UNIVERSITY OF CALIFORNIA
Santa Barbara

Automated Design of Optimal Medium Access Control Protocols for Wireless Networking

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

by

Jian Zhen

December 2014

The Dissertation of
Jian Zhen is approved:

Professor Forrest Brewer

Professor Michael Melliar-Smith

Professor Louise Moser

Professor John Shynk

Professor Volkan Rodoplu, Committee Chairperson

March 2014

Automated Design of Optimal Medium Access Control Protocols for Wireless Networking

To My Beloved Family

Acknowledgements

When I was a Master's student in Spring Quarter 2009, I took ECE 250 from Professor Rodoplu. In that course, I learned a lot in the area of wireless communication and networking, and read for the first time in my life, many scientific papers. In contrast to the other courses I was taking at the time, in Professor Rodoplu's course, we discussed a lot and were encouraged to contribute our own ideas, something that I enjoy very much. Professor Rodoplu believed that I had research potential, and encouraged me to pursue the Ph.D. degree. I am now glad that I made the decision at that time to continue my studies for a Ph.D.

First, I would like to express my gratitude to my advisor, Professor Rodoplu, for his guidance, support, and encouragement throughout my M.S. and Ph.D. studies. This dissertation would not be possible without his advice and guidance. During the past few years, I have learned a lot from Professor Rodoplu: how to read scientific papers critically, how to think independently, how to create my own research idea, how to write papers, and how to give a presentation. All of these helped me become an independent researcher. But the most valuable gift Professor Rodoplu has given to me is the long discussion hours spent with him. Looking back, it is those numerous discussions which improved my academic capability and brought me confidence in scientific research.

Then, I would like to thank Professor Brewer, for his eye-opening inspiration and strong belief in me. Without the help and support from him, I would not have been able to connect the dots and find the creative solutions I have found.

I also like to thank Professor Moser, Professor Melliar-Smith and Professor John Shynk for their general guidance and valuable advice. I am very lucky to have them on my PhD committee. Their valuable feedback and suggestions on my PhD research have helped me improve the standard of my PhD work significantly.

Next, I would like to thank my fellow lab-mates: Xin Yang, Yichu Wang, and Peng Liu. We have spent a lot of happy and valuable time together; without your company, my time working in the lab would be much less memorable.

I would also like to thank Dongxu Zhou, my roommate and one of my best friends in Santa Barbara. We can always understand each other and have very supportive conversations. Without your warm support, it would be much harder for me to get over the downtimes during my PhD studies.

And a lot thanks to my friends: Mingzhi Lu, Xun Li, Zengbin Zhang, Chao-I Chen, Danny Fan, Wei Sun, and everyone from whose friendship and acquaintance I have benefited at UCSB. Without all of you, the days in Santa Barbara would not be so vivid and colorful.

In addition, a special thank to Xin Yang and Ying Wu for the dinners you cooked during my final stages of finishing this dissertation. The dissertation would not have been finished so smoothly without your food!

Finally, I am deeply grateful to my parents, who are incredibly wise and generous. Without your love and sacrifice, it would be unimaginable for me to accomplish this research. Without both of you, my first advisors in life for my first twenty-something years, a better foundation could not have been laid for me, my career, and my whole life.

Thank you all!

Curriculum Vitæ

Jian Zhen

Education

Sept.2009 – Mar.2014	Ph.D. in Electrical and Computer Engineering, University of California, Santa Barbara, USA
Jan.2009 – Mar.2012	M.S. in Electrical and Computer Engineering, University of California, Santa Barbara, USA
Oct.2004 – Nov.2008	Dipl.-Ing. in Electrical Engineering, Technical University of Munich, Germany
Apr.2008 – June 2008	Visiting Student, University of A Coruna, Spain (DAAD sponsored research project between Germany and Spain)

Research Experience

Jan.2010 – present	Research Assistant, University of California, Santa Barbara Project: Automated Design of Medium Access Control Protocol for Wireless Networks Impact: published and submitted 4 conference papers in the top-tier conferences and currently working on 1 journal paper for a top-tier journal in the area.

Teaching Experience

Winter 2014	Teaching Assistant,Digital Design Principles, Electrical and Computer Engineering, University of California, Santa Barbara
Winter 2013	Teaching Assistant,Digital Design Principles, Electrical and Computer Engineering, University of California, Santa Barbara
Spring 2011	Teaching Assistant, Computer Organization, Electrical and Computer Engineering, University of California, Santa Barbara
May 2010	Guest Lecturer, University of California, Santa Barbara Topics: Automated Generation of Medium Access Control Protocols: Recent Progress.

Work Experience

June 2012 – Sept.2012	Software Engineer Intern, Citrix Online, Santa Barbara, USA Mentor: Software Architect/Researcher Allan Knight, in Communication Group (1) optimization and implementation of audio/video processing in WebRTC library for iOS/Android platform under ARMv7 neon architecture; (2) test and analyze the audio/video performance under the iOS/Android system.
June 2011 – Sept.2011	Software Engineer Intern, Citrix Online, Santa Barbara, USA Advisor: Chief Scientist Albert Alexandrov, in Research Group Working on improving QoS of the new product Gotomeeting HD,

	developed a bandwidth estimation/allocation algorithm which measures the Internet bandwidth between two communicating end users in realtime and allocates the available bandwidth to different types of traffic (video, audio and screen sharing data traffic, etc.) according to their different QoS requirements.
July 2005 – Feb.2006	Engineering Intern, Siemens AG, Munich, Germany Worked in DVB-T mobile handheld device development team

Professional Memberships

2008 – 2014	IEEE student member
2011 – 2014	IET Networks/Communications reviewer
2012	Session Chair, Globecom 2012, Dec. 2012
2013	Session Chair, Globecom 2013, Dec.2013

Selected Publications

GLOBECOM13	Jian Zhen, Volkan Rodoplu: "Automated MAC Protocol Generation under Dynamic Traffic Conditions," in *Proc. IEEE GLOBECOM*, Dec 2013.
GLOBECOM12	Jian Zhen, Volkan Rodoplu: "Automated MAC Protocol Generation for Dynamic Topologies," in *Proc. IEEE GLOBECOM*, Dec 2012.

GLOBECOM11	Jian Zhen, Forrest Brewer, Volkan Rodoplu: "Automated MAC Protocol Generation with Multiple Neighborhoods and Acknowledgments Based on Symbolic Monte Carlo Simulation," in *Proc. IEEE GLOBECOM*, Dec 2011.
GLOBECOM10	Jian Zhen, Forrest Brewer, Volkan Rodoplu: "A Methodology for Optimal MAC Protocol Generation: Case Study of a Synchronous MAC Channel," in *Proc. IEEE GLOBECOM*, Dec 2010.

Honors and Awards

2013 – 2014	ECE Department Dissertation Fellowship, University of California, Santa Barbara.
2012 – 2013	Doctoral Student Travel Grant from Graduate Division, University of California, Santa Barbara.
2009 – 2013	President's Work Study Award, University of California, Santa Barbara. (4 times)
2005 – 2007	Scholarship for Foreign Students from the Bavarian Government (5 times), Germany

Activities

2010 – 2013	CSSA Badminton Group coordinator, University of California, Santa Barbara.
2010 – 2011	Vice president of Chinese Scholar and Students Association (CSSA), University of California, Santa Barbara.

Languages

Chinese: native speaker

English: full professional proficiency

German: full professional proficiency

Abstract

Automated Design of Optimal Medium Access Control Protocols for Wireless Networking

Jian Zhen

We present a framework for the automated design of optimal Medium Access Control (MAC) protocols for wireless networks.

First, we describe a methodology that incorporates the impact of control information transfer into MAC protocol optimization. We apply this methodology to the problem of a synchronous broadcast MAC channel in order to generate the optimal protocol when the objective function is the average network throughput per time slot. We describe a recursive procedure for the symbolic generation of the optimization program for any choice of the objective function. We demonstrate that this methodology subsumes two structurally different types of protocols, namely, pure random access protocols and protocols with data advertisements, as special cases of the regimes where they are optimal. We examine the scaling of the optimal throughput and the computational complexity as a function of the number of nodes and the control lifetime.

Second, we generate optimal MAC protocols based on a more general MAC model that incorporates multiple MAC neighborhoods as well as acknowledgments. In this model, both the advertisement and acknowledgment frames are automatically generated by an optimization program that is built based on symbolic Monte Carlo simulation. The design flow chain produces an optimal MAC protocol with respect to the desired objective function.

Third, we formulate the automated optimal MAC protocol generation problem for dynamic topologies, as encountered in wireless ad hoc networks, under multiple neighborhoods and in the presence of acknowledgments. The probability distribution over the set of local topologies encountered in the global network serves as a model for which an optimization program may be formulated that takes the per-node average throughput as its objective function. Symbolic Monte Carlo simulation is used to generate the optimization program, which is subsequently solved via state-of-the-art nonlinear solvers. A quantitative comparison with the standard RTS/CTS protocol provides information on the value of side information on the probability distribution of local topologies, which RTS/CTS does not presume. Our investigations of computational complexity show that the time to generate the program dominates over the time to solve the resulting nonlinear program, and that the complete program can be solved within a reasonable computational time.

Fourth, we formulate the automated optimal MAC protocol generation problem under dynamic traffic conditions for multiple neighborhoods and in the presence of acknowledgments. We show that the problem can be formulated as a functional optimization program in which each design (a.k.a. decision) function of the program is the probability that a node takes an action given its knowledge state, as a function of the effective traffic demand at the current time at that node. In order to achieve a viable computational complexity for the functional optimization program, we discretize the effective traffic demands by virtue of which a look-up table is produced for each design function. Structurally different MAC protocols can be represented in this framework, and are generated automatically with respect to traffic demand. The symbolic Monte Carlo method is used to generate an approximate expression for the objective function as well as for the non-linear constraints, in a manner that trades off accuracy versus computational complexity. Symbolic simulation results are presented for a fixed network topology under the assumption of Poisson traffic. The objective is to minimize the average power consumption of a node subject to a minimum average throughput constraint that incorporates soft delay guarantees. Our research demonstrates that a MAC protocol that incorporates acknowledgments in a multi-hop setting under dynamic traffic can be generated automatically.

This thesis opens the way for the design of an automated design flow chain for network protocols that are based only on local information, of which MAC protocols constitute an example. In the future, our framework can be integrated

as a “back end” to Software Defined Networks (SDN’s) which are envisioned to run on optimizable protocols as the ones described in this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless networking is ubiquitous today, ranging from Wi-Fi networks, 4G (Fourth Generation) cellular networks, to wireless networks in specialized domains such as RF military networks, underwater acoustic networks, sensor networks, as well as newly proposed domains such as inter-vehicular networks [1]. It is expected that wireless networks will continue to proliferate and to penetrate our lives through smart-phone usage and short-range wireless technologies [2] that will deliver much higher data rates at the 60 GHz carrier frequency in the near future.

Wireless networks have inherited the layered model of design [3] from wired networks, which is shown in Fig. 1.1. In this layered model, a.k.a. the "protocol stack," the lowest layer is the Physical Layer (PHY) whose job it is to create an abstraction for the reliable delivery of individual bits to the upper layers. The Data Link Layer (DLL), built on top of the physical layer, takes blocks of bits called "frames" as its Protocol Data Units (PDUs) that are to be delivered

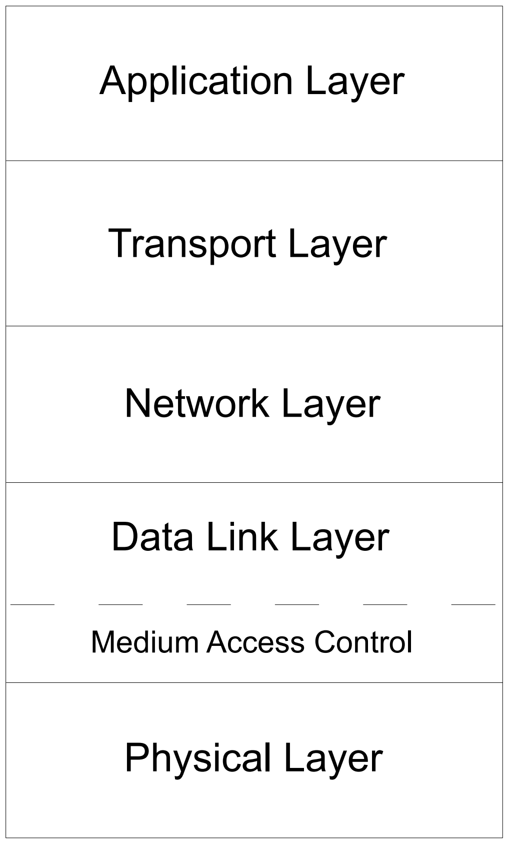


Figure 1.1: TCP/IP protocol stack

reliably. If frames get lost due to channel errors, then those frames would need to be re-transmitted, which is the job of the Data Link Layer. The Medium Access Control (MAC) Layer solves the problems of achieving access of multiple nodes into the same wireless medium. The MAC Layer is the first layer at which the distinction between wired and wireless networks becomes significant, as wireless networks allow multiple nodes in the same vicinity to access the same bandwidth, whereas wired networks contain the electric fields within wires and do not have the "broadcast effect" of wireless networks. This characteristic makes wireless networks much harder to design, and sources of inefficiency at the MAC Layer translate to big losses in performance at all layers of a wireless network. The Network Layer, as it appears in this more prominent TCP/IP stack (rather than the original OSI stack [4]), addresses the delivery of "packets" of data from one node to another node (usually far away) across the network.

The Internet is based on the phenomenal success of the Internet Protocol (IP), the most important and famous of all network protocols and the model upon which most of the networking protocols in other domains have been built. The Transport Layer addresses the end-to-end reliable delivery of packets. In the Transport Layer view, the routers that make up the network are abstracted, and only the end-to-end reliability from the source to the destination is considered. The Transport Control Protocol (TCP) of the TCP/IP suite is the most important one that has been developed and the one still in use. Even though it has been found not to be suitable for wireless networks, TCP still serves as the benchmark transport protocol for end-to-end reliable delivery of data.

The Application Layer, built on top of the Transport Layer, is the layer at which individual applications become visible in the protocol stack. The success of the Layered Protocol Model (or Protocol Stack) has been phenomenal since its birth in the 1970's, and we will use this layered model as one of the key assumptions of this thesis. Even though this thesis involves only the MAC Layer, understanding the usefulness of the approach developed can only be achieved when viewed in the larger context of this layered protocol stack.

As we mentioned, inefficiency at the MAC Layer of wireless networks typically translates into big losses in the overall performance of the protocol stack. Despite the enormous emphasis on the design of the Physical Layer since the 1950's [5][6] and the number of sophisticated technologies [7] developed for the Physical Layer, the MAC Layer has received relatively less emphasis and there has been only a handful of MAC protocols [8] that have been used in wireless standards. A part of the reason for this has been to keep wireless MAC protocols simple. Wireless MAC protocols must operate within a locality. Especially if the wireless network is mobile, the neighbor relationships change; as a result, the emphasis on a *local* protocol that can be applied to *any* neighborhood across the network translates to a simple one that addresses effectively the access of nodes only in that locality. Hence, the types of topologies that can occur and be persistent for reasonable durations before topology changes occur due to mobility are limited: For example, for cellular networks, the only relevant topologies are direct hops between the base station of a cell and the nodes within that cell. For a Wi-Fi network, the relevant topologies are those that have a direct link between the access point, and the nodes. In Wi-Fi deployments, there is no direct link[1] between the nodes themselves. When the nodes in the network have direct links with each other, we call such networks "ad hoc networks". The MAC Layer problems for cellular and (non ad-hoc) Wi-Fi networks are more straightforward, and to some extent, subsumed by the more general "ad hoc network MAC Layer" problem that allows for nodes to receive transmissions from each other without the need for a base station or an access point, such as Wi-Fi Direct. This thesis focuses on the MAC Layer problem for ad hoc networks, as opposed to cellular or Wi-Fi networks with access points.

The precursor of wireless ad hoc networks was the military packet radio networks [10] in the 1970's. The advances [11] in Complementary Metal Oxide Semiconductor (CMOS) technology in the 1980's enabled *low-power* Very Large-Scale Integrated (VLSI) circuits, which translated into low-cost silicon for portable wire-

[1]Wi-Fi Direct [9] is an exception to this since it uses ad hoc connections.

less devices. This technology enabled the wireless revolution [12] in the late 1990's, and wireless ad hoc networks became popular, now for civilian networks, as sophisticated, integrated technologies such as GPS (Global Positioning System) receivers [13], turbo decoders [14] for modems, and wavelet transforms [15] could now be fit into a hand-held device at low power. This advance raised the question of whether it would be effective for nodes to communicate to each other directly rather going through a wired backbone, that is, the possibility of wireless ad hoc networks.

The 2000's saw the rise of the 3G (Third Generation) and 4G (Fourth Generation) cellular wireless standards, as well as the proliferation of Wi-Fi networks in homes and offices. Cellular networks have evolved to control data access for maximum data rates. High data rates are most easily achieved when the base station allocates the bandwidth in advance using very tight control over the hand-held devices. The MAC Layer issues, in this setting, rarely come up for cellular phone networks, except for the Random Access Channel that a phone uses to access the base station for the first time, when it is turned on. Because cellular phone networks spend the majority of their time in the deterministic access mode, the MAC Layer issues in these networks are not considered significant. In contrast, in Wi-Fi networks, the MAC Layer plays an important role. If ad hoc networking is enabled for Wi-Fi, called the Distributed Coordination Function (DCF) mode, the

MAC Layer issues become significant. Even though this thesis constitutes fundamental work on the design of the wireless MAC Layer, its immediate application would be to Wi-Fi networks in the DCF mode.

The inefficiency of the MAC Layer for Wi-Fi networks is well-documented: For example, results in [16] on the IEEE 802.11a standard, one of the first Wi-Fi standards from the late 1990's, clearly display the MAC layer inefficiencies empirically. These MAC Layer inefficiencies have persisted all the way to the newest Wi-Fi standard, IEEE 802.11ac [17], where, according to the standard, Physical Layer raw data rate is above 500 Mbps; however, the achieved data rates are 15 Mbps for the downlink, and 5 Mpbs for the uplink. In other words, there is a 10 to 100-fold loss in going from the Physical Layer to the Application Layer, a large part of which is attributable to the MAC Layer. Both the industry and academia have taken piecemeal approaches to improve MAC Layer performance. For example, one of the most common approaches is to tweak the MAC Layer in wireless LANs (as was done with 802.11n networks [18],[19]) so as to drive the loss from around 90% to 58% [20]. The main problem is that unlike the Physical Layer, the MAC Layer lacks measures of "optimality" and "capacity"; hence, rigorous design of the MAC Layer (as could be done for the Physical Layer) has so far not been possible.

Cross-layer design [21],[22],[23] was proposed in the 2000's, as a method to design the protocol stack optimally. Such cross-layer decision does not entirely break down the layered abstraction model, but rather uses the exchange of important parameters between the layers so that each layer could be optimized. This clearly increases the complexity of protocol design significantly, but more importantly, even for a single layer, the notion of optimality does not concern the control plane. Because no optimization theory currently exists that could model the impact of control packet exchanges on data, the cost of exchanging control packets in the network cannot be incorporated into these cross-layer design frameworks. As a result, a design of a layer claimed to be "optimal", might, in fact, result in such a significant amount of control packets generated that it might completely sweep away any real benefits that will accrue in the data plane.

