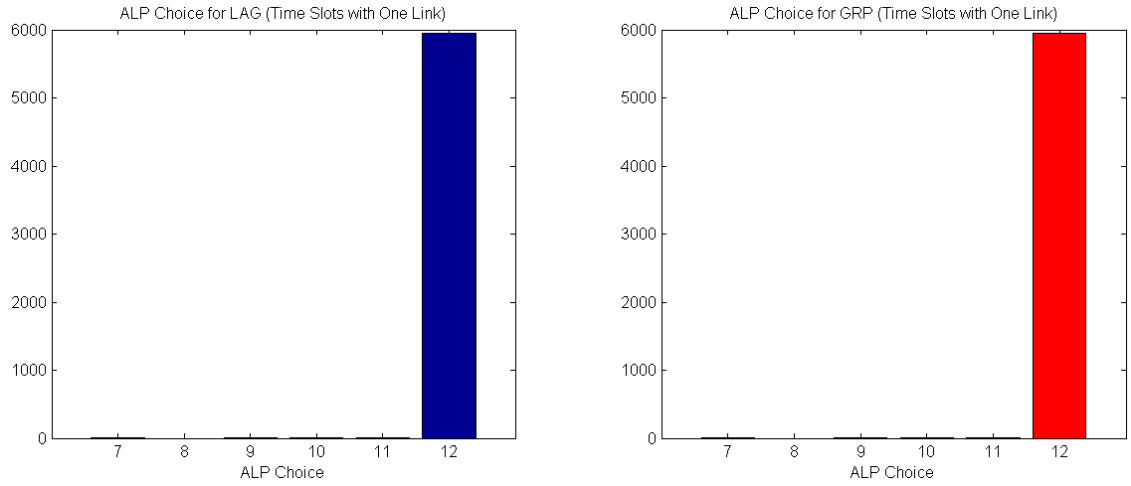


Figure 5.2: Histograms of the ALP chosen by LAG (left) and GRP (right) for links that were alone in a time slot in the 1000 iterations.

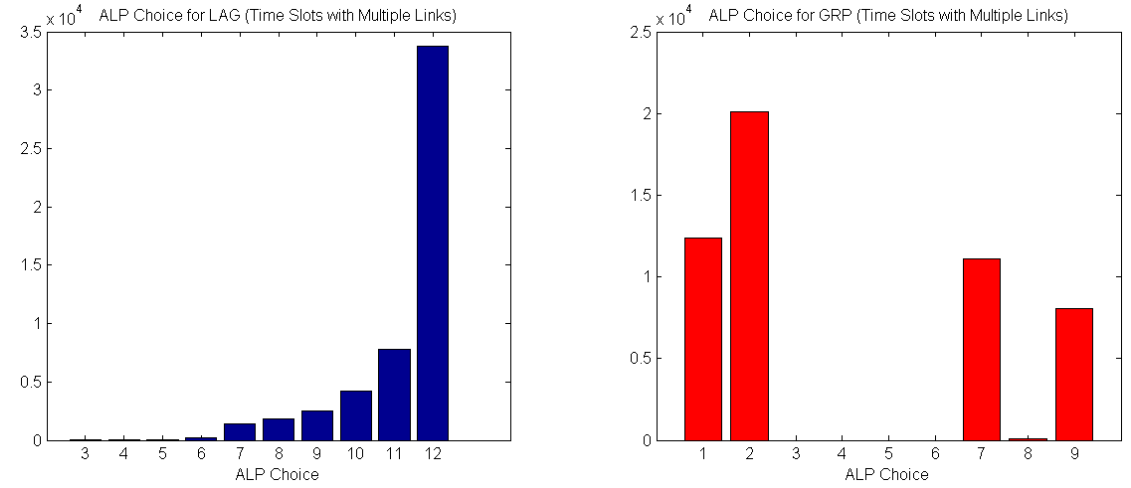


Figure 5.3: Histograms of the ALP chosen by LAG (left) and GRP (right) for links in a time slot that had at least one other link in the 1000 iterations.

Next, we observe the CDFs of the system throughputs, presented in figure 5.4. It can be seen that the LAG achieves much higher system throughputs than the GRP, which is explained by the histograms discussed above. The higher ALP choices are directly proportional to higher rates. Looking at the CDFs split by the number of links in a time slot, presented in figure 5.5, we can see the same results as presented in the histograms. The throughput of the links alone in a time slot are almost equal, the difference coming from the small difference in

resulting SINRs from the LAG and GRP, while there is a vast deviation in the throughput of links in time slots with more than one link.