One of the pioneering approaches to cross-layer design, that is still based on the assumption of not incorporating the impact of control into optimization, is the Layering via Optimization Decomposition [24] framework, which shows how cross-layer design can rigorously turn into a layered architecture. In this approach, the optimization problem (taking only data flows into account) is first formulated across multiple layers of the protocol stack, treating the whole stack as if it were only one layer. By decomposing the optimization problem into smaller problems that interface with each other via "price signals" (whereby each layer indicates the price of making certain decisions), the assumed single layer is “decomposed” into multiple layers in an optimal fashion, taking into account only data flows, and not the cost of any control signals that would need to exchanged between the nodes. This approach uses only classical decomposition techniques in optimization, and has been used as a basis to claim the “optimality” of the layered approach. Examples have also been given to show how existing protocols can be reverse-engineered in order to show under what scenarios they can be claimed to be optimal. One of the problems with this formulation, however, is the usage of “utility functions” to characterize optimality. These utility functions have to be ingeniously engineered in order to produce a reasonable solution that is then claimed to be optimal. As a result, at least in reverse engineering protocols[25], the complexity of protocol design is pushed towards the ingenious design of utility functions for which, when the network is optimized, will lead to reasonable protocols.

For the MAC Layer, the main drawback of the Layering via Decomposition approach is the fact that it does not incorporate the impact of control information into optimization. However, for the MAC Layer, the control information exchanges and their impact on performance are the most significant of all the layers. The MAC Layer, when not properly designed, can be very costly in terms of control frame exchanges. After all, it is the control frames that determine whether and when nodes are available, and when each one can transmit. Furthermore, when MAC protocols are applied to sensor networks, the control frames can take up as much bandwidth and energy as the data frames.

The aim of this thesis is not an incremental advance over methods in particular wireless networks but rather the design of a general framework that will optimize the MAC Layer in a manner that directly takes into account the impact of the decisions made in control channels on the resulting data rate. Such a framework currently does not exist. If such a framework were designed, it would have enormous implications not only for wireless networks, but also for many different subfields of electrical engineering such as (1) controls, which use coordinated nodes that must communicate with each other and make joint decisions, (2) artificial intelligence, where multiple robots communicate with each other to make individual or joint decisions, and (3) network science, in which the communication costs of exchanging control information can now be directly modeled. The development of the rudiments of such a framework is one of the main tasks of this thesis. We focus on the problem of modeling the impact of control information exchanges on the decisions that the nodes make, within the context of only the MAC Layer in this thesis; however, we keep the above broader implications in mind.

The methodological contributions of this thesis are two-fold:

1. The incorporation of the impact of control information exchanges into optimization, and the solutions of such optimization problems for the MAC Layer.

2. The development of a novel method, called "Symbolic Monte Carlo Simulation", by which the objective function of an optimization program can be built symbolically by traversing the state space of the network.

These methodologies are applied to solve the problem of *optimally* designing the MAC Layer, with the eventual goal of *automating* MAC Layer design.

Automation has significant advantages: When systems become very complex, it becomes extremely difficult to manage complexity, not to mention design and operate a system optimally. One of the key drivers of "Software Defined Networking" (SDN) in recent years [26] has been the recognition of how complex networking protocols have become, and the lack of modularity in the design and maintenance of networking protocols. Even though automation of networking protocols has not been a driver of Software Defined Networking, it appears as the next step that Software Defined Networking can take. In some sense, what networking lacks is the kind of "silicon compiler revolution" [27] that occurred in the 1980's, whereby high-level goals could be mapped optimally to low-level architectures in a way that could be optimized via the use of synthesis tools. Networks have grown large, and in a sense, the complexity problems that are faced by the networking community today are not unlike the complexity problems that were faced by the VLSI community in the 1980's and 1990's, when chips went from thousands of transistors to 10's of millions of transistors. The need for electronic design automation (EDA) [28] arose as a result, which was met by companies such as Cadence and Synopsys. The main problem today is the enormous growth of the Internet, now increasingly wireless, and the complexity problems that arise with its design, maintenance, management, and performance.

In EDA, the designer puts in a high-level description of the design conception in a hardware description language, such as Verilog. Using this high-level description in Verilog, the synthesis tools generate hardware connections on a given target architecture, such as a Field Programmable Gate Array (FPGA), or using standard cells from a design library. The goals and the methodology of network protocol automation is quite different from this methodology: In network protocol automation, we are interested in being able to generate an optimal protocol, given a certain communication model (that describes the MAC Layer in our case), with an objective function that is a combination of network-specific metrics such as throughput and energy, subject to constraints such as achieving a minimum throughput required per flow. One of the main differences is that while in EDA, the final design is synthesized from a set of existing modules (as in the case of standard cells), there is no library from which protocols can be easily synthesized. For network protocols, because the synthesis would be for software rather than hardware, it is also much more efficient for the designer to input a high-level description of the protocol in question from which an optimized description can be produced, than to attempt to synthesize it from a collection of lower-layer pre-designed modules. The designer inputs, not a specific protocol, but rather a "skeleton", whose parameters are open to optimization. While this is not the first attempt to optimize the parameters of a given networking protocol, it is one of the first attempts to unify the different protocols under the same design framework so that each can be generated as instances of a much larger family.

In Chapter 3, we give our first example of the results of our framework where a family that spans both random access control protocols (such as ALOHA) [29] and controlled access protocols that use advertisements of data transmission is specified as the "skeleton" for protocol generation. It is then shown that our optimization framework can generate a throughput-optimal protocol that is a random access protocol when the number of data frames to follow the advertisement is 1, and an advertisement protocol when the number of data frames that follow the advertisement is greater than 1. This example is the simplest example of how an optimization program that can incorporate the impact of control information exchanges can result in *structurally* different protocols. Put another way, structurally different protocols can be subsumed under the same umbrella framework *when the control information exchanges are incorporated into optimization.* Under different circumstances, the optimal protocol generated then turns out to be different. Furthermore, when the conditions change, the optimization can be run easily again, to produce a new, optimal protocol. Such quick generation, and reconfigurability, as well as guarantees of optimality, cannot be found in other current network protocol design frameworks. While the job still rests on the designer to write the protocol "skeleton" in the general way that will subsume different protocols under the same umbrella, the resulting benefits once the program has been written will be significant.

The aim of this thesis is not the development of such automated network protocol design tools, which would take a whole generation of network designers. In developing such tools, significant effort would need to be spent in the "front end" of the design, which would include building the interfaces, and the programming languages that would be used to design protocols. Network protocol design severely needs new languages to express higher-level data abstractions than those currently provided by C, C++, or Java, the main languages used to implement network protocols. The recent Software Defined Networking (SDN) efforts may lead to new advances in this regard, by building better languages that express higher-level networking abstractions. In contrast, this thesis focuses on the meat of the matter, which is the "back end" of these technologies, and can be used to optimize network protocols, within the larger framework of Software Defined Networks. The back end consists of the actual algorithms that can optimize networks. As is the case in compilers, this constitutes the meat of the matter.

The key abstraction that we found that would help with the design of this "back end" is the notion of *probabilistic* branching. Even though probabilistic automata (a.k.a. Rabin automata [30]) are not new, the use of probabilistic automata in describing the "space" of possible networking protocols (and in particular, wireless protocols) is new. In the simplest example, a random access protocol and a protocol that uses control frames to advertise upcoming data frames can be both subsumed under the same framework if we assume that in each slot a node transmits data, transmits control information, or listens, with *certain probabilities that will be optimized.* The traditional conception of a protocol is a deterministic one, with only few random components that could include features like exponential back-off duration parameters. Instead of this traditional conception, we build our entire protocol framework on *probabilistic branching.* That is, at each slot, the nodes' actions can branch into possibilities each of which is specified with a certain probability. The "optimal protocol", then, is the specification of these optimal values of the probabilities. If these probabilities turn out to be 0's and 1's, the optimal protocols will turn out to coincide with deterministic protocols; however, this is not necessarily the case. In fact, we shall find that many "optimal" protocols make full use of this probability space that is available to them. The traditional problem of picking parameter values for certain parameters such as exponential back-off duration then turns into the problem of picking these optimal probabilities of action (as well as state transition) at each slot. In the simplest case, the exponential back-off is now implemented as follows: At each slot, under no advertisements, the node decides whether to attempt to carrier-sense the medium again, or not. Doing so results in a geometric distribution for the number of slots that pass until the node carrier senses the medium and possibly attempts to transmit again. Contrasting this with the randomly generated and then fixed exponentially back-off duration, we see that they are in fact equivalent in the limit as the discrete slotted system becomes continuous. The continuous limit of a geometric random variable is an exponential random variable. Hence, in those instances, we subsume those protocols with exponential back-off parameters, in our framework, in the continuous limit.

Traditional protocols also involve deterministic timers, whose parameter values are set as deterministic numbers. Such protocols are not subsumed in our framework. All our timers time out after a random duration that has a geometric distribution. The experience of this work has shown us, though, that the insistence on deterministic timers may be one of convention rather than anything fundamental. Through this work, timers that last random durations (based on optimal probabilities at each slot) are more amenable to optimization, and may become the types of timers of choice in future networking protocols. While determinism stresses maximum control over all parameters, such maximum control does not easily leave space for design-time optimization when we insist that structurally different protocols be subsumed under the same framework.

Chapter 3 applies this framework in our first scenario of N nodes that share a common bandwidth. These nodes are said to be in the same “neighborhood”, which means that if one node transmits, all others can hear and decode that transmission, as long as the transmissions are not colliding at a receiver, that is, as long as multiple nodes are not transmitting at the same time. The RF propagation delays are minimal, and the nodes are assumed to have been slot-synchronized, although this latter assumption can be relaxed. The main problem here is the “information asymmetry”, namely, that a node does not know when other nodes will transmit. This information asymmetry problem can be solved via the exchange of advertisement frames that nodes can use to advertise to others for how long they will send data frames over the next W slots, where W is a parameter. (W is a fixed parameter in this case.) One of the key facets of wireless transmission is that a node cannot know whether a frame that it sent has been received by an intended node, unless that node sends back an Acknowledgment (ACK) for that frame. In this chapter, we do not model ACK's (these are modeled in the following chapter); hence, the fact that a node sends an advertisement (which guarantees that it will follow it with data of that duration) does not guarantee to the node itself the control frame that it sent has been received. That control frame will not be received if it collides with other control or data frames at the intended node's receiver. However, sending control frames definitely increases the *probability* that the control frame will be received, and that the other nodes, including the intended node, will be apprised of the data transmission that is about to start. When the decisions of all N nodes are to be taken into account, the challenges associated with this optimization problem of interacting machines become clear. It also becomes clear that the probabilistic calculus of state transitions and action probabilities will be of utmost importance in formulating the framework for optimal protocol generation.

Because Chapter 3 is applies to a *single* wireless neighborhood, significant simplification is possible in the state space description of the entire network: Because nodes are symmetric in whether they are able to hear from or transmit to each other, for the entire group of N nodes, a "reduced global state space" description is possible; hence, the state space does not grow exponentially with N in this case. This reduced state space description is found "by hand"; however, being able to write down the objective function to be optimized for the whole network is still very challenging, as it involves terms for every possible transition that can potentially take place in the global state diagram of this N-node network. In traditional optimization, writing down the objective function is usually simple, because it involves simple expressions. In our case, we generate the objective function via a program that attaches transition probabilities to individual arcs on the state diagram for the N-node network, in terms of the probabilities of actions given states for individual nodes, and collects the objective function by traversing all the paths in the global state diagram and arrives at an exact expression of the objective function. Thus, most of the complexity of optimization is incurred in generating the optimization problem itself, rather than in solving it. The resulting optimization program is a non-linear, non-convex program that can then be solved by state-of-the-art non-linear solvers. The result is the optimal decision probabilities at each node. An important simplification occurs in our assumption that all the N nodes are identical (the ad hoc networking assumption). This controls renders the optimization problem polynomial-time as a function of N, the number of nodes. This problem can be generalized easily to types or classes of nodes: As long as the types or classes of nodes is $\mathcal{O}(1)$ in N, the resulting algorithm is polynomial-time.

Because MAC protocols are inherently local, scaling of protocols as a function of N, the number of nodes, is not a strict requirement. There are two scenarios of interest: First, multiple nodes may be in the same locality (e.g. the same room), forming an ad hoc network. In this case, all nodes can be assumed to hear from and transmit to each other. For the MAC protocol to successfully address the case of 30 different laptops in the same room, the MAC protocol design *must* scale as a function of N in this case. Above, in Chapter 3, the protocol design complexity scales in polynomial-time in N; hence, it effectively solves that case. Second, for mobile ad hoc networks, the neighborhood associations change frequently; hence, at any given time, the number of nodes that are in a given neighborhood is small, since the transmit power is controlled in these networks to reach only a few surrounding nodes. (Doing so helps conserve the battery of nodes.) In this case, finding the optimal protocol can be done as follows: Generate the objective function expression over small topologies that occur in each neighborhood. (These topologies are possibly multi-hop, but do not have too many hops.) Then, optimize the whole network by computing the objective function as a weighted sum of these small topologies where the weight is given by the frequency with which that topology occurs in a wireless neighborhood in a given mobile network. These are the topics of Chapters 4 and 5.

In Chapter 4, we first enhance our basic MAC model by adding Acknowledgments (ACKs) to the MAC protocol model, and by allowing multi-hop topologies. Such multi-hop topologies are achieved by disabling a subset of the links in the single neighborhood model so that those nodes cannot hear from each other. We do not model Internet routing or Data Link Layers in this thesis. As a result, the problems of relaying bits (routing) and the problem of re-transmission of data frames (Data Link Layer) are abstracted away. Such abstraction is possible by virtue of the layered protocol model laid out in the beginning of this Introduction. The MAC Layer should be able to operate on its own, and can have protocols that are indigenous to this layer, without dealing with the challenges that occur at other layers. (Hence, all the frames encountered in this thesis are MAC Layer frames.) The ACKs that are added at this layer thus serve to bring in the feature that is found in the most prevalent wireless MAC protocol, CSMA/CA, which stands for Carrier Sensing Multiple Access / Collision Avoidance.

Originally developed by Tobagi and Kleinrock in the 1970's for packet radio networks, CSMA/CA [31] adds the "collision avoidance" feature to the even more traditional Ethernet protocol used for wired networks. The main idea is that because the environment is wireless, and because the node cannot hear (as it could on a wire) whether there was a collision at the receiver, it uses a mechanism called RTS/CTS (Ready To Send / Clear to Send) to negotiate such access. In the first part, named RTS, a node sends an RTS frame to the wireless medium, to say that it has unicast data to send to a particular neighbor, whose node ID is given in the RTS frame. The key assumption is that due to the broadcast nature of the wireless channel, all nodes in that neighborhood potentially hear this RTS. The particular node for which this RTS was intended, then replies with a CTS, if at that time, there are no data transmissions that are known to be scheduled for the near future in the neighborhood of the intended receiver. Again, a key assumption is that due to the broadcast nature of the wireless channel, the nodes in the neighborhood of the second node will hear this CTS, and keep quiet for the duration that is specified in the data length duration of the RTS frame. (There is a further, optional part, where, upon receiving the last data frame of that particular transmission, the second node sends back an acknowledgment that says that the data reception is complete. However, this step is not required.) CSMA/CA is famous for having solved both the hidden and exposed terminal problems in wireless networks. It still is the most widely used MAC protocol in any wireless system today, due to its effectiveness as well as its simplicity. However, as mentioned before, CSMA/CA does not give any guarantees as to MAC protocol optimality. Such a notion of protocol optimality was not even formulable in a manner that incorporates the overhead of control information exchanges until this work. It is safe to say, however, that any protocol that claims to be optimal, must do at least well as CSMA/CA. If it does better, than this opens the way to the possible replacement of CSMA/CA with an optimally generated protocol.

In Chapter 5, we first solve the protocol generation problem for small topologies that are likely to be encountered as local topologies in dynamic mobile networks. These topologies also provide the first examples of optimal protocols, as specified by optimal transition probabilities in state diagrams that now incorporate more features such as ACKs, and that can operate in a multi-neighborhood scenario. The multi-neighborhood scenario renders the "reduced global state space" approach that was developed for the single neighborhood scenario useless. A global state space can no longer exist because each node now has its own view of the world based on the information that it receives through its particular links. This calls for an entirely new computational method that will grant each node only a local view of the world.

We called the method that we developed to solve this problem, "Symbolic Monte Carlo simulation". Monte Carlo simulation [32] is well-known as the method to simulate a complex system by feeding it uniformly distributed input vectors, and getting a characterization of its performance via simulation. In our case, we have a complex system as well, but one whose transition probabilities are functions of the probabilities of actions upon states of different nodes, the latter of which are optimization variables. Hence, the idea is to simulate this complex system and "accumulate" expressions for the objective function based on the particular paths traversed in the state diagram. If a sufficient number of the important paths can be sampled in this fashion, then an approximation of the objective function will have been accumulated.

While the idea appears easy to explain, the important catch is that the state space of such a complex network is not something that can be written down or enumerated in any fashion. The state space, which is the direct product of individual nodes' state spaces, certainly exists, but must be "generated" on demand, as a path on which the objective function is being constructed. This can be done either in a breadth-first, or depth-first manner, because in the end, not just one, but multiple paths that carry the rewards and the probability weight have to be discovered.

The "Symbolic Monte Carlo simulation" method that we developed, is thus not only the idea of accumulating expressions along paths to compute an approximation of the objective function, but also the computationally feasible way of doing this via a recursive method (that has both breadth-first and depth-first features) to explore (and to generate explicitly as needed) the state space in an intelligent manner. Despite the merit of this recursive method, the complexity of the exploration is still exponential in N; hence, the method is suitable only for small-size MAC neighborhoods as addressed in this work. Finding polynomial-time methods to solve the same problem would indeed open new vistas not only for this problem, but also for computational problems in general.

In Chapter 5, the results of the previous chapter are applied to mobile ad hoc networks that have dynamic topologies. The main assumption that underlies this chapter is that the frequencies of local topologies can be measured and collected for a dynamic network, and the optimal protocol generation can be run off-line. A comparison with CSMA/CA is presented whereupon we find that the optimal protocol does significantly better than CSMA/CA. At the same time, CSMA/CA does not have access to information on the frequency with which local topologies occur. However, the main point is that even if CSMA/CA had access to such information, it would not know how to *use* that information. Using that information requires a protocol, of the type we generate, where nodes make not deterministic but probabilistic decisions on their action space given the state in which they are. The closest analogue to this would be the p-persistent CSMA protocol, in which the node transmits not with probability 1 as in CSMA/CA, but rather with probability p into the medium. The p-persistent CSMA/CA protocol is indeed quite similar to our protocol, except that p-persistent CSMA cannot decide by itself whether and when to switch to a pure random access (ALOHA) mode, whereas our framework can.

Wireless networks also have bursty traffic patterns, which means that a node may be idle for a while, until some application (such as Facebook) places a certain demand for data. The generation of an optimal protocol for such bursty data is more difficult than the static data that we assumed so far, where the node always has something to transmit. Even though the latter models high-traffic periods, it cannot model the moderate or low-traffic periods, where traffic demands in time are random.

In Chapter 6, we address this problem. The main conceptual step that we have to add in is the model of the amount of traffic demand at a node. If this is done is a node-specific fashion, then the complexity of the resulting state space will definitely grow exponentially with N. However, using the assumption that nodes are identical (as far as their MAC Layer pictures are concerned) in an ad hoc network, we can add the local traffic demand rate at a node (e.g., the rate λ of arrivals of a Poisson traffic stream) as an *optimization parameter*. Under this scheme, an optimal protocol specifies the probability with which the node takes a given action in a given state *and* given a certain local traffic demand that it sees at that time. However, because the local traffic demand is a continuous variable, in order to make discrete decisions, it must be discretized. The intelligent discretization of the traffic demand via a vector codebook is the key new technique that Chapter 6 adds to the repertoire of techniques that we develop in this thesis to enable the generation of optimal MAC Layer protocols.

Chapter 2

Related Work

There have been only a few works in the past that have attempted to generate protocols automatically while extremizing an objective function. First, there is the work on the automatic generation of security protocols [33][34]. The primary method used in this approach is iterative deepening, where a cost threshold is set at each iteration, and a search is performed in the protocol space to generate all of the protocols below the given cost threshold. The protocol space of security protocols is specified by a grammar such that the leaves of the tree generated are either principals or public keys. The number of protocols generated in this fashion are exponential in the value of the cost threshold specified, hence multiple heuristic pruning methods are used to reduce the search space. Our approach differs significantly from this work, both with respect to the problems in the domain of application, and the techniques employed. Our main approach is to represent the control information exchanges directly within an optimization program. In particular, the “form” of the protocol for which we search is not specified a priori via a grammar, but rather extracted later. That is, the rules (or grammar) that specify the protocol are discovered later, and not specified a priori. The research challenges of (1) representing control information within optimization programs, and (2) extracting protocols via rule detection within optimal waveform sets, have the potential for generalizability to different domains if they can be demonstrated for the set of MAC protocols.

The second body of related work is layering via optimization decomposition [24]. In this approach, the entire optimization problem of network resource allocation is formulated as a large (nonlinear) optimization program. The decomposition of this problem via standard techniques into subproblems that interface via price variables can produce different layers of the protocol stack (in a vertical decomposition). In this approach, the control information exchanges are not modeled in the optimization program, but rather are produced as by-products. It is checked, only afterwards, that the protocol that results has reasonable control overhead. As such, the protocols that are produced are suitable for the regime in which the data takes up much larger resources than control information. Reference [35] has also pursued this approach where decisions of whether and when to schedule control information, and the overhead associated with these decisions, are not part of the framework. However, for wireless networks, in particular for MAC protocols for wireless sensor networks, the control information exchanges and data take up comparable resources. Our main contribution in Stage I is that we are able to represent control information exchanges within optimization programs, which is what can lead to protocols that contain control packet exchanges. In particular, when an optimization engine finishes its job at the end of Stage II, it must produce whether and when the control packets are to be scheduled. Hence, an important challenge is describing the most general form of the control information, which we address through this work. Note that in [24], the control packet exchanges that result at the output of this decomposition, have to be treated as "externalities". Second, extra care must be taken with the utility functions used in [24] as they can hide under the choice of a utility function the fundamental reasons for choosing one protocol over another in the regime where the control overhead is non-negligible. As in microeconomics, the space of utility functions is very large, and it is easy to fit a utility function to "explain" observed behaviors in situations where the actual reasons lie outside the model. For example, the fundamental reasons for choosing a reactive versus a pro-active routing protocol in mobile networks would lie outside of the framework of [24].

References [36] and [37] proposed a framework, for the first time, that incorporates the impact of control information exchanges between the nodes into the optimization of a single-neighborhood MAC protocol, for the performance metrics of throughput and energy. However, these formulations used deterministic optimization variables to model the transmission schedules of the nodes. Deterministic optimization variables cannot handle the mixture of random access and advertised transmission schedules, and the optimal solutions, which will be identical for identical nodes, will produce continuous collisions when followed by each node. These researchers solved this problem by purposely breaking the symmetry between the nodes and creating a pre-determined chain of leaders and followers that determine the order in which the control will be transmitted. However, such presumed leader election, if resolved, should practically obviate the MAC problem in the first place. Hence, a different representation of node knowledge states is required, that successfully captures the mixture of random access and deterministic transmission modes under the same optimization framework. In this thesis, we derive that representation and the corresponding optimization framework.

Similar to the concept of automated protocol generation, Ergen et.al. [38] introduced a "MAC protocol engine" to accelerate the design process, in which the designer provides the design requirements and the protocol engine chooses a particular protocol from a pre-existing protocol library. With this approach, deciding which protocols outperform others relies on extensive prior mathematical analysis for each protocol in the library, which is labor-intensive to produce. Furthermore, as new protocols are added to the library, a comparative analysis of their performance must be analyzed for the protocol engine to be able to make a choice. In contrast, in our approach, we aim at *automatic* protocol generation that does not require any complex mathematical analysis.

Other works [39] [40] have proposed a framework for automated combination of MAC protocols for unknown conditions. Reference [41] further proposed an adaptive MAC framework for dynamic radio networking. References [42] and [43] proposed a compiler-assisted approach to design MAC protocols, which can be reconfigured by exchanging and combining different components. References [44] and [45] proposed a flexible MAC development framework using "decomposable" MAC structures, which are claimed to be highly flexible and adaptive in the actual realization in the working system. Even though the two former works achieve configurability and the latter flexibility and extensibility, there is no framework for design automation and no guarantees of protocol optimality in these cases.

Chapter 3

Optimal MAC Protocol Design for a Single Neighborhood

In this chapter, we address the automatic generation of a MAC protocol for a single wireless neighborhood, in which all the nodes can hear from and transmit to each other. We assume that there are N identical nodes, with infinitely long data in their MAC buffers. We further assume that nodes have been slot-synchronized, but that they do not have any knowledge of each other's future transmissions, unless advertised through a control frame. We allow for the simplest node actions possible: In each slot, a node can transmit data, transmit control, or listen. Consider first the scenario where the control information that each node can transmit is of a single type: The control information frame indicates that this node will send data for the next W slots, where W is a parameter that we call "control lifetime", which indicates the data length that will follow the control frame.

Our main aim in this chapter is to illustrate how the framework that we shall establish can generate structurally different protocols for different values of W. There are only three different types of protocols that are subsumed under the same umbrella: First, the optimal protocol might be a pure random access control protocol that will transmit data with a certain (optimal) probability and listen the remainder of the time. The second possibility is that the optimal protocol might be one that will use explicit advertisements to send data every time that a node wants to transmit data. The third possibility is that the optimal protocol might be a mixture of random access and advertisements: It sends advertised data with a certain probability, sends pure data into the channel with another probability, and listens the remainder of the time. The key point is that in the existing protocol frameworks, the designer would typically need to make a decision, based on a priori intuition, as to which of these (i.e., sending advertisements or pure random access) would work the best. Even if the designer allowed for a mixture under the existing frameworks, each type of protocol would have to be simulated separately and then compared, as there is currently no unified framework to model the impact of control information on performance in an optimizable fashion in protocols today. We shall show in this chapter how we solve this problem by modeling the mixture as a Markov chain, whose arcs are given by expressions that are made up of the action probabilities of the nodes in this wireless neighborhood.

Now, we set up a mathematical notation for node actions. (We further enlarge the set of node actions by another type of control information, which advertises that the node will listen to the channel for the next W slots.) The set of actions of each node shall be identical and will be given by $\mathcal{A} = \{d, n, \hat{d}, \hat{n}\}$, where d stands for sending data in that slot, n for listening, $\hat{d}$ for sending a control frame into the channel that announces to the other nodes that this node will send data in the following W slots, and $\hat{n}$ for sending a control frame into the channel that announces to the other nodes that this node will listen to the channel for the next W slots. Note that we use broadcast transmission for both data and control frames; hence, if a control frame is sent successfully, it is heard by all of the $N-1$ nodes. In addition, note that W is a fixed number for simplicity, and can be advertised within a control frame. We call W the "control lifetime", namely the number of slots in the immediate future over which the control frame has any effect. In our case, W is the same for both $\hat{d}$ and $\hat{n}$ control frames, but can be made to be different for a more general optimization program. We take our optimization metric to be the average throughput, namely the long-run average of successful data transmissions per slot.

In order to explain how the impact of control information on node decisions can be modeled, we analyze the impact of control for the N-node, single neighborhood scenario, with two of the nodes A and B as shown at the top part of Fig. 3.1. For

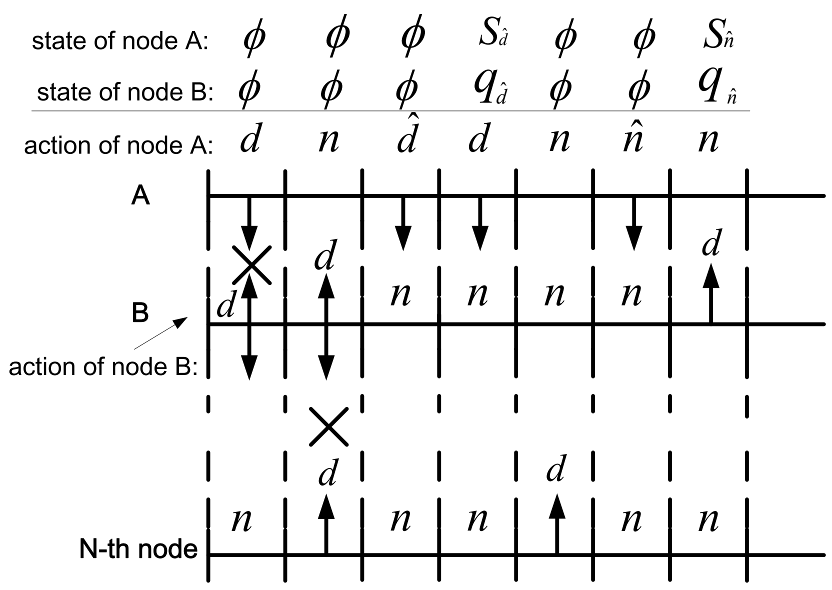

state of node A: φ φ φ S_d̂ φ φ S_n̂
state of node B: φ φ φ q_d̂ φ φ q_n̂
action of node A: d n d̂ d n n̂ n
A
B
action of node B:
N-th node
d
n

Figure 3.1: An illustration of the node actions and states, for W = 1.

this figure, the control lifetime has been set as $W = 1$. When viewing this figure, recall that d stands for the node action "transmit data"; n stands for the node action "listen"; $\hat{d}$ stands for the node action "send advertisement for data"; and $\hat{n}$ stands for the node action "send advertisement for listen".

The "knowledge state" of a node encodes whether a node has sent a control (i.e., advertisement) frame in the last W slots, denoted by the state $s_{\hat{d}}$; whether it has received a control frame in the last W slots, denoted by the state $q_{\hat{d}}$ or neither of these, denoted by the state ϕ.

In the figure, in the first slot, both nodes A and B transmit data, and their transmissions collide, wasting that slot for all of the nodes in the network. In the second slot, A is silent, but B collides with the Nth node. In the third slot, A sends $\hat{d}$, which advertises that it will send data d in the next W slots ($W = 1$ here). As a result, A's knowledge state in the next slot goes from ϕ (the null knowledge state) to $s_{\hat{d}}$, which indicates that node A has just sent $\hat{d}$ in the previous slot. In this example, all of the other nodes were quiet in the third slot, and A's control frame successfully got through to all of the nodes, and because control frames are transmitted in broadcast mode, each of these nodes switches from ϕ to $q_{\hat{d}}$ which indicates that it has just received a control information frame $\hat{d}$. Due to the broadcast mode, there is no need to mark from whom this control frame was received. Note that node A's switching to knowledge state $S_{\hat{d}}$ depends only on A's sending $\hat{d}$ in the previous slot, and cannot depend on successful reception of this $\hat{d}$ (which happens to be the case in this example), of which A can have no knowledge due to the wireless nature of the channel. (No acknowledgments are modeled here. They will be added to the model in the next chapter.)

The global knowledge state of the network at any time is the Cartesian product of the Markov chains of these N nodes. In Fig. 3.2, we have drawn the Markov chain for the global knowledge state of the network for $N = 2$ and $W = 1$. The doubles in the state bubbles correspond to the pairs of the knowledge states of the individual nodes. In this diagram, there are 4 states besides the ϕ knowledge state: These states correspond to the cases where one of the nodes has successfully sent control (either $\hat{d}$ or $\hat{n}$) to the other node (indicated by $S_{\hat{d}}$ or $S_{\hat{n}}$, respectively), and the other node has received this control frame, and hence transitioned in its own state space to a $q_{\hat{d}}$ or $q_{\hat{n}}$ state.

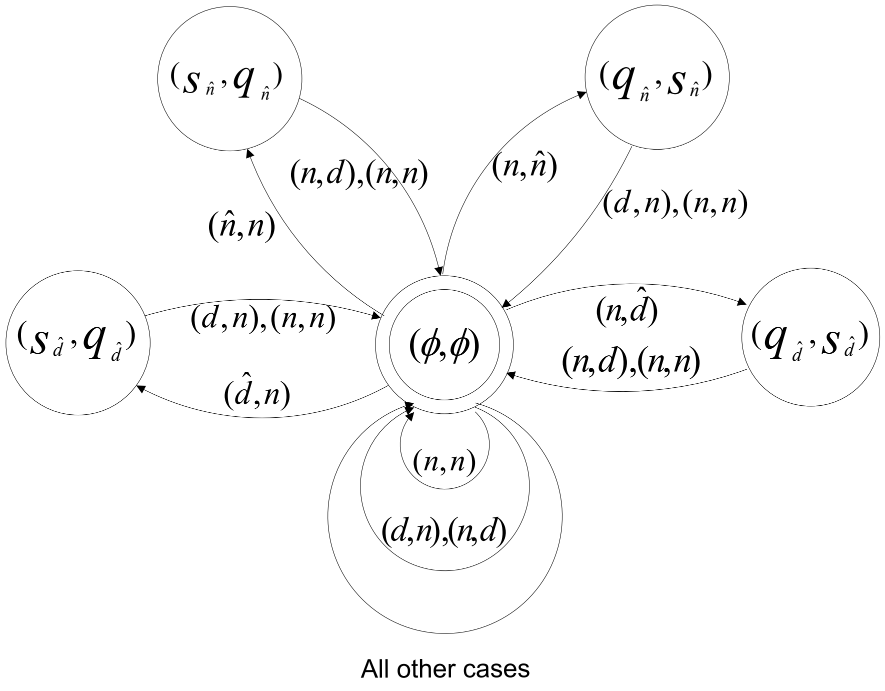


Figure 3.2: Global State Space for N=2, W=1

For the general case of N nodes, both the knowledge states of the nodes and the global state given by their Cartesian product are well-defined, however, difficult to draw. Based on this general description, an important state space reduction occurs when we note that due to the broadcast nature of the control frame transfer, the state of a node is uniquely given by one of the following 5 states, $\mathcal{S} = \{\phi, s_{\hat{d}}, s_{\hat{n}}, q_{\hat{d}}, q_{\hat{n}}^{m}\}$, where $\mathcal{S}$ denotes the state space of an individual node. The first three states correspond to the null knowledge state, the state where a node has just sent $\hat{d}$, and the state where the node has just sent $\hat{n}$. The fourth state indicates that the node has received $\hat{d}$ somewhere in the last W slots. This is the correct generalization of the above example for $W = 1$ to the case of the general W. Because $\hat{d}$ promises that this node will follow this control frame with W slots of data, a node can have received at most one $\hat{d}$ in the last W slots. In contrast, a node might have received multiple (m) $\hat{n}$'s in the last W slots; hence, the fifth state in the list is a set of states parameterized by m where $q_{\hat{n}}^{m}$ indicates that this node has received m $\hat{n}$'s in the last W slots, where $0 \leq m \leq \min(N-1, W)$. (We let $q_{\hat{n}}^{0} = \phi$.)

In general, we denote the "action set" of a node, namely, the set of actions that a node can take in each slot, by $\mathcal{A}$, and the state space of a node by $\mathcal{S}$. We let $\theta_{a|s}$, where $a \in \mathcal{A}$ and $s \in \mathcal{S}$, denote the conditional probability that the node takes action a in state s. These $\theta_{a|s}$s are the optimization variables of our program.

Based on the above reduced representation of the global state space, the transition rules from one global state to another are simply given as follows: If a $\hat{d}$ control frame has been transmitted successfully some time in the last W slots, which we represent by $I_{\hat{d}}$, then the node that has advertised this will be sending data d in the next slot. Even though all of the other nodes have the flexibility to take any action they want in the next slot, if they transmit anything in the next slot, they will collide with the d, and if all of them listen, then they will receive that d successfully. Either of these cases cannot cause a knowledge state transition in that slot for any of these nodes. As a result, the global state remains in $I_{\hat{d}}$ with probability 1. For all other cases, the probability of transition from one global state to the next, for any action $a \in \mathcal{A} = \{n, d, \hat{n}, \hat{d}\}$, is given by:

$$\binom{N-m}{1} \cdot \theta_{a|q_{\hat{n}}^m} \cdot \theta_{n|q_{\hat{n}}^m}^{N-m-1} \cdot \theta_{n|s_{\hat{n}}}^m, \tag{3.1}$$

where m is the number of $\hat{n}$ control frames that have been sent by all nodes in the last W slots, and $\theta_{n|s_{\hat{n}}} = 1$.

What we have illustrated so far is an example of the general design flow methodology that we establish in this chapter for optimal MAC protocol generation. This design flow consists of the following steps:

1. The impact of successful transmission of control information is represented by the "knowledge state" of a node, that encodes what control information the node has sent into the channel, and what control information it has received. Note that because these are wireless links, a node can never be sure that the control it has sent has been perfectly received.

2. The knowledge states of a node form a Markov chain, and the probability that the node takes action a upon knowledge state s is an *optimization variable*. It is key to note that these variables are defined only with respect to the local information available at that node, which is encoded in its knowledge state.

3. The Cartesian product of the Markov chains of all the nodes is the global knowledge state representation of the network. The optimizer uses this global state representation, even though no individual node has access to the global state representation.

4. Based on the symmetries of the network (such as the case of identical nodes), the global state representation is mapped to an equivalent representation, called the "reduced global state representation". This is a significant step in reducing the total number of variables of the program, as well as the computation of the symbolic expression of the objective function.

5. An objective function is given exogenously (e.g., average network throughput) as a metric defined on each arc or state of the Markov chain. It is essential to note that no symbolic expression of the objective function is given a priori (because its number of terms scales with the size of the Markov chain).

6. The objective function is symbolically computed recursively by a depth-first search down the reduced global state representation, and the optimization program is symbolically generated.

7. The generated optimization program is solved via state-of-the-art non-linear solvers.

So far, we have applied the first five steps in the above design flow. In the remainder of the chapter, we shall be concerned with the remainder of the steps, namely, generating the optimization program, and solving it.

3.1 Symbolic Generation of the Optimization Program

We may impose *any* metric of our choice on the Markov state space of the previous section. In this chapter, we shall maximize the average network throughput per slot. The network scores a throughput of 1 unit whenever only a single node sends data (d) in a slot, and all of the other nodes are quiet. Note that $\hat{d}$ does not score any throughput in that slot. (It is easy to generalize this to a model where the control information consumes only a fraction α, e.g., the header of a frame, by assigning a throughput of $1 - \alpha$ to every arc where $\hat{d}$ is transmitted.)

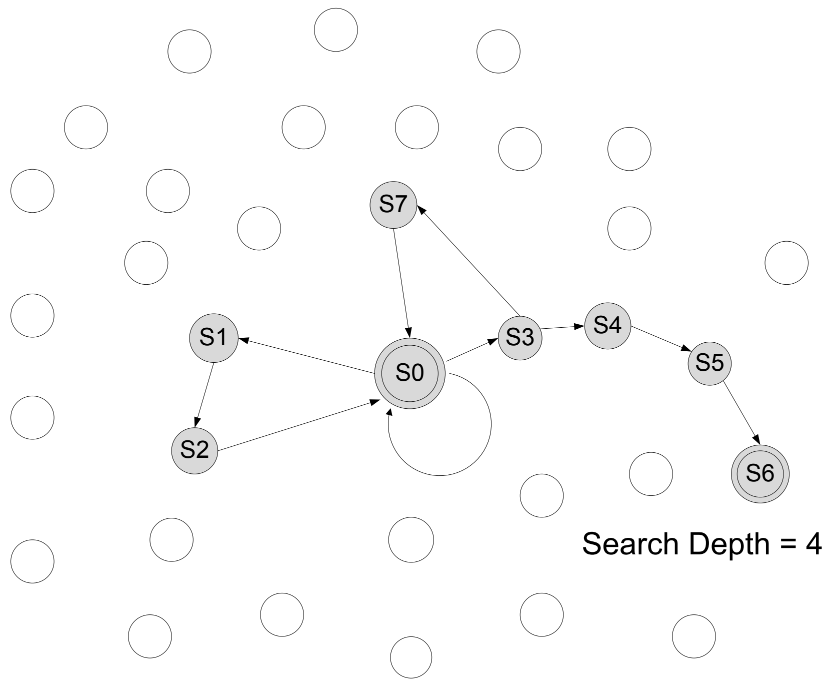


Figure 3.3: Recursive accumulation of the metrics in the AccumulateMetrics() function

The average throughput (reward) per slot on a Markov chain is calculated as follows: Define a "cycle" of a recurrent Markov chain as the pair of events from the time that the chain starts in a recurrent state to the time that the chain returns to the same state (for the first time). Let T denote the duration of this cycle. (Note that T is a random variable.) Let F denote the reward that the chain collects (on its arcs) in that cycle. That, independent of the choice of the recurrent state, the average reward per slot is computed as $E[F]/E[T]$. Hence, our first task is to generate automatically the expression for this objective function.