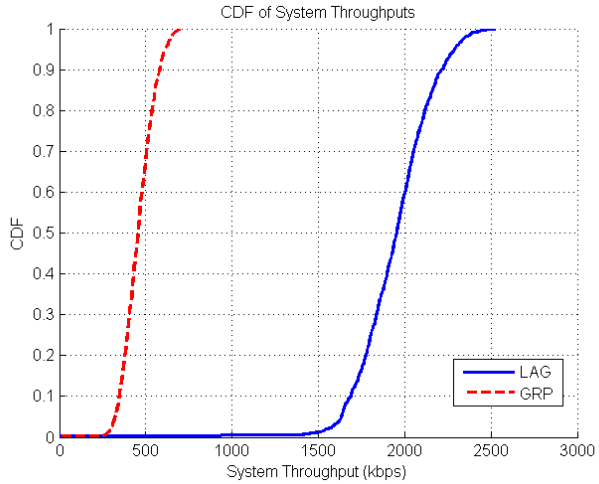


Figure 5.4: CDFs of the system throughput for both LAG and GRP.

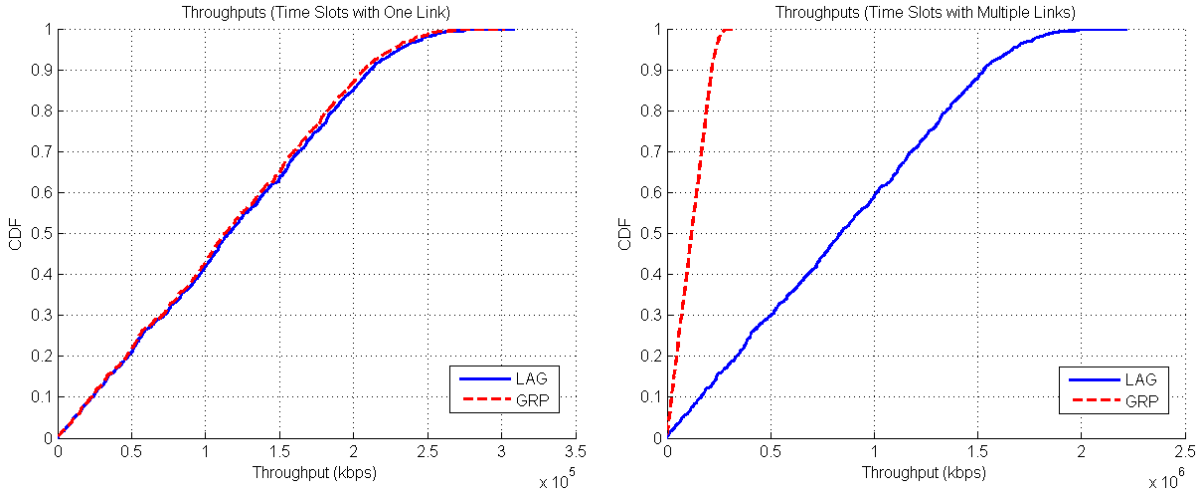


Figure 5.5: CDFs of the throughput for links that were alone in a time slot (left) and for links that had at least one other link in the time slot (right).

Like the TC game, the convergence rate of LAG is dependent on a threshold value that is used to determine when deviations in a players decision of transmit power are negligible. In our simulations, this threshold is compared to the sum of the changes in transmit power level made by each link in that round. The average number of rounds until convergence for

three different threshold values is shown in Table 5.2, organized by the number of links in the time slot. Obviously, the time slots with only a single link converged immediately. It can be seen however, that due to the relatively small number of links in each time slot, the number of rounds to convergence are quite small. Convergence rates were only reduced by approximately one round when the threshold was increased from 0.001 to 0.01 W. Because LAG can be run in parallel for each time slot, the overall rounds until convergence for a full sample case remains small.

| Threshold/Number of Links | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0.001 | 1 | 4.99 | 5.64 | 5.67 | 5.6 |
| 0.01 | 1 | 3.74 | 4.47 | 4.59 | 4.63 |
| 0.1 | 1 | 2.69 | 3.31 | 3.50 | 3.55 |

Table 5.2: Average number of rounds to convergence, by threshold value and number of links per timeslot.

# Chapter 6

# Conclusion

SDRs and CRs have become popular research topics in wireless networking, as their flexibility potentially allows for significant improvements in performance. However, these improvements often cannot be achieved within a standard layered architecture, such as the OSI model. This has resulted in much research on cross-layer design, with the general idea being that by violating the layered architecture in some way, designers can take better advantage of the advanced PHY layer technology offered by SDRs and CRs, and in turn, achieve improvements in performance.

Game theory has proven to be an important set of tools for analyzing the behavior of selfish users, and thus applies nicely to CNs, where each node is attempting to achieve the best performance. Using game theory, we were able to design a cross-layer control mechanism that can be used to control SDRs and CRs. In our mechanism, game theory is used for both topology control and link adaptation (via transmit power, modulation scheme, and channel coding rate), while also using a distributed (non-game theory) TDMA scheduling algorithm. The mechanism is split into three parts, the TC game, the distributed TDMA scheduling algorithm, and the LAG.

In the TC game, nodes use their utility function to weigh the costs of their links versus

their average path length to other nodes in the network and remove unnecessary links. We were able to show that the TC game is actually a potential game under certain conditions and that the distributed adaptation process converges to a NE. However, even when the conditions to guarantee a potential game were not met, in many cases the game was still a potential game and would still converge to a NE. Through simulations, we also showed that under certain link cost conditions, the results of the TC game were similar to those obtained by a genetic algorithm used to solve for the topology that maximized the potential function. However, we did find that the topologies that resulted from the TC game were quite sparse.

Once the TC game has determined which links should be supported by the topology, this set of links is passed to the TDMA scheduling algorithm. The scheduling algorithm is run on the resulting topology from the TC game, and then that topology and the TDMA schedule are passed on to the LAG. We did not use game theory in the scheduling algorithm, instead we adapted the distributed node TDMA scheduling algorithm DRAND into one that schedules individual links. Simulations showed that while the average number of time slots required was a bit high, it was still realistic. This number could also be reduced if relaxed constraints were used in determining which links could be scheduled in the same time slot. Another benefit to our control mechanism was that our scheduling algorithm can easily be replaced by any other link scheduling algorithm.

Lastly in our control mechanism, we incorporated the LAG from [30]. In this game, each link optimizes their transmit power and their ALP (combinations of modulation scheme and channel coding rate). It was shown through simulations that the LAG can achieve much better throughput than GRP.

## 6.1 Future Work

There are a few things that can continue to be explored with this topic. First, the actual implementations for the cross-layer interactions need to be designed. Also, a prototype

implementation of the mechanism, possibly in CORNET, would be useful in determining the effectiveness of our approach. The biggest obstacle to a prototype using the USRPs in CORNET is the design of the MAC layer. Two of the most suitable software options with which to design the MAC layer are GNU Radio and Iris. While the Iris architecture seems to be the better suited one for MAC layer design, there is far less work available on the subject with which to build. Lastly, the scheduling algorithm may be adjusted to provide a greater benefit to links with higher priorities by allowing certain links to be assigned multiple time slots.

# Bibliography


[1] I. Mitola, J., "Software radios: Survey, critical evaluation and future directions," *IEEE Aerospace and Electronic Systems Magazine*, vol. 8, no. 4, pp. 25–36, 1993.

[2] "What is software defined radio?" http://www.wirelessinnovation.org/introduction_to_sdr.

[3] J. Mitola and J. Maguire, G.Q., "Cognitive radio: making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, 1999.

[4] V. Srivastava and M. Motani, "Cross-layer design: a survey and the road ahead," *IEEE Communications Magazine*, vol. 43, no. 12, pp. 112–119, 2005.

[5] I. Akyildiz and X. Wang, "Cross-layer design in wireless mesh networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 2, pp. 1061–1076, 2008.

[6] K. Binmore, *Fun and games, a text on game theory.* DC Heath and Company, 1992.

[7] R. Komali, A. MacKenzie, and R. Gilles, "Effect of selfish node behavior on efficient topology design," *IEEE Transactions on Mobile Computing*, vol. 7, no. 9, pp. 1057–1070, 2008.

[8] I. Rhee, A. Warrier, J. Min, and L. Xu, "Drand: Distributed randomized tdma scheduling for wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 10, pp. 1384–1396, 2009.

[9] M. Sheikh, M. Drieberg, and N. Ali, "Fair scheduling algorithm for wireless sensor networks," in *National Postgraduate Conference (NPC)*, 2011, pp. 1–5.