We use the depth-first recursive algorithm, shown in Alg.1, to accumulate and generate the symbolic expressions for $E[F]$ and $E[T]$, using the null knowledge state as the initial state of the cycle. Because the cumulative probability of a path diminishes with the number of hops in the state space, we use a maximum *search depth* N_{sd}. In this way, we trade off optimization accuracy with computation complexity.

Algorithm 1 The *AccumulateMetrics* algorithm that recursively computes the average branch cycle length T_{br} and the average branch reward F_{br}

```
 1: // $P_{\text{br}}$, average branch probability
 2: // $F_{\text{br}}$, average branch reward
 3: // $T_{\text{br}}$, average branch cycle length
 4: // $F_{\text{c}}$, cumulative reward
 5: // $N_{\text{sd}}$, search depth of algorithm
 6: // $S_i$, the $i$-th next state
 7: // $S$, the current state
 8: // $P_{S \to S_i}$, transition probability
 9: // $F_{S \to S_i}$, the reward earned during the transition
10: $[P_{\text{br}}, F_{\text{br}}, T_{\text{br}}]$ = **AccumulateMetrics**$(N_{\text{step}}, P_{\text{br}}, F_{\text{c}}, S)$
11: **if** (HasReturnedTo $S0$) or (HasHitSearchDepth $N_{\text{sd}}$) **then**
12:     $T_{\text{br}} = N_{\text{step}} \cdot P_{\text{br}}$;
13:     $F_{\text{br}} = F_{\text{c}} \cdot P_{\text{br}}$;
14:     return $[P_{\text{br}}, F_{\text{br}}, T_{\text{br}}]$;
15: **else**
16:     $\{[S_i, P_{S \to S_i}, F_{S \to S_i}]\}$ =**Neighbor**$(S)$;
17:     $N_{\text{step}} = N_{\text{step}} + 1$;
18:     **for** each next state $S_i$ **do**
19:         $F_{\text{c},i} = F_{\text{c}} + F_{S \to S_i}$;
20:         $P_{\text{br},i} = P_{\text{br}} \cdot P_{S \to S_i}$;
21:         $[P_{\text{br,i}}, F_{\text{br},i}, T_{\text{br},i}]$ = **AccumulateMetrics**$(N_{\text{step}}, P_{\text{br},i}, F_{\text{c},i}, S_i)$;
22:     **end for**
23:     $F_{\text{br}} = \sum F_{\text{br},i}$;
24:     $T_{\text{br}} = \sum T_{\text{br},i}$;
25:     $P_{\text{br}} = \sum P_{\text{br,i}}$;
26:     return $[P_{\text{br}}, F_{\text{br}}, T_{\text{br}}]$;
27: **end if**
```

Fig. 3.3 provides a picture of the state space through which the recursive algorithm branches out in its calculation. The key point is that many of the states, even though they reside in the state space, are not reachable states. Because the transitions to those states are never computed in the depth-first search, this approach obviates any a priori pruning of unreachable states from the state space. The algorithm starts from $S0$ (the null knowledge state), and calls the recursive function *AccumulateMetrics*(). In the base case, the algorithm checks the stop condition (line 11 of the Algorithm): (1) it has returned to the initial null knowledge state, or (2) it has arrived at the search depth N_{sd}. If the stop condition is satisfied, the algorithm returns the average branch probability, average branch cycle length, and the average branch reward. Otherwise, the algorithm calls the function *Neighbor()* to calculate the set of valid states to which it can transition, with associated transition probabilities and rewards, then accumulates the rewards and multiplies the probabilities for each next state; after that, *AccumulateMetrics*() function is called recursively for each next state. The total reward and total cycle length are calculated on lines 23, 24, and 25.

We have written the automatic expression generation in MATLAB, using its support for symbolic expressions. An example objective function generated this way is given below, for $N = 10$, and $W = 2$, for network throughput maximization

(The objective function is $E[F]/E[T]$):

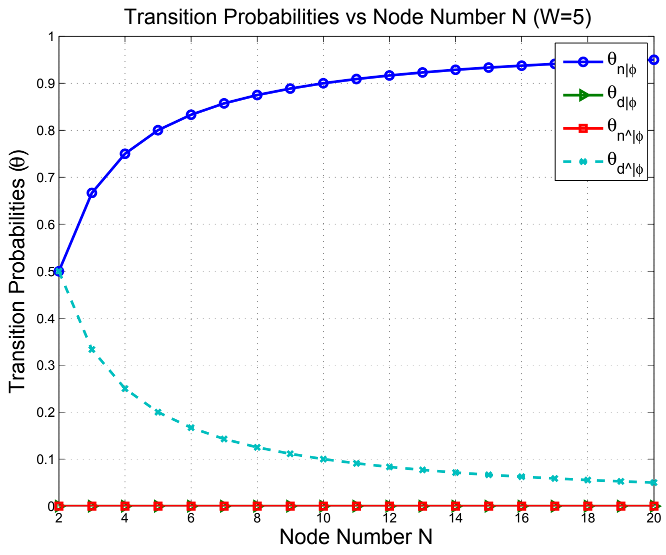
$$\begin{aligned}
\mathrm{E}[F] = {} & \theta_{n|\phi}^{9}(10\ \theta_{d|\phi} + 20\ \theta_{\hat{n}|\phi} + 180\ \theta_{\hat{n}|\phi}\theta_{n|q_{\hat{n}}}^{8}\theta_{d|q_{\hat{n}}} \\
& + 90\ \theta_{\hat{n}|\phi}\theta_{n|q_{\hat{n}}}^{8}\theta_{\hat{d}|q_{\hat{n}}} - 810\ \theta_{\hat{n}|\phi}\theta_{n|q_{\hat{n}}}^{16}\theta_{d|q_{\hat{n}}}\theta_{\hat{n}|q_{\hat{n}}} \\
& - 810\ \theta_{\hat{n}|\phi}\theta_{n|q_{\hat{n}}}^{16}\theta_{d|q_{\hat{n}}}\theta_{\hat{d}|q_{\hat{n}}} \\
& + 720\ \theta_{\hat{n}|\phi}\theta_{n|q_{\hat{n}}}^{8}\theta_{\hat{n}|q_{\hat{n}}}\theta_{n|q_{\hat{n}}^{2}}^{7}\theta_{\hat{d}|q_{\hat{n}}^{2}}) && (3.2) \\
\mathrm{E}[T] = {} & 20\ \theta_{n|\phi}^{9}\theta_{\hat{n}|\phi} + 20\ \theta_{n|\phi}^{9}\theta_{\hat{d}|\phi} + 1 && (3.3)
\end{aligned}$$

subject to the following constraints: $\theta_{d|s_{\hat{d}}} = 1$, $\theta_{n|s_{\hat{n}}} = 1$, $0 \leq \theta_{a|s} \leq 1$, and $\forall$ $a \in \mathcal{A}$, $\sum_a \theta_{a|s} = 1$. We use MATLAB's *fmincon* function, which uses sequential quadratic programming, to solve the resulting objective function. This function is non-linear because it is a rational function of two polynomials, and all of the constraints that specify that the probabilities $\theta_{a|s}$ that emanate from state s add up to 1, are linear. Hence, the result is a non-linear program, that exhibits local maxima. As a result, the choice of the starting point for the sequential quadratic program is crucial. One approach that has very good empirical performance is Monte Carlo simulation, treating the non-linear optimization program as a complex system. When the starting points are chosen randomly within the feasible set, and the maximum is retained, the procedure converges to the global optimal value with high probability; however, the convergence time can be long. A better procedure, which we use in the next section, is to start from the optimal solution of the network with one fewer nodes. This procedure is shown to have fast convergence, and is validated, in the large W regime, to converge to the optimal value obtained by asymptotic analysis.

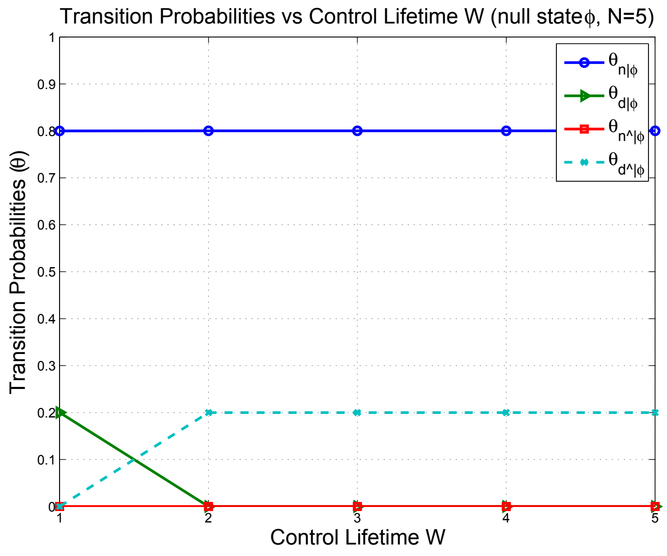
3.2 Results and Discussion

Our goal in this section is to analyze the scaling of both performance of the optimal solutions produced for throughput maximization as well as the computational complexity of those solutions, as a function of the number of nodes N and the control lifetime W.

Fig. 3.4(a) displays the optimal transition probabilities computed via our methodology as a function of N, for $W = 5$, for the null knowledge state. We see that the probability that a node listens in a slot is $\theta_{n|\phi} = 1 - 1/N$; that is, the optimal probability that a node is silent in this model where control information can be sent, is the same as it would be for the pure Random Access Channel. Further, we see that the probability that control information $\hat{d}$ is sent when the node is in the null knowledge state is $1/N$, and no data d is ever sent without preceding it with a control frame $\hat{d}$, for $W = 5$; that is, $\theta_{d|\phi} = 0$. What is important to notice



(a)



(b)

Figure 3.4: (a) Optimal transition probabilities versus the number of nodes N, for $W = 5$ (b) Optimal transition probabilities versus control lifetime W, for $N = 5$

here is that this optimal decision was generated by our methodology, and did not need to be hand-designed.

Fig. 3.4(b) displays the optimal transition probabilities from the null knowledge state ϕ, for $N = 5$, as a function of W. We see that the probability that a node is silent is $1 - 1/5 = 0.8$ for all W, and that a "phase transition" occurs from $W = 1$ to $W = 2$, where the optimal decision switches from a pure Random Access Channel, to one that always precedes data with $\hat{d}$. Because control information consumes 1 slot, it is not worth sending $\hat{d}$, if the length of the data that will follow it is not long enough. It is important to note that this decision was automatically generated by our methodology, and if the optimization metric changes, which results in a new switching point, the optimal solution for that case, which is different, can quickly be generated. Neither a carefully hand-designed protocol nor any mathematical analysis is needed to derive the switching point. The main effort goes into the general formulation and the symbolic generation of the optimization program, and the result is quickly re-configurable for different metrics.

Fig. 3.5 displays the optimal average throughput as a function of W. We see that throughput increases monotonically as a function of W, and the inflection point that occurs at $W = 2$ corresponds to the phase transition that occurs when the optimal solution switches from a pure Random Access Channel solution to one

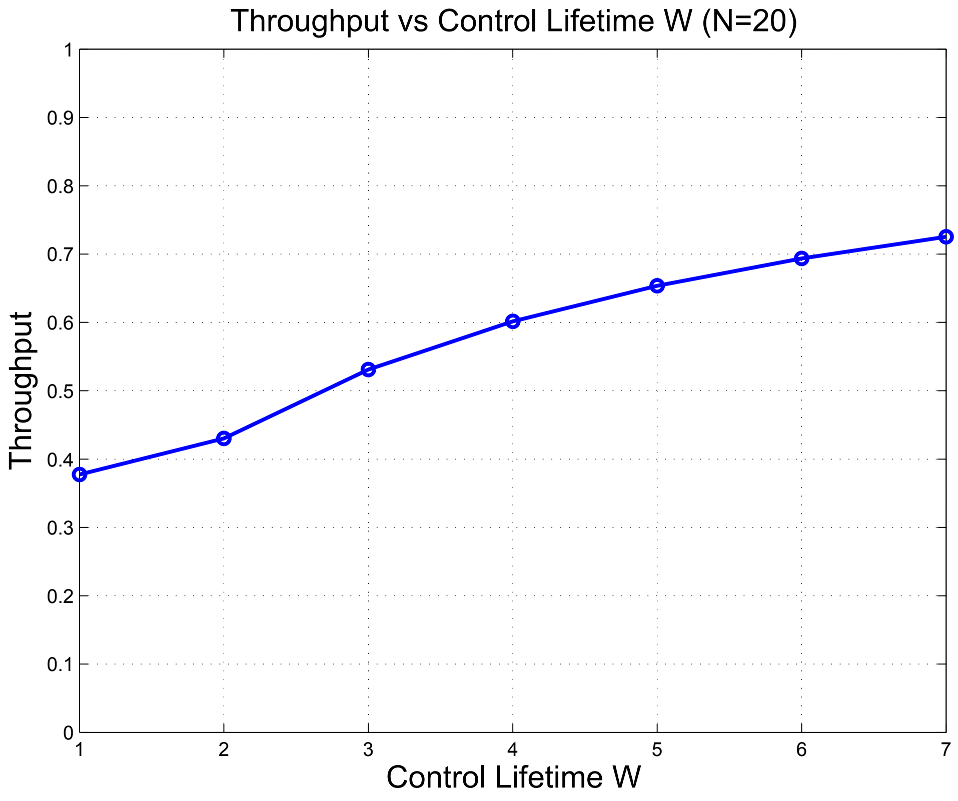
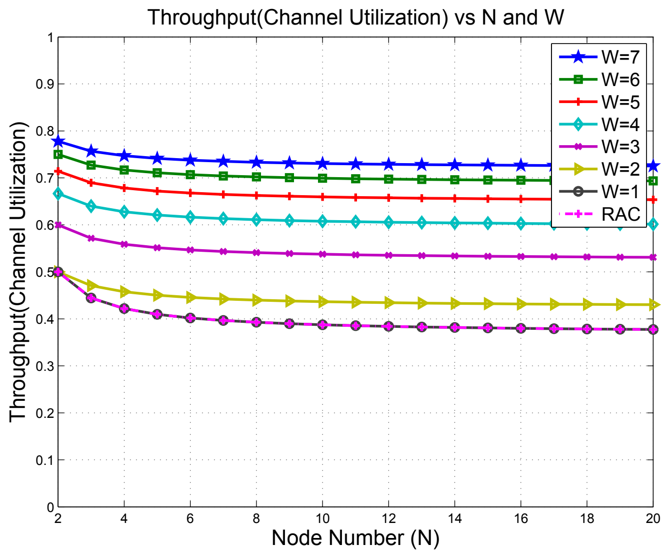


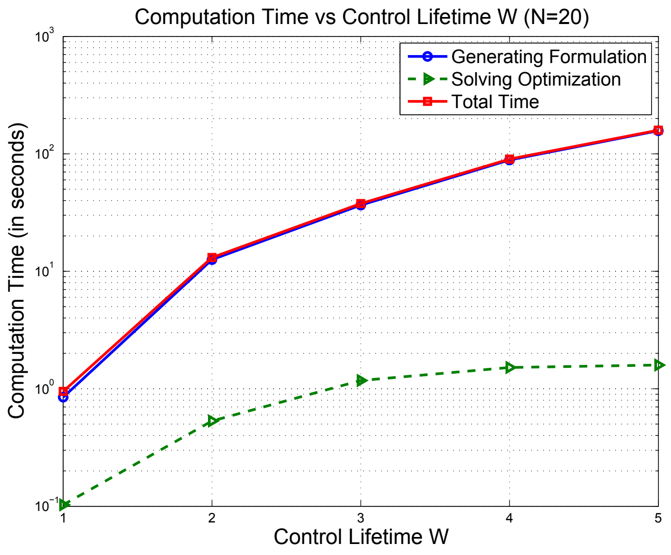
Figure 3.5: Network throughput versus the control lifetime W, for $N = 20$

that always utilizes control. The monotonic increase as a function of W is due to the fact that the successful transmission of $\hat{d}$, when it occurs, allows the other nodes to know to keep quiet, which continues to increase the average throughput as W increases, albeit with diminishing returns. We shall analytically show below, that $\lim_{W\to\infty} \bar{F} \equiv E[F]/E[T] = 1$; that is, this graph converges to 1; however, the rate at which it converges $\partial\bar{F}/\partial W$ goes as $1/W$, which is very slow. Practically, we would decide on a W, e.g. 8, beyond which the returns are not worth the increase in complexity.

Fig. 3.6(a) displays the optimal average throughput as a function of the number of nodes N, parameterized by W. We see that the optimal average throughput decreases with N, albeit flattening out quickly, and for each N, monotonically increases with W. We can validate these results for the large W regime by the following analysis: Let K denote the number of slots until the first successful transmission of $\hat{d}$ in the network. As W becomes large, the node that sent the $\hat{d}$ will send data for a very long time (W); hence, the average throughput will be approximately $\bar{F} \approx \frac{W}{W+\mathrm{E}[K]}$. Because the probability that $\hat{d}$ is transmitted successfully is $p_N \equiv (1-1/N)^{N-1}$, and K has a geometric distribution with this success probability, $\bar{F} \approx \frac{W}{W+\frac{1}{(1-1/N)^{N-1}}}$ for large W. Hence, $\lim_{N\to\infty} \bar{F} \approx \frac{W}{W+e}$, which concurs perfectly with all of the obtained results that appear for $N = 20$ in Fig. 3.6(a). From the same expression, it follows that $\lim_{W\to\infty} \bar{F} \approx 1$, for every



(a)



(b)

Figure 3.6: (a) Network throughput as a function of the number of nodes, parameterized by the control lifetime W. (b) Computation time versus the control lifetime W, for $N = 20$

N, which shows the asymptotic limit in Fig. 3.5 is 1. Calculating the convergence rate $\partial\bar{F}/\partial\bar{W}$ as a function of p_N gives a maximum of $1/4W$, which shows the rate at which it approaches the asymptote of 1. It is important to note that this mathematical analysis is not necessary for our methodology to work, and only serves as a cross-validation for our results.

We have collected the measurements of the computational complexity by measuring the execution time of the entire procedure in MATLAB, when the process was run on a Dell Studio 540 Mini-Tower, with Intel Core 2 Quad Processor Q9550 (2.83GHz, 1333MHz FSB) and 12MB cache.

Fig. 3.6(b) displays the plots for both the time needed for the symbolic generation of the optimization program, and the process of solving the optimization program, using the tic-toc feature of MATLAB, when the program was run on a Dell Studio 540, in the absence of any other active computationally intensive processes. The plot shows that the generation of the symbolic expression of the optimization program dominates the total computation time. (This will be reduced by performing the symbolic generation in a native language such as C, rather than in an interpreter such as MATLAB.) For this case study, because this is a broadcast channel, the number of states in the reduced global state space remains the same as N grows; hence, the complexity of symbolic generation is roughly constant in N. If control information were unicast, the computational complexity would grow with N.

3.3 Summary

In this chapter, we developed a methodology for optimal MAC protocol generation for a single neighborhood. The methodology that we have developed in this chapter can be applied to any choice of an objective function defined on the global state space, and the same symbolic generator can be invoked to produce the program. For example, if the objective is to minimize the average energy consumption subject to a minimum throughput constraint for each node, then adding one line of code in the constraints, and changing a single line in the recursive function, quickly produces the new optimal protocol. This is in sharp contrast with hand-designed protocols that would be designed differently for each metric. We shall take up these issues in the next chapter.