[10] S. Mahfoudh and P. Minet, "Survey of energy efficient strategies in wireless ad hoc and sensor networks," in *Seventh International Conference on Networking*. IEEE, 2008, pp. 1–7.

[11] J. Jaromczyk and G. Toussaint, "Relative neighborhood graphs and their relatives," *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1502–1517, 1992.

[12] R. Graham and P. Hell, "On the history of the minimum spanning tree problem," *Annals of the History of Computing*, vol. 7, no. 1, pp. 43–57, 1985.

[13] N. Li, J. Hou, and L. Sha, "Design and analysis of an mst-based topology control algorithm," in *INFOCOM. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3. IEEE, 2003, pp. 1702–1712.

[14] R. Wattenhofer, L. Li, P. Bahl, and Y. Wang, "Distributed topology control for power efficient operation in multihop wireless ad hoc networks," in *INFOCOM. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2001, pp. 1388–1397.

[15] L. Li, J. Halpern, P. Bahl, Y. Wang, and R. Wattenhofer, "A cone-based distributed topology-control algorithm for wireless multi-hop networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 1, pp. 147–159, 2005.

[16] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment," in *INFOCOM. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. Ieee, 2000, pp. 404–413.

[17] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM Computing Surveys (CSUR)*, vol. 37, no. 2, pp. 164–194, 2005.

[18] R. Rajaraman, "Topology control and routing in ad hoc networks: a survey," *ACM SIGACT News*, vol. 33, no. 2, pp. 60–73, 2002.

[19] K. Sohrabi and G. Pottie, "Performance of a novel self-organization protocol for wireless ad-hoc sensor networks," in *IEEE VTS 50th Vehicular Technology Conference*, vol. 2, 1999, pp. 1222–1226 vol.2.

[20] T. ElBatt and A. Ephremides, "Joint scheduling and power control for wireless ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 3, no. 1, pp. 74–85, 2004.

[21] K. Sato and S. Sakata, "A distance-measurement-oriented distributed tdma scheduling algorithm for sensor networks," in *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, 2011, pp. 1–3.

[22] M. Sheikh, M. Drieberg, and N. Ali, "An improved distributed scheduling algorithm for wireless sensor networks," in *4th International Conference on Intelligent and Advanced Systems (ICIAS)*, vol. 1, 2012, pp. 274–279.

[23] V. Srivastava, J. Neel, A. MacKenzie, R. Menon, L. Dasilva, J. Hicks, J. Reed, and R. Gilles, "Using game theory to analyze wireless ad hoc networks," *IEEE Communications Surveys Tutorials*, vol. 7, no. 4, pp. 46–56, 2005.

[24] S. Buzzi, H. Poor, and D. Saturnino, "Noncooperative waveform adaptation games in multiuser wireless communications," *IEEE Signal Processing Magazine*, vol. 26, no. 5, pp. 64–76, 2009.

[25] S. V. Ginde, A. B. MacKenzie, R. M. Buehrer, and R. S. Komali, "A game-theoretic analysis of link adaptation in cellular radio networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 5, pp. 3108–3120, 2008.

[26] D. Li, X. Dai, and H. Zhang, "Game theoretic analysis of joint rate and power control in cognitive radio networks," in *International Conference on Communications, Circuits and Systems*, 2008, pp. 319–322.

[27] A. Tyagi, D. Popov, and H. Venkataraman, "Game-theoretic algorithms for power control and link adaptation in 3g-umts cellular systems," in *IEEE Region 10 Conference (TENCON)*, 2008, pp. 1–5.

[28] B. A. Fette, *Cognitive Radio Technology*, 2nd ed. Burlington, MA: Academic Press, 2009.

[29] A. B. MacKenzie and L. A. DaSilva, "Game theory for wireless engineers," *Synthesis Lectures on Communications*, vol. 1, no. 1, pp. 1–86, 2006.

[30] S. V. Ginde, "A game-theoretic analysis of link adaptation in cellular radio networks," Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2004.

[31] J. Deaton, M. Benonis, L. DaSilva, and R. Irwin, "Supporting dynamic spectrum access in heterogeneous lte+ networks," in *IEEE International Symposium on Dynamic Spectrum Access Networks (DYSPAN)*, 2012, pp. 305–316.

[32] M. J. Osborne and A. Rubinstein, *Course in game theory*. The MIT press, 1994.

[33] F. Berggren, "Power control, transmission rate control and scheduling in cellular radio systems," Ph.D. dissertation, Royal Institute of Technology, 2001.