Chapter 4

Optimal MAC Protocol Design with Multiple Neighborhoods Based on Symbolic Monte Carlo Simulation

In the previous chapter, we addressed the scenario of a single wireless neighborhood with N nodes. We found that a reduced global state representation was possible to represent the knowledge state of the nodes in the network. In contrast, in this chapter, we address multiple neighborhoods, where such a reduced global state representation is no longer possible, because each node experiences a channel that is not necessarily symmetric with the other node's channels. Because some links no longer exist (as compared with the single neighborhood topology), the probabilities of nodes' actions depend on where they are located in the multi-neighborhood topology. We also need to consider only small multi-neighborhood topologies because MAC protocols are local. The small topologies that we consider in this chapter will become the local topologies of the mobile network in the next chapter.

4.1 A General Model for the Multiple Neighborhood MAC

In this chapter, we focus on the case of multiple neighborhoods, i.e., each node can only send to and/or receive from a *subset* of the $N-1$ other nodes. This *subset* is called that node's "neighborhood". We first construct a general model for a multi-neighborhood MAC as follows: We assume that there are N identical nodes in the network, all of which are slot-synchronized. We assume that each node always has data to send to its "neighbor nodes", namely, the nodes which could hear that node in the absence of any other transmissions. We utilize a collision-based model; an interference-based model is not part of this chapter. We assume that each node has no knowledge about any other node unless it obtains control information through a successfully received control frame.

We assume that each node takes any one of four basic actions in every slot: (1) n, which denotes the action that the node remains silent, (2) d, which denotes that it sends a data frame, (3) a, which denotes that it sends an acknowledgment, (4) c, which denotes that it sends a control frame that is *not* an acknowledgment.

Now, a can be further categorized into two types: a_c (an acknowledgment for a received *control* frame) and a_d (acknowledgment for a received *data* frame). Once the node receives a_c, it will grab the channel if a_c is destined to it, and it will keep silent if a_c is not destined to it.

The control frame c can be broken down into four types: 1) c_0, which stands for the case where the node will send data d for the next W slots, where W is the so-called "control lifetime", namely, how long the effect of that control information will last. Neither c_0 nor d needs to be acknowledged; the node that receives c_0 will listen to the channel and keep quiet during the W slots of the transmission of the data frame d. 2) c_1, which is similar to c_0 but requires that the node, which is supposed to receive d in the next W slots, sends back a_c in order to acknowledge the successfully received c_1; 3) c_2, which is similar to c_0 but requires that the node, that has received d (only one node), sends back the acknowledgment a_d to the sender; 4) c_3, which requires that both a_c and a_d be sent back. Note that we use broadcast transmission for control frames but unicast transmission for data frames, that is, the control frames (including acknowledgments) sent by a node can be heard by all the neighbors of that node, but the data frames sent by that node are heard but not decoded (i.e., discarded) once the receiving node realizes from the header that it is not a frame destined for itself. To summarize, the actions that

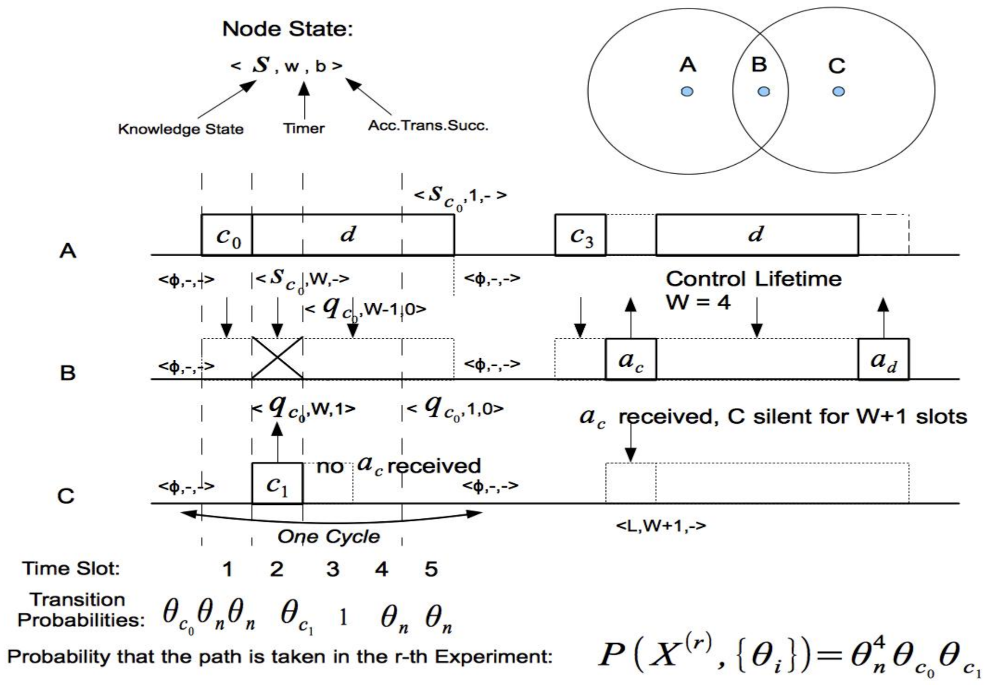

Node State:
< s , w , b >
Knowledge State
Timer
Acc.Trans.Succ.
A
B
C
Control Lifetime W = 4
a_c received, C silent for W+1 slots
no a_c received
One Cycle
<L,W+1,->
Time Slot: 1 2 3 4 5
Transition Probabilities: $\theta_{c_0}\theta_n\theta_n$ θ_{c_1} 1 θ_n θ_n
Probability that the path is taken in the r-th Experiment: $P(X^{(r)}, \{\theta_i\}) = \theta_n^4\theta_{c_0}\theta_{c_1}$

Figure 4.1: A MAC example with two neighborhoods

a node can take form the set of actions denoted by $\mathcal{A} = \{n, d, c_0, c_1, c_2, c_3, a_c, a_d\}$. We define $\mathcal{A}_I \equiv \{n, d, c_0, c_1, c_2, c_3\}$, which is a subset of $\mathcal{A}$.

We define the "state" of the system given above as a triple $\langle s, w, b\rangle$, where $s \in \mathcal{S}$ is the "knowledge state", and includes the control information that is "owned" by a node, namely, the control information frames it has sent and the control information frames that it has received in the last W slots. (Recall that W is the control information lifetime, namely, how long the effects of each control frame was designed to last.) The set of knowledge states of a node (recall that all nodes are identical) is denoted by $\mathcal{S} = \{\phi, l, s_{c_0}, s_{c_1}, s_{c_2}, s_{c_3}, q_{c_0}, q_{c_1}, q_{c_2}, q_{c_3}\}$, where ϕ is the null state, which means that no control information has been sent or received by this node during the last W slots; s_{c_i} and q_{c_i} mean that control frame c_i has just been sent or received (in the last slot), with $i \in \{0, 1, 2, 3\}$ indicating the 4 different types of control frames, as explained in the previous paragraph; l means that the node has received a control frame c_i which was not destined to it, and thus this node will keep silent during the transmission of other nodes. (We shall refer to l by the name "cooperative silence", which means that the node keeps silent to make way for other nodes' transmissions.) The second element of the triple is $w \in \mathcal{N}$, a timer that is associated with each node, and restarts every time that a new control frame is either sent or received, and counts down by one to zero at each discrete time interval. The third element of the triple is $b \in \{0, 1\}$, a binary variable that indicates whether the data reception is successful: b is used only after a control frame has been received (c_0, c_1, c_2 or c_3). After this reception, the node knows that it will receive d in the following W slots, and it sets $b = 1$. If there is a slot when no data frame d is received, b is multiplied by zero. At the end of the W slots, by checking b, the receiving node will know whether there is a missing data frame during the last W slots: if $b = 1$, which means that all the data frames have been received, an acknowledgment will be sent back; if $b = 0$, some data frames are missing, in which case no acknowledgment is sent back.

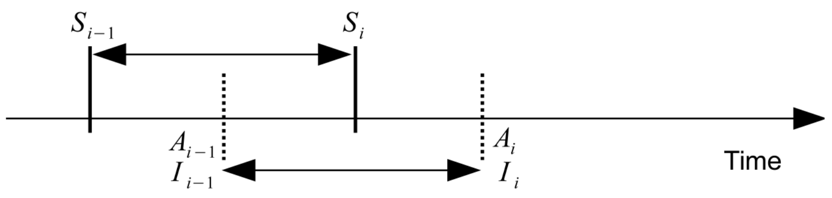


Figure 4.2: Timing of FSM with a 2-phase design

During the transition between states, any of these elements in the triplet can be marked as "−", which stands for "don't care", meaning that this element can be anything. For example, $\langle s_{c0}, W, - \rangle$ means that the sender does not care about the third element, b, in the triplet. Finally, we note that upon initialization, the initial knowledge state of each node is ϕ.

Now, we illustrate the above by a simple example in Fig. 4.1: Three nodes A, B and C are located as shown in the figure, such that B can communicate with both A and C, but A and C can only communicate with B, i.e., two neighborhoods A-B and B-C are formed. (Note that the original definition of 'neighborhood' in this chapter allows for asymmetric links; hence, this is a special case where all links are bidirectional.) Before the system starts, all the nodes are in state $\langle \phi, -, - \rangle$, where "−" means "don't care". In this example in Fig. 1, at the beginning, node

A chooses to send control frame c_0 to its neighbor B before it sends data d to node B (d is a unicast transmission), and as a result, A transitions from $\langle \phi, -, - \rangle$ to $\langle s_{c_0}, W, - \rangle$. The timer is started and set to W, which is the control lifetime. On the other side, only node B can hear c_0 and will listen to A for the next W slots; thus, B transitions to $\langle q_{c_0}, W, 1 \rangle$ while C keeps its original state. b is initialized to 1 at the receiver when the receiving process starts. At the same time, node C cannot hear c_0 sent by A (and stays in the initial state); thus, it still has its choice to access the channel, and here, sends a control frame c_2 to the channel, which leads to a collision at node B in the second slot (we assume that the whole sequence gets corrupted when a collision happens at any slot during the sequence), and node B changes its b to 0 to indicate that a data frame has been corrupted. In the next slot, no acknowledgment a_c is sent back by B because of the collision and node C stops the process of sending d and returns to its initial state. After node A finishes sending d, it returns to its initial state $\langle \phi, -, - \rangle$. After two slots, node A again wants to take hold of the channel to continue to send data, but this time it chooses to send control frame c_3 (instead of c_0 used in the first part), which requires that both control and data be acknowledged (a_c and a_d). Node B replies to node A with a_c which is also received by node C; thus node C receives a control frame which is not destined to itself and transitions to the "passive silent state", $\langle l, W+1, -\rangle$, and keeps silent for the next $W+1$ slots (W slots for d, and 1 additional slot for a_d).

The complete transition table for the finite state machine (FSM) is displayed in Table 4.1. This transition table in the figure fully characterizes the state space of a node with a 2-phase design. The timing of this design appears in Fig. 4.2. The output and the state computations are staggered as shown. In the first phase of the clock cycle, the next state S_i is computed based on the current state S_{i-1} and the current inputs A_{i-1} and I_{i-1}, where A_{i-1} denotes the action of this node at (discrete) time $i-1$, and I_{i-1} denotes the input to this node at time $i-1$. In the second phase of the clock cycle, the output A_i is computed based on only S_i, which becomes available at the end of the first phase of the clock cycle. This table encodes a general template which can be applied to scenarios that involve the design of MAC protocols for multiple neighborhoods. Hence, even though the table might appear complex, it needs to be derived only once for a very general class. Second, note that the output action A shown in the rightmost column is a random variable. (Note that many of the rows also contain deterministic actions such as n, d, and a_1.) This random variable A, has the sample space A_I, which is the set of initial node actions, $\mathcal{A}_I = \{n, d, c_0, c_1, c_2, c_3\}$. Third, $\langle \phi \vee l, -, -\rangle$ and $\langle q_{c_1}, w > 1, -\rangle$ are abbreviations for "$\langle \phi, -, -\rangle$ or $\langle l, -, -\rangle$" and "$\langle q_{c_1}, w, -\rangle$ with $w > 1$".

State Transition Rules				
Phase 1				
			Phase 2	
Input		Current	Next	Output
receive	send	State	State	Action
–	d	$< \phi, -, - >$	$< \phi, -, - >$	A
d	n	$< \phi, -, - >$	$< \phi, -, - >$	A
n	n	$< \phi, -, - >$	$< \phi, -, - >$	A
$c_0 \wedge$ dest.	n	$< \phi \vee l, -, - >$	$< q_{c_0}, W-1, - >$	n
–	–	$< q_{c_0}, w > 1, - >$	$< q_{c_0}, w-1, - >$	n
–	–	$< q_{c_0}, 1, - >$	$< \phi, -, - >$	A
$c_0 \wedge$ ndest.	n	$< \phi, -, - >$	$< l, W-1, - >$	n
$n \vee d \vee X$	n	$< l, w > 1, - >$	$< l, w-1, - >$	n
$n \vee d \vee X$	n	$< l, 1, - >$	$< \phi, -, - >$	A
$c_0 \wedge$ ndest.	n	$< l, -, - >$	$< l, W-1, - >$	n
$c_1 \wedge$ ndest.	n	$< l, -, - >$	$< l, W, - >$	n
$c_2 \wedge$ ndest.	n	$< l, -, - >$	$< l, W, - >$	n
$c_3 \wedge$ ndest.	n	$< l, -, - >$	$< l, W+1, - >$	n
$c_1 \wedge$ dest.	n	$< \phi \vee l, -, - >$	$< q_{c_1}, W, - >$	a_c
–	–	$< q_{c_1}, w > 1, - >$	$< q_{c_1}, w-1, - >$	n
–	–	$< q_{c_1}, 1, - >$	$< \phi, -, - >$	n
$c_2 \wedge$ dest.	n	$< \phi \vee l, -, - >$	$< q_{c_2}, W, - >$	n
d	–	$< q_{c_2}, w > 1, 1 >$	$< q_{c_2}, w-1, 1 >$	n
d	–	$< q_{c_2}, 1, 1 >$	$< \phi, -, - >$	a_d
$c_3 \wedge$ dest.	n	$< \phi \vee l, -, - >$	$< q_{c_3}, W+1, - >$	a_c
d	–	$< q_{c_3}, w > 1, 1 >$	$< q_{c_3}, w-1, 1 >$	n
d	–	$< q_{c_3}, 1, 1 >$	$< \phi, -, - >$	a_d
$X \vee n$	–	$< q_{c_2}, w > 1, - >$	$< q_{c_2}, w-1, 0 >$	n
–	–	$< q_{c_2}, w > 1, 0 >$	$< q_{c_2}, w-1, 0 >$	n
–	–	$< q_{c_2}, 1, 0 >$	$< \phi, -, - >$	n
$X \vee n$	–	$< q_{c_2}, 1, - >$	$< \phi, -, - >$	n
$X \vee n$	–	$< q_{c_3}, w > 1, - >$	$< q_{c_3}, w-1, 0 >$	n
–	–	$< q_{c_3}, w > 1, 0 >$	$< q_{c_3}, w-1, 0 >$	n
–	–	$< q_{c_3}, 1, 0 >$	$< \phi, -, - >$	n
$X \vee n$	–	$< q_{c_3}, 1, - >$	$< \phi, -, - >$	n
–	c_0	$< \phi, -, - >$	$< s_{c_0}, W-1, - >$	d
–	–	$< s_{c_0}, w > 1, - >$	$< s_{c_0}, w-1, - >$	d
–	–	$< s_{c_0}, 1, - >$	$< \phi, -, - >$	n
–	c_1	$< \phi, -, - >$	$< s_{c_1}, W, - >$	n
a_c	–	$< s_{c_1}, W, - >$	$< s_{c_1}, W-1, - >$	d
no a_c	–	$< s_{c_1}, W, - >$	$< \phi, -, - >$	A
–	–	$< s_{c_1}, 1 < w < W, - >$	$< s_{c_1}, w-1, - >$	d
–	–	$< s_{c_1}, 1, - >$	$< \phi, -, - >$	A
–	c_2	$< \phi, -, - >$	$< s_{c_2}, W, - >$	d
–	–	$< s_{c_2}, w > 2, - >$	$< s_{c_2}, w-1, - >$	d
–	–	$< s_{c_2}, w = 2, - >$	$< s_{c_2}, 1, - >$	n
–	–	$< s_{c_2}, 1, - >$	$< \phi, -, - >$	A
–	c_3	$< \phi, -, - >$	$< s_{c_3}, W+1, - >$	n
a_c	–	$< s_{c_3}, W+1, - >$	$< s_{c_3}, W, - >$	d
no a_c	–	$< s_{c_3}, W+1, - >$	$< \phi, -, - >$	A
–	–	$< s_{c_3}, 2 < w < W+1, - >$	$< s_{c_3}, w-1, - >$	d
–	–	$< s_{c_3}, 2, - >$	$< s_{c_3}, 1, - >$	n
–	–	$< s_{c_3}, 1, - >$	$< \phi, -, - >$	A

$\vee$ and $\wedge$ denote OR and AND;
A is a random variable and $A \in \mathcal{A}_I = \{n, d, c_0, c_1, c_2, c_3\}$;
"dest."\"ndest." means this "is"\"is not" the destined node;
X stands for "Collision" at the receiver;
"–" stands for "don't care" when making transition decisions.

Table 4.1: General transition table of a node's finite state machine

Now that we have the transition table, we introduce the design variables, $\theta_A \in [0, 1]$ with $A \in \mathcal{A}_I$, that is, $\theta_A \in \{\theta_n, \theta_d, \theta_{c_0}, \theta_{c_1}, \theta_{c_2}, \theta_{c_3}\}$, which is the probability of taking a particular action from the set of initial actions, $\mathcal{A}_I$, based on the initial state $\langle \phi, -, - \rangle$. This is the probability mass function of the discrete random variable A over the sample space $\mathcal{A}_I$, where $\sum \theta_A = 1$. The probability of taking each action is an optimization variable.

4.2 Approximation of the Optimization Program via Symbolic Monte Carlo Simulation

The goal of this section is two-fold: Our first aim is to introduce the *Symbolic Monte Carlo* approach to sample the global state space and collect *symbolic* terms (hence, sampling symbolic expressions). This is different from the traditional Monte Carlo simulation which accumulates *numbers*; here, in contrast, we accumulate *expressions*. Second, our goal is to obtain, using this method, an approximate but much shorter expression of the objective function of the optimization (which optimizes for any of the desired metrics, such as throughput, energy, or any weighted combination), and find the optimal values of the design variables. As shown in [46], the design problem we consider can always be described as a maximization of average reward, which is equal to $\mathrm{E}[F]/\mathrm{E}[T]$, i.e., the expected reward per cycle divided by the expected cycle length, where a cycle is defined as the (discrete) time between two subsequent visits to the same recurrent state (the initial state in our case). $\mathrm{E}[F]$ and $\mathrm{E}[T]$ are functions of the design variables $\theta_A, A \in \mathcal{A}_I$. A sufficient condition for the recurrence of a Markov chain is imposed on the design variables: $\theta_i \geq \epsilon$ for all θ_i, where ϵ is taken to be a small positive number. The objective function can be written as: $\lim_{\epsilon \to 0} E[F]/E[T]$ such that $\theta_A \in [\epsilon, 1]$ and $\sum \theta_A = 1$.

In order to construct the symbolic expressions for $\mathrm{E}[F]$ and $\mathrm{E}[T]$, we need to explore the global state space to collect the expression for each path that has been traversed. For the case with multiple neighborhoods and acknowledgments, which brings in much more complexity into the model compared with the single neighborhood case, an exhaustive exploration of the global state space is not viable. However, using the fact that many symmetries exist in the global state space due to the identicality of the nodes and due to the fact that the global transition table of these finite state machines (with random variable outputs) that communicate with each other is an ergodic process, the global state space can be captured by sampling this space in a manner we shall explain. Due to the ergodicity, we are guaranteed that the symbolic expression that is accumulated by this random sampling and averaging will converge eventually to the correct expression of the objective function. We call this approach "symbolic Monte

Carlo", and obtain an approximate expression for E$[F]$ and E$[T]$ by using this method.

To this end, we define the random variable $X^{(r)}$ as the path traversed in one cycle in the rth experiment, that starts from the initial state and returns to the initial state. The $f(X^{(r)})$ is the reward collected along the path $X^{(r)}$, which can be any metric such as the amount of data successfully transferred. Define $P(X^{(r)}, \{\theta_i\}) = \prod_{i \in \mathcal{A}_I} \{\theta_i^{N_i(X^{(r)})}\}$ as a probability generating function, which generates the probability of taking the path $X^{(r)}$, given the set of decision variables $\{\theta_i, i \in \mathcal{A}_I\}$. The exponent $N_i(X^{(r)})$ means that by taking path $X^{(r)}$, the node has chosen the node action $i \in \mathcal{A}_I$ exactly $N_i(X^{(r)})$ times. Before symbolic Monte Carlo is started, the initial decision variables, $\theta_{\text{init},i}$, are chosen. If the designer has any intuition as to what might be a good starting point for the symbolic Monte Carlo simulation, this can be incorporated into this choice at this stage. We define a "sampler" as an experiment that randomly chooses a path in the global state space; hence, each realization of $X^{(r)}$, i.e., $x^{(r)}$, is an outcome of the sampler. Based on this, we introduce the "sampling density" denoted by $Q(X^{(r)}) = P(X^{(r)}, \{\theta_{\text{expl},i}\})$, which denotes the probability that the path $X^{(r)}$ is taken by the sampler in an experiment, where the $\theta_{\text{expl},i}$ are the design variables used by sampler in the exploration. In order to obtain a good cover of the whole design space, we use a uniform sampler, which sets $\theta_{\text{expl},i} = 1/N$. Note that $\{\theta_i\}$

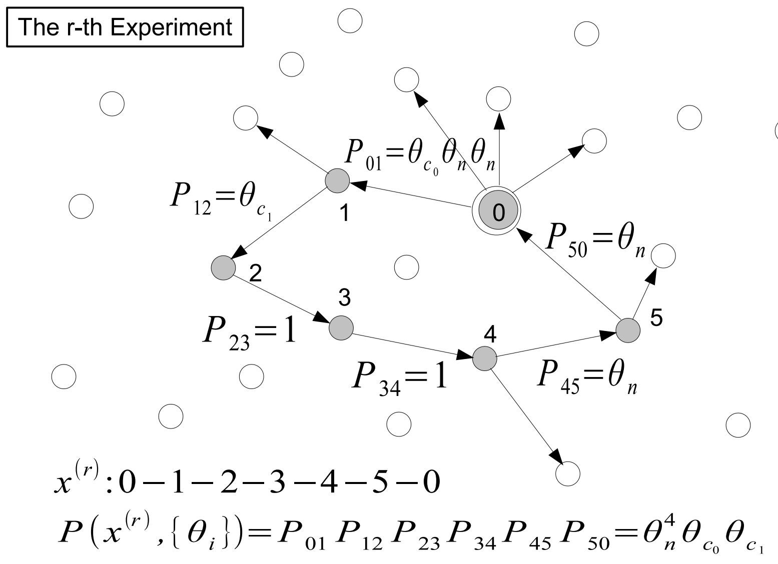

The r-th Experiment
$P_{01}=\theta_{c_0}\theta_n\theta_n$
$P_{12}=\theta_{c_1}$
$P_{23}=1$
$P_{34}=1$
$P_{45}=\theta_n$
$P_{50}=\theta_n$
0
1
2
3
4
5
$x^{(r)}:0-1-2-3-4-5-0$
$P(x^{(r)},\{\theta_i\})=P_{01}P_{12}P_{23}P_{34}P_{45}P_{50}=\theta_n^4\theta_{c_0}\theta_{c_1}$

Figure 4.3: Illustration of Symbolic Monte Carlo

are kept symbolic in the expression and later used as optimization variables. Finally, the approximated optimization expression, $\hat{F}$, of the average reward per time slot E[F]/E[T] can be obtained as follows,

$$\hat{F} = \frac{\sum_r P(X^{(r)}, \{\theta_i\}) f(X^{(r)})}{\sum_r P(X^{(r)}, \{\theta_i\}) t(X^{(r)})} \tag{4.1}$$

where $f(X^{(r)})$ is the reward associated with the path $X^{(r)}$ and $t(X^{(r)})$ as the discrete time length of the path $X^{(r)}$.

For the example shown in Fig. 4.1, if the sequence of states between time slots 0 and 6 (that is, a cycle) is the r-th experiment of the symbolic Monte Carlo simulation, then the outcome of the experiment, $x^{(r)}$, is the path shown in Fig. 4.3, namely "$0 - 1 - 2 - 3 - 4 - 5 - 0$". The symbolic term accumulated as a result of this sampling in this experiment, namely $P(x^{(r)}, \{\theta_i\})$, is $\theta_n^4 \theta_{c_0} \theta_{c_1}$.

After the objective function has been obtained by Symbolic Monte Carlo, the optimization is performed by using the *OpenOpt* package provided in Python.

4.3 Optimal MAC Protocol Design with Multi-objective Optimization

Wireless networks have to trade off a variety of factors, which together determine whether a protocol can be claimed to be optimal. To this end, we now consider *multi-objective optimization* where the objective function is a weighted sum of multiple objective functions. Because different objectives usually have different ranges, they need to be normalized before they are weighted. Thus, the general form of a multiple objective function is:

$$\max_{\bar{\theta}} \sum_j \lambda_j F_j(\bar{\theta}) \tag{4.2}$$

where $\bar{\theta}$ is a vector of design variables, F_j is the jth normalized design objective, and λ_j is the weight of the jth objective. Here, we assume that $F_j \geq 0$ for all j.

In this chapter, we choose two of the most important metrics for wireless network design: throughput and energy efficiency. In particular, we introduce the following two metrics: (1) the *per node throughput* R, which is the average number of frames successfully delivered between two neighboring nodes in unit time and (2) the *average per node power consumption* P, which is the average energy consumed by a node in one time slot. After normalization, we obtain the *normalized throughput* as

$$R_{\text{norm}} = \frac{R - R_{\min}}{R_{\max} - R_{\min}} \tag{4.3}$$

and the *normalized power efficiency* as

$$P_{\text{norm}} = \frac{P_{\max} - P}{P_{\max} - P_{\min}}. \tag{4.4}$$

Thus the throughput-energy optimization can be written as

$$\max_{\theta_i} \lambda R_{\text{norm}}(\theta_i) + (1-\lambda) P_{\text{norm}}(\theta_i) \tag{4.5}$$

$$\text{s.t.} \sum_i \theta_i = 1 \tag{4.6}$$

$$1 \geq \theta_i \geq 0 \tag{4.7}$$

where $R_{\text{norm}}(\theta_i)$ and $P_{\text{norm}}(\theta_i)$ are functions of the design variables $\{\theta_i\}$, and λ is the weight of the throughput objective. We assume that $1 \geq \lambda \geq 0$; thus, we have $1 - \lambda$ as the weight of energy efficiency. The simulation results for this multi-objective function will be presented at the end of the next section.

4.4 Simulation Results and Discussion

In this section, we discuss our results from three different angles: (1) optimality, where we take throughput as our optimization metric, (2) computational complexity, which we take as the time required to run the entire design chain (which is comprised of the generation of the approximate optimization program via symbolic Monte Carlo, followed by solving it via the *OpenOpt* package provided in Python), and (3) convergence, which is a measure of the rate at which the symbolic Monte Carlo simulation (which uses random sampling) converges. This convergence is measured directly in terms of convergence to the actual optimal throughput.

In contrast with the single MAC neighborhood scenario, where the network is a fully connected graph and does not change with the nodes' real geographical locations, in a multiple neighborhood scenario, the network topology is determined by the nodes' locations (as well as other channel factors such as shadowing).

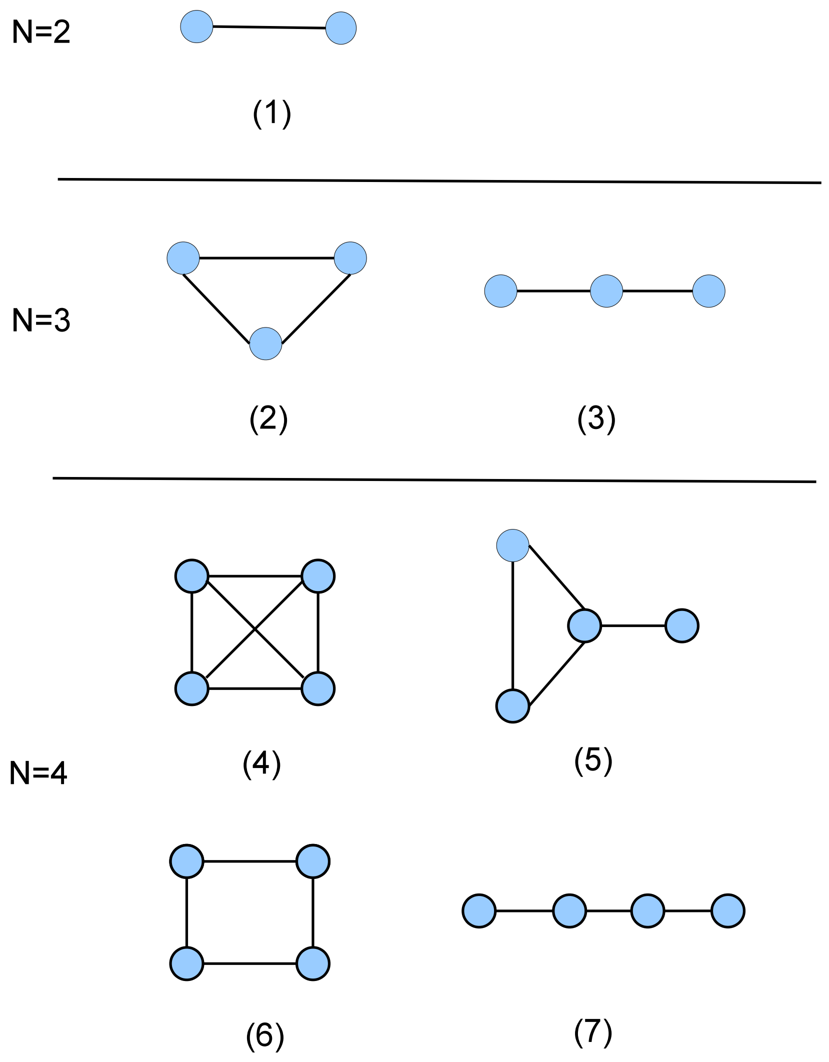


Figure 4.4: Seven different network topologies for simulation

As mentioned before, because MAC protocols are local, finding an optimal solution for small-size topologies suffices. The objective functions that are symbolically accumulated for these topologies will then be used as submodules for subgraphs of mobile networks in the next chapter.

As an illustration, we generate and discuss the optimal MAC protocols for seven sample topologies that range from $N = 2$ to $N = 4$, These topologies are shown in Fig. 4.4. (Each pair of nodes that are neighbors is connected via a solid line. We assume that a bidirectional channel exists between these shown node pairs.)

We graph the obtained optimal design variables $\{\theta_i\}$ as well as the average throughput, as a function of the control lifetime, W, for the seven simulation cases illustrated. In order to see the impact of the number of nodes, we also graph $\{\theta_i\}$ and the average throughput, as a function of the number of nodes, N, for the fully connected graph, and for control lifetime $W = 5$.

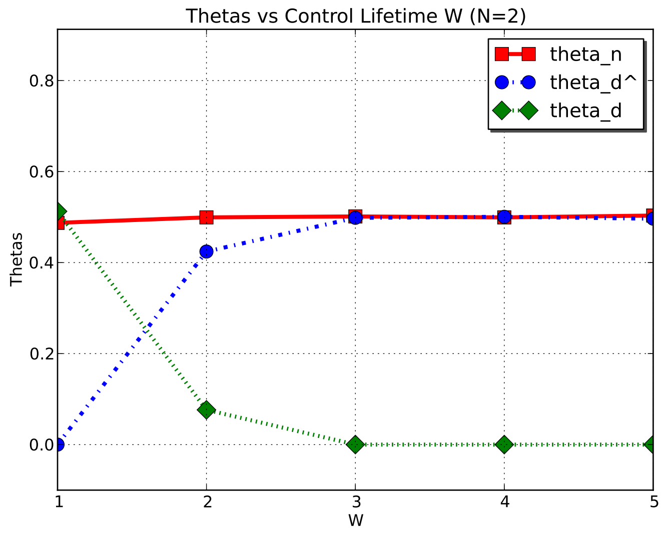
In Chapter 3, where we considered a model for the single neighborhood case, one of the results that we found was that the nodes had to keep silent for a fraction $(N-1)/N$ of the time to achieve the optimal throughput. In this chapter, we use this result to set the initial values of the design variables as follows: $\theta_n = (N-1)/N$, and $\theta_i = 1/(5N)$ for $i \neq n$. These initial values also define the sampling density (see the previous section for the definition) in the symbolic Monte Carlo simulation. In our simulations, each objective function is generated by doing a 15-step *depth-first search* (D_{sd}=15) and we run the entire optimization generation and solution chain in Python environment on a Macbook Pro computer (2.6GHz Intel Core i7 Processor, 8GB 1600MHz DDR3) with no other concurrent compute-intensive processes.

We now discuss the simulation results for the seven different topologies:

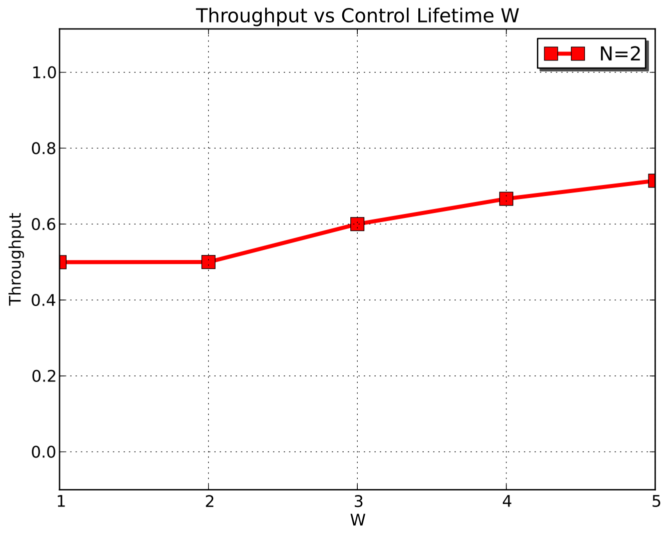
1. *Case (1) ($N = 2$):*

   In Fig. 4.5(a), the resulting optimal values of the design variables $\{\theta_i\}$ are shown. In Fig. 4.5(a), we see that almost no control frames are generated in the optimal solution when the control lifetime W is small ($\theta_d > 0$ and $\theta_{\hat{d}} \approx 0$). As the control lifetime W increases, the nodes start to send control frames before data frames (that is, $\theta_{\hat{d}}$ becomes positive). Note that this control frame $\hat{d}$ is generated automatically from a general model of the MAC Layer, rather than being "hand-designed" into the protocol; hence, it has the advantage that it can be dynamically generated based on the network conditions (and will not be generated if it is not advantageous to do so given the objective function).

   In Fig. 4.5(b), the average throughput of the generated MAC protocol is shown as a function of the control lifetime W. In Fig. 4.5(b), we see that



(a)



(b)

Figure 4.5: Case (1): Number of nodes $N = 2$ (a) Optimized design variable θ_i versus control lifetime W (b) Throughput versus control lifetime W

the throughput increases as the control lifetime W increases, which is due to the coordination facilitated by the control frame $\hat{d}$ between the nodes.

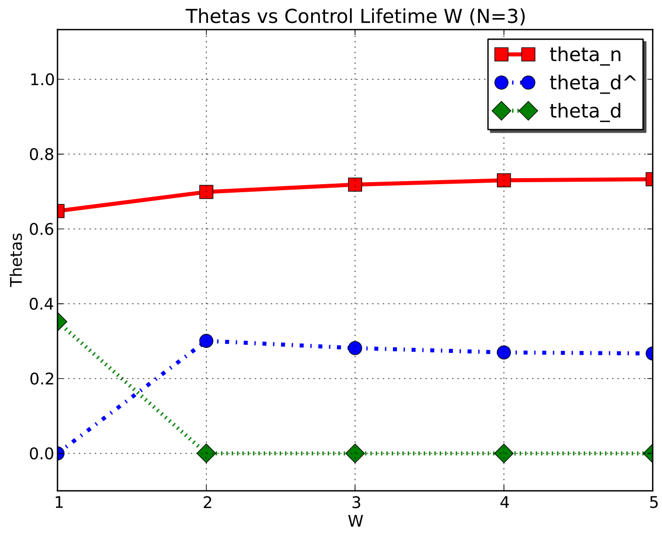
2. *Case (2) ($N = 3$, fully connected):*

   In Fig. 4.6(a), the resulting optimal values of the design variables $\{\theta_i\}$ are shown. In Fig. 4.6(a), we see a pattern that is similar to the one shown for the $N = 2$ case: as the control lifetime W increases, the nodes start to send advertisements for data frames (that is, $\theta_{\hat{d}}$ becomes positive). However, the nodes also spend more time listening to the channel, and less time accessing the channel (that is, θ_n is larger, whereas θ_d and $\theta_{\hat{d}}$ are smaller), which is due to the fact that more nodes are using the same single channel and hence it is preferred to have fewer transmissions.

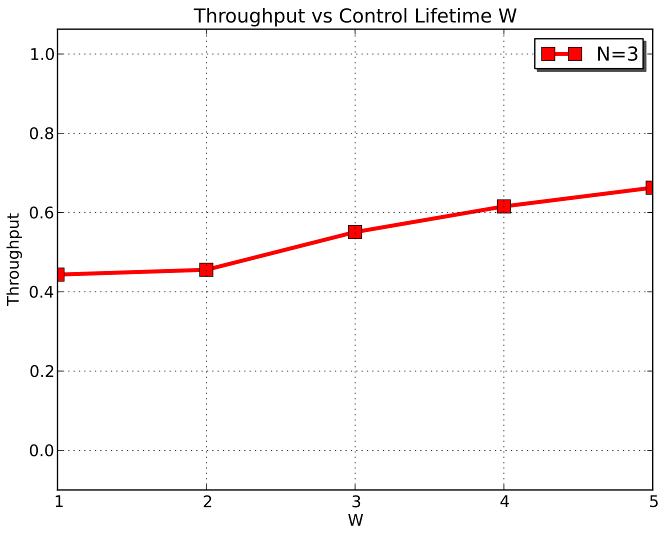
   In Fig. 4.6(b), the average throughput of the generated optimal MAC protocol is shown as a function of the control lifetime W. In Fig. 4.6(b), we see that the throughput increases as the control lifetime W increases, which is similar to the $N = 2$ case.

3. *Case (3) ($N = 3$, linear topology):*

   In Fig. 4.7(a), the resulting optimal values of the design variables $\{\theta_i\}$ are shown. In Fig. 4.7(a), we can see that no control frames are generated in the optimal solution when the control lifetime W is small ($\theta_d > 0$ and $\theta_{\hat{d}} \approx 0$).

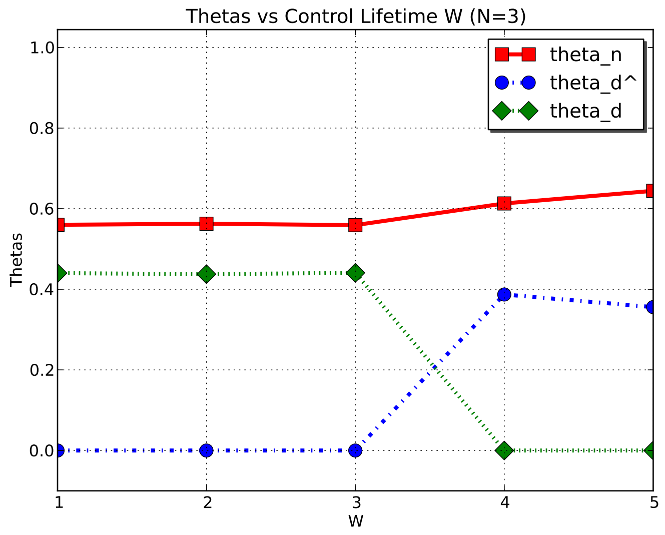


(a)

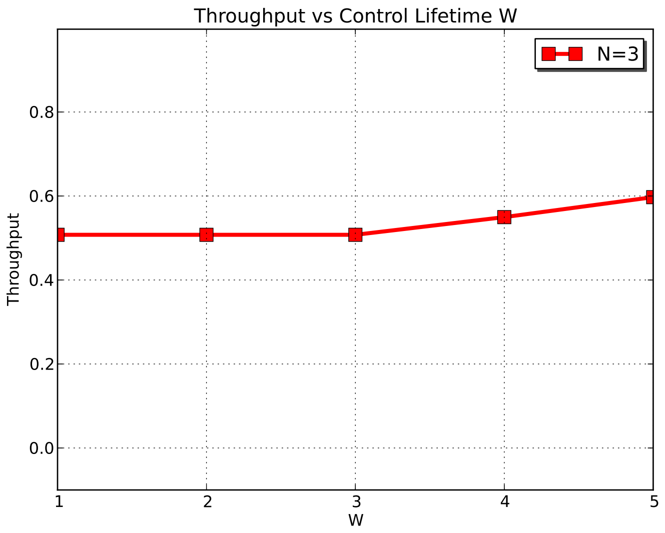


(b)

Figure 4.6: Case (2): Number of nodes $N = 3$ (fully connected) (a) Optimized design variable θ_i versus control lifetime W (b) Throughput versus control lifetime W



(a)



(b)

Figure 4.7: Case (3): Number of nodes $N = 3$ (linear topologies) (a) Optimized design variable θ_i versus control lifetime W (b) Throughput versus control lifetime W

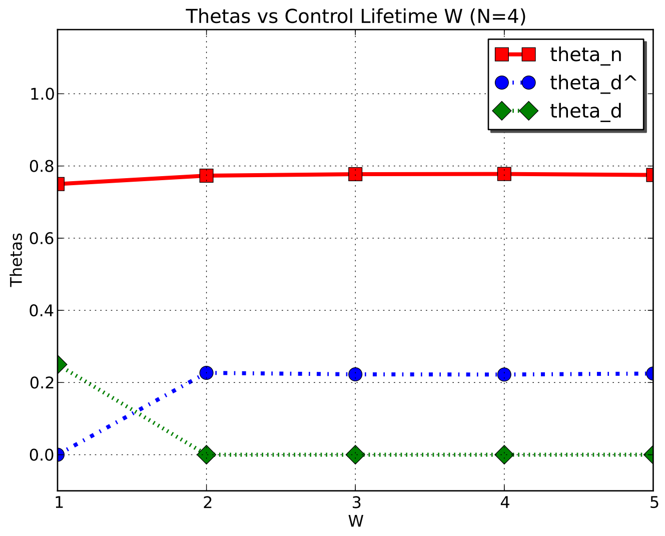
However, as the linear topology belongs to the multi-neighborhood scenario, the nodes start to send control frames (that is, $\theta_{\hat{d}}$ becomes positive) after the control lifetime becomes much larger ($W \geq 4$). This is quite different from the optimal results for the fully-connected graph case (i.e., the single neighborhood case), where the nodes start to send control frames as soon as $W > 1$.

In Fig. 4.7(b), the average throughput of the generated MAC protocol is shown as a function of the control lifetime W. In Fig. 4.7(b), we see that the throughput increases as the control lifetime W increases, which is again, due to coordination facilitated by the control frame $\hat{d}$ between the nodes. However, again, a difference from the single-neighborhood case is that this increase starts after the control lifetime becomes much larger, namely at $W \geq 4$ rather than at $W > 1$.

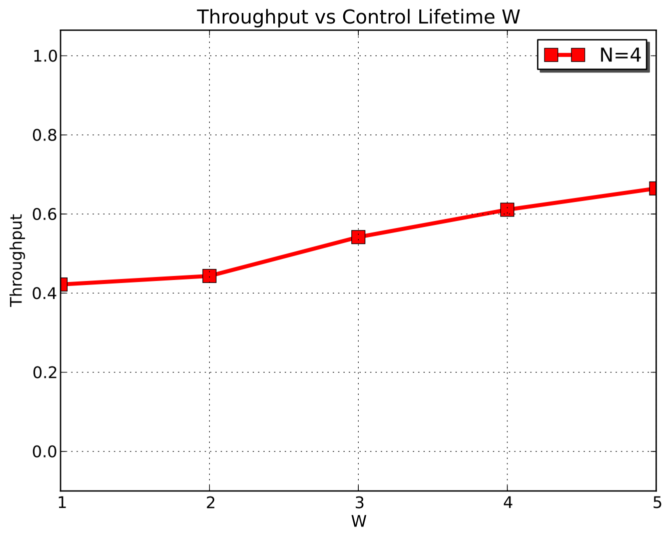
For $N = 4$, we have considered the following topologies: fully-connected graph (case 4), topology of case 5, the ring topology (case 6) and the linear topology (case7).

4. *Case (4) ($N = 4$, fully connected graph):*

In Fig. 4.8(a), the resulting optimal values of the design variables $\{\theta_i\}$ are shown. In Fig. 4.8(a), we see a pattern that is similar to the one shown for



(a)



(b)

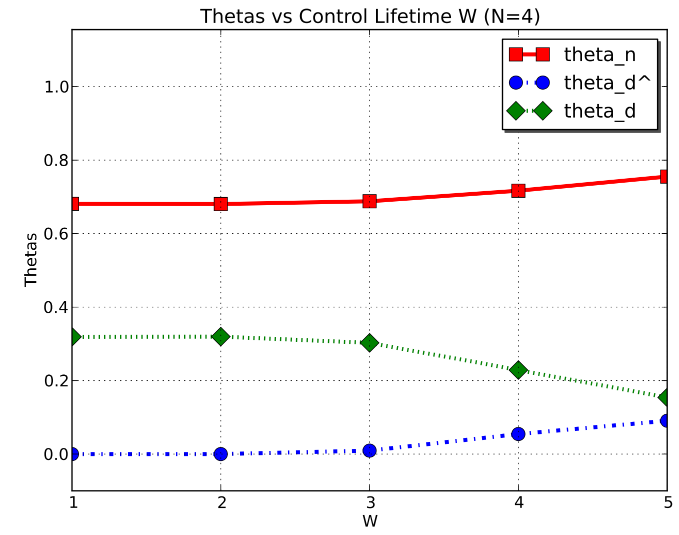
Figure 4.8: Case (4): Number of nodes $N = 4$ (fully connected) (a) Optimized design variable θ_i versus control lifetime W (b) Throughput versus control lifetime W

the $N = 2$ and $N = 3$ cases: as the control lifetime W increases, the nodes start to send data advertisements (that is, $\theta_{\hat{d}}$ becomes positive). However, unlike the cases for $N = 2$ and $N = 3$, the time that nodes spend on listening to the channel does *not* change much as the control lifetime increases. (Note that θ_n is flat as W increases.) This result coincides with the one seen in Fig. 3.4(b) in Chapter 3, which validates the results of the Symbolic Monte Carlo approach against the one reduced global state solution in Chapter 3.

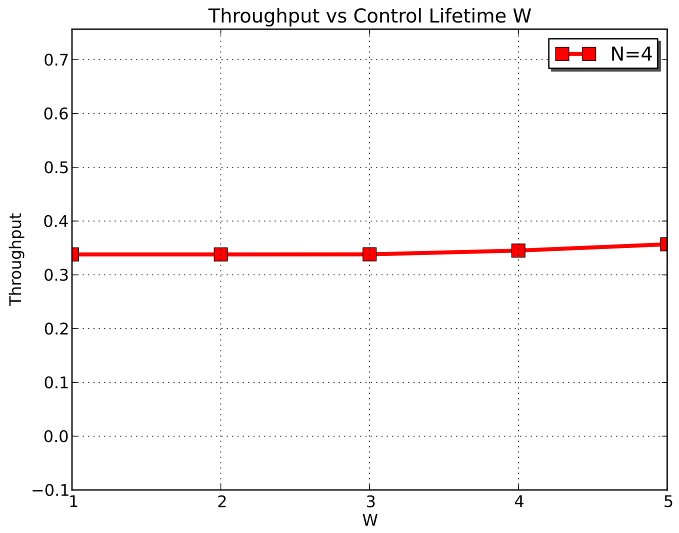
In Fig. 4.8(b), the average throughput of the generated optimal MAC protocol is shown, as a function of the control lifetime W, where we can see a result that is similar to the one for the $N = 3$ fully-connected graph case.

5. *Case (5) ($N = 4$):*

In Fig. 4.9(a), the resulting optimal values of the design variables $\{\theta_i\}$ are shown. In Fig. 4.9(a), we can see that almost no control frames are generated in the optimal solution when the control lifetime W is small ($\theta_d > 0$ and $\theta_{\hat{d}} \approx 0$). The nodes start to send control frames only after the control lifetime becomes larger, which is similar to the $N = 3$ multiple neighborhood case. However, the switch-over point where the node starts to send a control frame is at an even larger control lifetime $W > 4$, compared with the $N = 3$ case, and the change appears to be more gradual than it was for the $N = 3$ case.



(a)



(b)

Figure 4.9: Case (5): Number of nodes $N = 4$ (a) Optimized design variable θ_i versus control lifetime W ($N = 4$) (b) Throughput versus control lifetime W ($N = 4$)

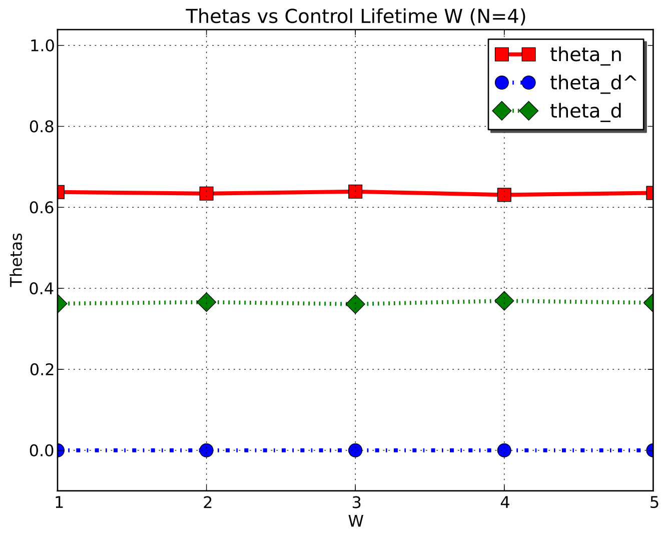
In Fig. 4.9(b), the average throughput of the generated MAC protocol is shown as a function of the control lifetime W, where we can see a result similar to the ones for the $N = 3$ cases.

6. *Case (6) (N* $= 4$*, ring topology):*

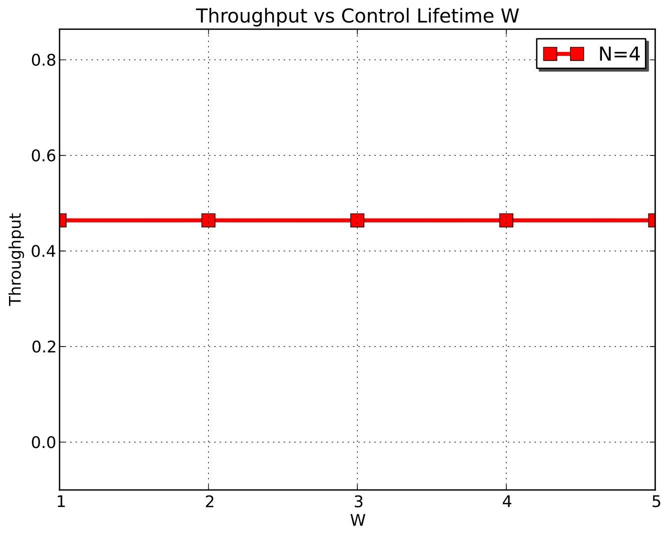
   In Fig. 4.10(a), the resulting optimal values of the design variables $\{\theta_i\}$ are shown. In Fig. 4.10(a), we can see that almost no control frames are generated in the optimal solution for any control lifetime W used in the simulation. The reason is that in a ring topology, the network is sparsely populated and fewer transmission collisions have a chance to happen; thus, sending a control frame such as $\hat{d}$ is not preferred, when one considers the overhead involved. It is quite remarkable that such complex decisions can be generated automatically in an optimal fashion, whereas it would take significant design intuition to arrive at these results by reasoning individually about these topologies.

   In Fig. 4.10(b), the average throughput of the generated optimal MAC protocol is shown as a function of the control lifetime W. In the figure, we can see that the throughput does not change with W, which is expected, considering that no control frames are sent at all.

7. *Case (7) (N* $= 4$*, linear topology) :*

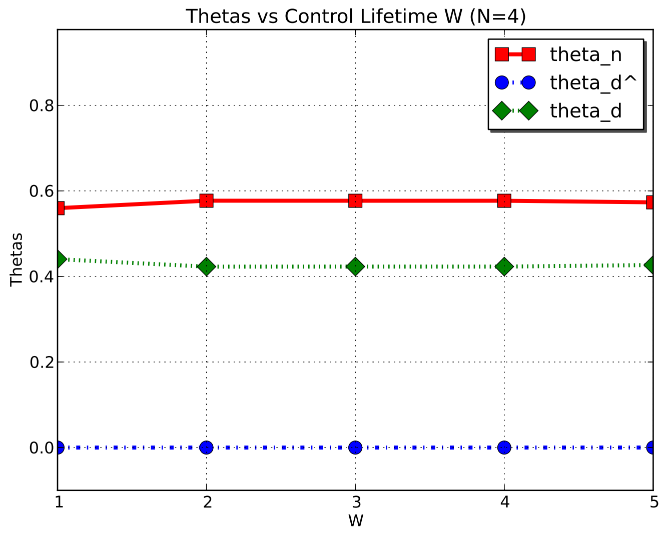


(a)

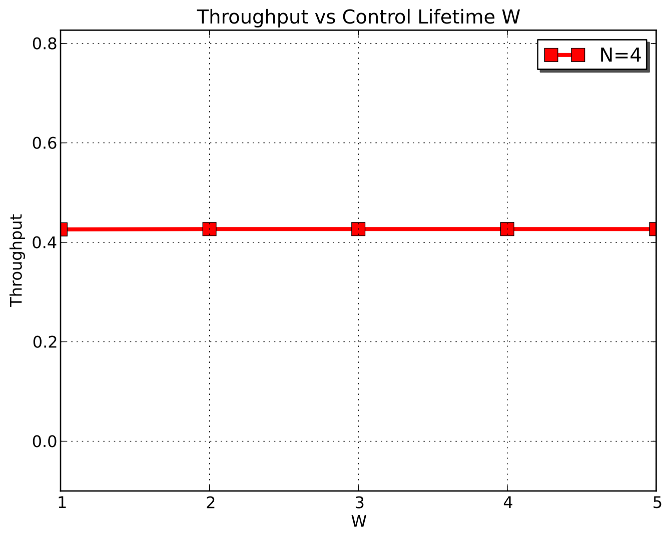


(b)

Figure 4.10: Case (6): Number of nodes $N = 4$ (ring topology) (a) Optimized design variable θ_i versus control lifetime W (b) Throughput versus control lifetime W



(a)



(b)

Figure 4.11: Case (7): Number of nodes $N = 4$ (linear topologies) (a) Optimized design variable θ_i versus control lifetime W (b) Throughput versus control lifetime W

In Fig. 4.11(a), the resulting optimal values of the design variables $\{\theta_i\}$ are shown. In Fig. 4.11(a), we can see that almost no control frames are generated in the optimal solution for any control lifetime W used in the simulation, which is similar to the results for the ring topology (case 6). Again, the reason is that in a linear topology, the network is sparsely populated and fewer transmit collisions have a chance to occur; thus, sending a control frame such as $\hat{d}$ is not preferred due to the overhead involved.

In Fig. 4.11(b), the average throughput of the generated MAC protocol is shown as a function of the control lifetime W. In the figure, we can see that the throughput does not change with the control lifetime, which is expected considering that no control frames are sent at all.

From case (4) to case (7), we see that the optimal design (i.e., the optimal $\{\theta_i\}$) depends very much on the network topology, even though we have the same number of nodes and the same control lifetime. One of the useful aspects of this automated protocol generation is that it gives us insight into understanding quickly what impact the "environmental parameters" (such as the number of nodes, the control information lifetime, and the topology) have on optimality. Hence, our paradigm can be used as an important tool to guide decision-making. Our paradigm provides a significant advantage especially because the "hand-designed" protocols do not offer benchmarks for optimality. Hence, another important contribution of this framework is that it provides such a benchmark for hand-designed protocols whose “design space” falls within the (rather general) design space that is encoded in the transition table.

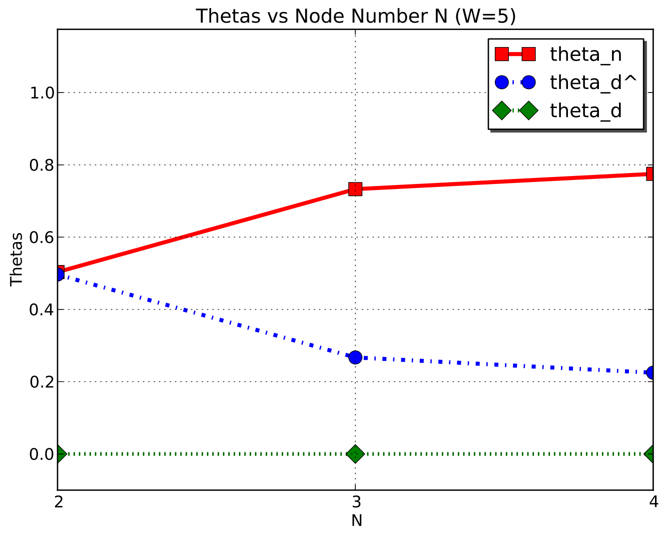
4.4.1 Design Variables and Throughput As Functions of N

In Fig. 4.12(a), the resulting optimal values of the design variables $\{\theta_i\}$ are shown as a function of N, for $W = 5$, in the case of a single neighborhood.

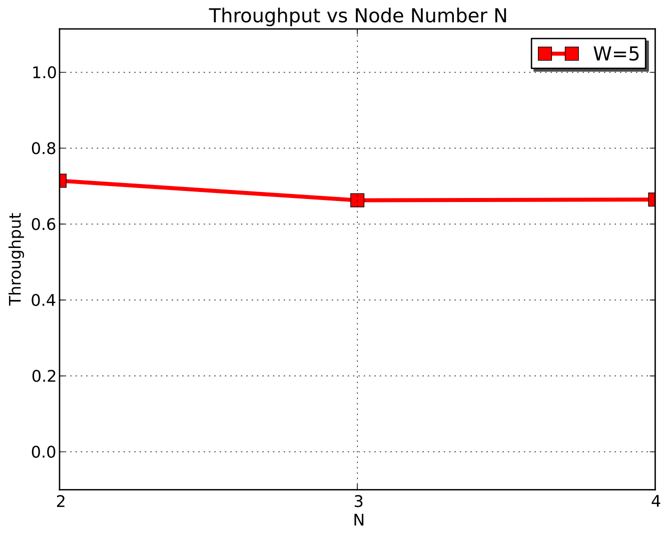
We see that the probability that a node listens in a slot is $\theta_n = 1 - 1/N$; that is, the optimal probability that a node is silent in this model where control information can be sent, is the same as it would be for the pure Random Access Channel. Furthermore, we see that the probability that a data frame d is sent without preceding it with a control frame $\hat{d}$ is $1/N$, and no control information $\hat{d}$ is sent before transmitting the data frame d ($\theta_{\hat{d}} = 0$).

In Fig. 4.12(b), the average throughput of the generated MAC protocol is shown as a function of the number N of nodes. In the figure, we can see that the throughput decreases slightly as the number of nodes increases; the same result has also been shown in Chapter 3.

What is important to notice here is that this result is the same as in Chapter 3, which again validates the results obtained via Symbolic Monte Carlo simulation against the earlier results obtained via the reduced global state method.



(a)



(b)

Figure 4.12: (a) Optimized design variable θ_i versus number of nodes N ($W = 5$) (b) Throughput versus number of nodes N ($W = 5$)

4.4.2 Optimal MAC Protocol Results under Multi-Objective Optimization

In this section, we present the optimal protocols obtained under a weighted combination of energy and throughput, as described in the previous section. In order to bring in energy efficiency into MAC protocol optimization, we add a *sleep* action into our original model (which is achieved by turning off the radio to save transceiver energy in one time slot). This additional action is denoted by s. The probability that a node chooses to go to sleep at the null state ϕ is denoted as $\theta_{s|\phi}$. Recall from the previous section that we denote by λ, the weight for throughput, and by $1-\lambda$, the weight for energy efficiency in the multi-objective function.

In Fig. 4.13, the resulting optimal values of the design variables $\{\theta_i\}$, as a function of λ, for $W = 2$ and $W = 8$, $N = 5$, in case of single neighborhood, are shown.

We see that if the weight for throughput is small, a node chooses optimally to sleep all the time. ($\lambda < 0.2$ for $W = 8$ and $\lambda < 0.4$ for $W = 2$ are examples of switch-over points where it becomes more advantageous for a node to be awake.) As λ increases, the nodes start to send data ($\theta_{\hat{d}|\phi} > 0$) and no longer sleep. We can also see that the "phase transition" (another name for the switch-over point) between sleep and sending data happen at different λs for different control

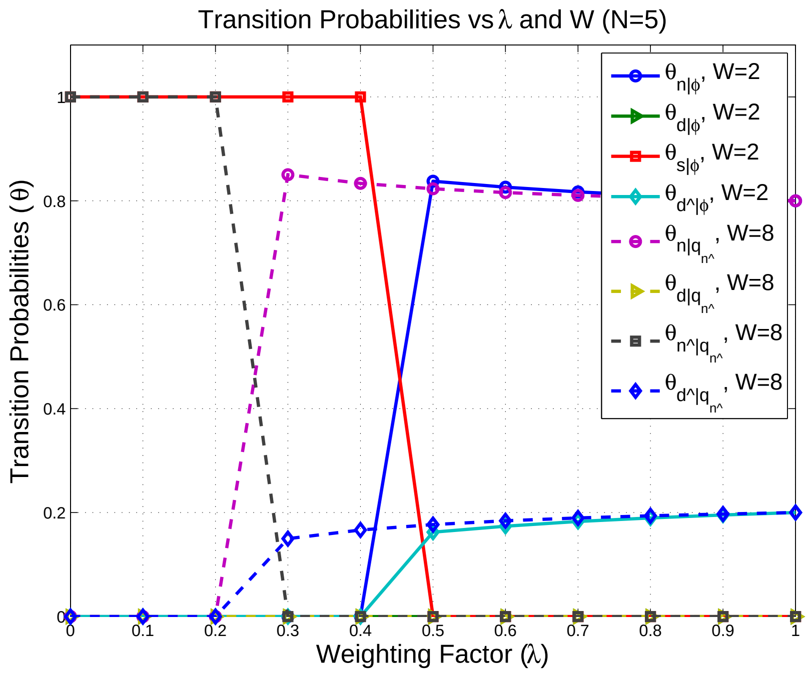


Figure 4.13: Multi-objective optimization: Transition probability versus λ and control lifetime W. (Number of nodes $N = 5$.)

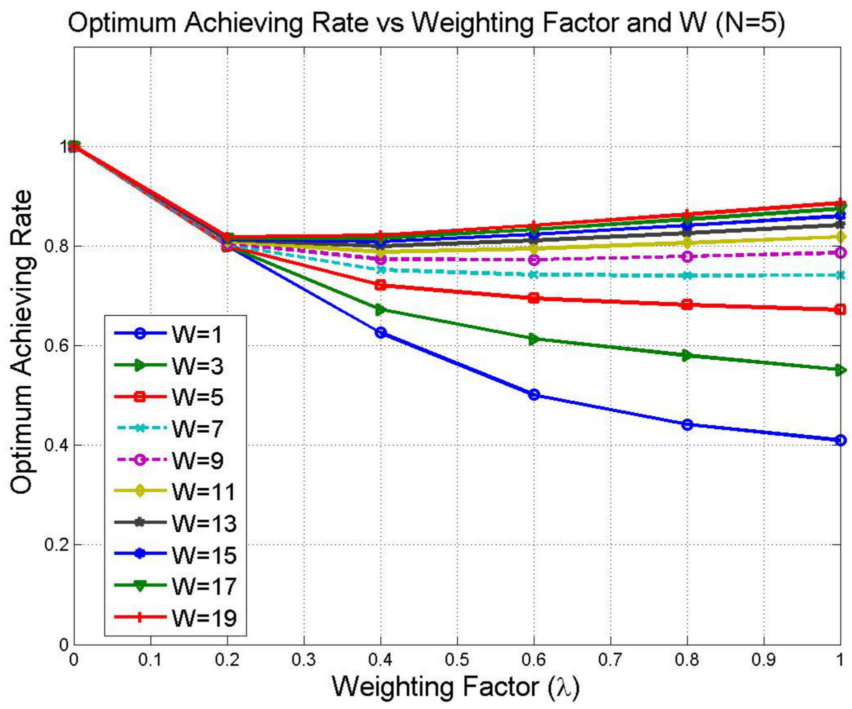


Figure 4.14: Multi-objective optimization: Weighted sum of objectives versus λ and control lifetime W. (Number of nodes $N = 5$.)

lifetimes W: nodes start to send data at smaller λs for larger control lifetimes than for smaller control lifetimes ($\lambda = 0.2$ at $W = 8$ versus $\lambda = 0.4$ at $W = 2$). The reason is that larger control lifetimes can achieve higher throughput and thus displace the optimal point for the trade-off between energy efficiency and throughput toward the throughput side.

In Fig. 4.14, the resulting optimal value for the multi-objective function (denoted as the Optimum Achieving Rate, namely, the normalized weighted sum of multiple objectives), is shown as a function of λ, for $W = \{1, 2, 3, ..., 19\}$, $N = 5$, for a single neighborhood.

We see that, similar to Fig. 4.13, when the weight of the throughput is small, nodes choose to sleep (note that the energy part of the objective function is equal to 1 whereas the throughput part of the objective function is equal to 0). As λ increases, nodes start to send data instead of going to sleep (the energy part of objective function is decreasing while the throughput part of the objective function increases). Overall, a larger control lifetime W can increase the objective function for larger volume of lambda ($\lambda > 0.4$), but with diminishing returns, and only to a certain extent.

From the two figures above, we see that automated design can optimize the design variables, not only for a single design objective, but also for a combination of design objectives, weighted by chosen weighting factors. These results certainly open doors to a variety of potential applications.

4.4.3 Convergence and Complexity

In Fig. 4.15, we show the convergence of the symbolic Monte Carlo simulation. The figure displays the normalized standard deviation of the optimized throughput for a single-neighborhood network with $N = 5$ and $W = 5$. We see that the

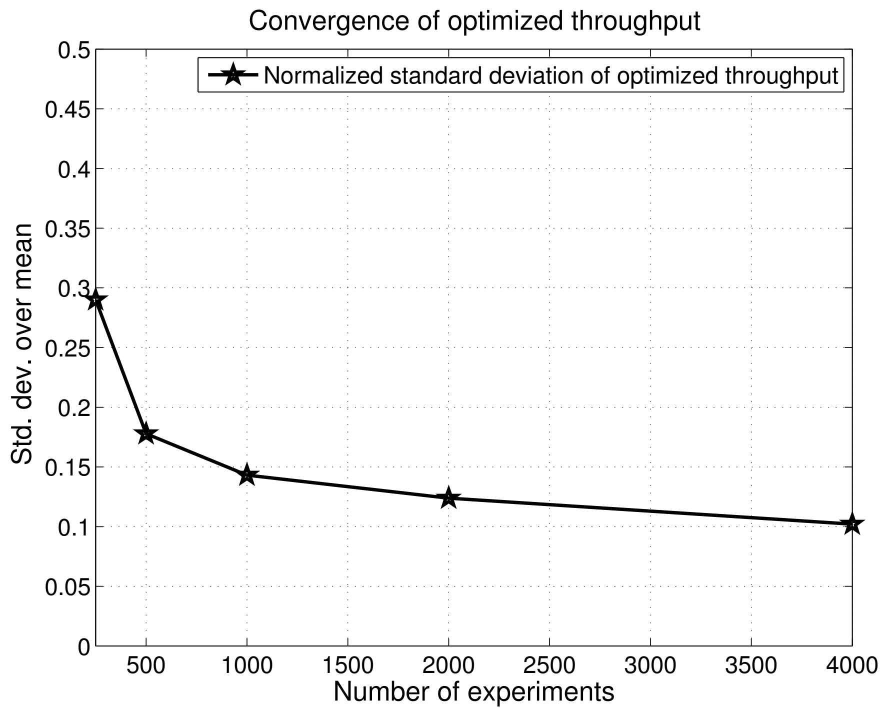


Figure 4.15: Convergence of optimized throughput

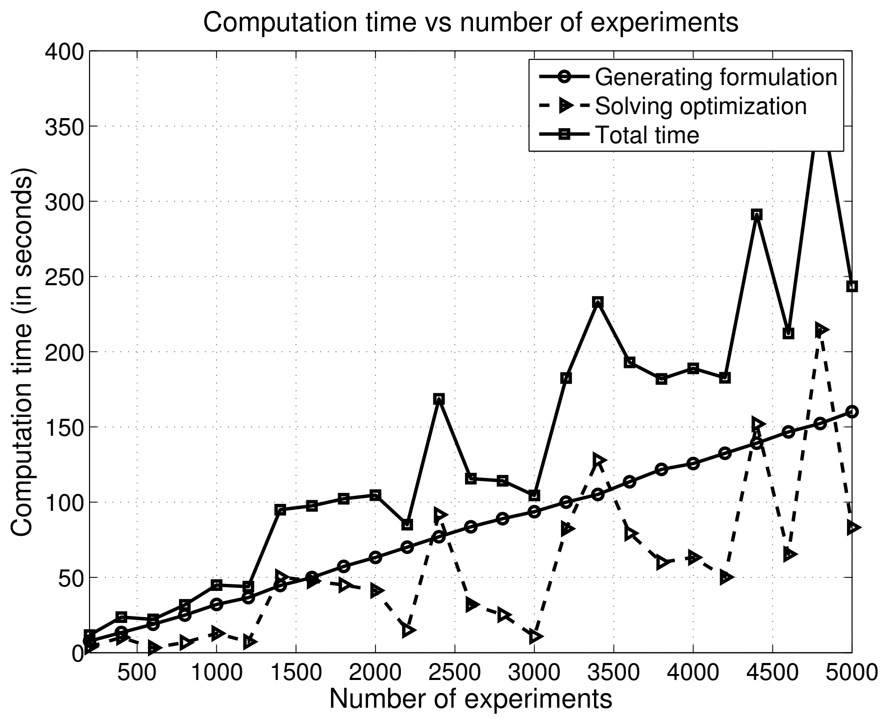


Figure 4.16: Computation time versus the total number of experiments

standard deviation is only 10% of the mean when the number of experiments is larger than 4000.

Finally, in Fig. 4.16, we see that the average computation time increases roughly linearly with the number of experiments. Compared with the computation time for generating the formulation, the time for solving the optimization program has a larger variance. The reason is that the optimization objective function that is generated each time is a random sample in the design space, and thus there is a standard deviation around the time required for the *OpenOpt* solver to solve the program. We expect that the computation time can be largely reduced when the algorithm is implemented in a more efficient language, such as C/C++.

4.5 Summary

In this chapter, we have described a framework for automated MAC protocol generation that models both multiple neighborhoods and acknowledgments, using Symbolic Monte Carlo simulation. As opposed to numerical Monte Carlo simulation methods, the symbolic Monte Carlo method samples the global state space without explicitly generating it, and computes an approximate expression of the objective function of the MAC protocol generation problem. This method works for any multi-objective function. In the next chapter, we shall apply this framework to the case of a mobile network, which has a dynamic topology.

Chapter 5

Optimal MAC Protocol Design for Dynamic Topologies

In this chapter, we extend the framework to address *dynamic* topologies, namely the topologies of wireless ad hoc networks, which change over time. In the previous two chapters, the network topology was static, i.e., the nodes did not change position, and the links were fixed. In contrast, for dynamic topologies, links break and new links form. As a result, at any given point, what can be considered persistent (for a short duration) around each node is a small local topology. In the previous chapter, we saw how we can find the optimal MAC protocol for a local, multi-hop topology under both advertisements of data, and acknowledgments. Here, we utilize the framework of the previous chapter to form an optimization program that weights each such topology according to the frequency with which that topology is observed to occur in a mobile network deployment.

We shall begin with a framework in which we can represent the dynamic wireless network via a subgraph codebook. Each subgraph in this codebook will be a local topology that a node might find itself in during its evolution in a mobile network.

5.1 Representation of Dynamic Wireless Network with a Subgraph Codebook

The problem setting is given as follows: N identical nodes, $n_i \in V$, with $V = \{n_1, n_2, ..., n_N\}$ with the node index $1 \leq i \leq N$, are deployed onto a (two-dimensional) deployment region. We assume that the transmission range, R_{TX}, and the interference range, R_{ITF}, are much smaller than the size of the deployment region, which allows for a multi-neighborhoods MAC. Thus, the N nodes form a graph, $G = (V, E)$ with vertex set V and edge set $E = \{\{n_i, n_j\} | 1 \leq i < j \leq N\}$. For simplicity of mathematical exposition, we assume that all of the nodes are slot-synchronized.

We assume that each node always has data to send to all of its neighbor nodes (that is, its transmission buffers are never empty). We also assume that each node has no knowledge about any other node unless it receives control information via successful reception of a control frame from that node. We assume that the network is "dynamic", i.e., each node $n_i \in V$ moves continuously within the deployment region, with a random speed and direction, which can change with respect to time (indexed as t with $0 \le t \le T_O$, where T_O is the "network operation time" or simulation duration). Depending on the nodes' movement patterns, the links between neighboring nodes can break and form.

We add the subindex t to G and to E to denote the network graph at different times; hence, $G_t = (V, E_t)$ denotes the complete network that evolves over time. (We assume that no nodes are lost or added to the network during the network operation time or simulation duration.) We define a "centered k-hop graph", denoted by $M(\tilde{V}, \tilde{E}; c, k)$ as a graph with a node c (called the "center" of the graph) and a set of nodes $\tilde{V} = \{m\}$ and a set of edges $\tilde{E} = \{m, m'\}$ such that $\text{hop\_count}(c, m) \le k$ and $\text{hop\_count}(c, m') \le k$.

Based on the above assumptions and definitions, we define the "k-hop centered subgraph" of n_i, denoted by $G^k_{i,t}(i) = (V^k_{i,t}, E^k_{i,t}; i)$ around node n_i at time t, as a subgraph of $G_t = (V, E_t)$, with

$$V^k_{i,t} = \{v : \text{distance}(v, n_i; t) \le R_{\text{ITF}} \cdot k, v \in V\} \subseteq V, \tag{5.1}$$

$$E^k_{i,t} = \{\{v, u\} : v, u \in V^k_{i,t}, v \ne u, \{v, u\} \in E_t\} \subseteq E_t, \tag{5.2}$$

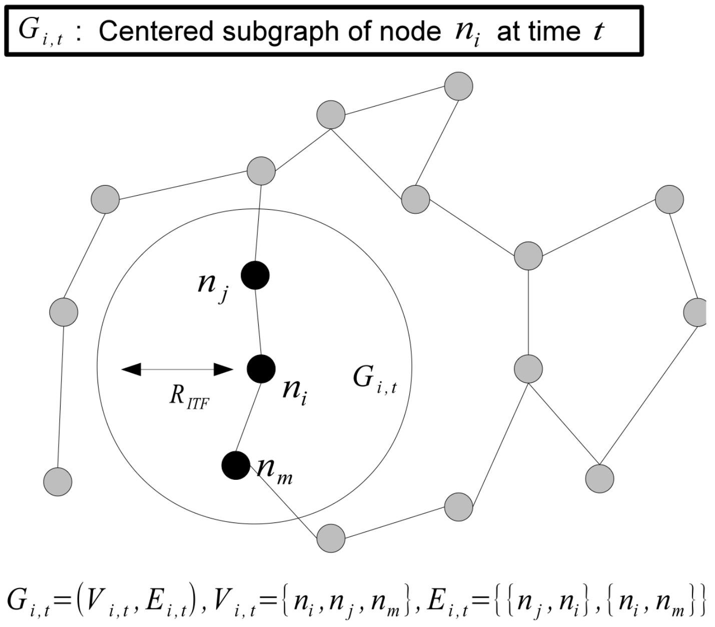

$G_{i,t}$: Centered subgraph of node n_i at time t
n_j
n_i
n_m
R_{ITF}
$G_{i,t}$
$G_{i,t}=(V_{i,t},E_{i,t}), V_{i,t}=\{n_i,n_j,n_m\}, E_{i,t}=\{\{n_j,n_i\},\{n_i,n_m\}\}$

Figure 5.1: Centered subgraph of node n_i at time t, i.e., $G_{i,t}$

where "distance$(v, u; t)$" is the physical distance between nodes v and u at time t. (Note that this may be roughly indicative of the hop count between v and u, but not necessarily so.)

In Fig. 5.1, an example of a 1-hop centered subgraph is given; $G^1_{i,t}$ is the 1-hop centered subgraph centered around node n_i at time t. The k-hop neighborhood subgraph of any node eventually converges to the whole network graph G_t, as k increases. In this chapter, we consider only the 1-hop centered subgraph, $G^1_{i,t}$; thus, we simply write it as $G_{i,t}$, and write $V^k_{i,t}$ as $V_{i,t}$ and $E^k_{i,t}$ as $E_{i,t}$, in the rest of the chapter. Then, the "subgraph codebook" of $\{G_t\}_{t=0}^{t=T_o}$, denoted by $\mathcal{C}_G$, is a hash table with elements $(\mathcal{C}_G)_{\text{key},w}$ such that each *key* of $\mathcal{C}_G$ is a centered graph $M(\tilde{V}, \tilde{E}; c, k)$ and each *value* of $\mathcal{C}_G$ is the fraction of the elements of $\{G_t^k(i) | 0 \leq t \leq T_O, 1 \leq i \leq N\}$ that are isomorphic to $M(\tilde{V}, \tilde{E}; c, k)$.

In this chapter, we are interested in the throughput of the network. First, we define node n_i's "instantaneous throughput", $\tilde{F}_{i,t}$, as the number of received frames in one time slot, at node n_i and at network time t. In our formulation, we let each data frame occupy exactly one time slot; thus, $\tilde{F}_{i,t}$ is a Bernoulli random variable (with the two values 0 and 1). Because the same MAC protocol is deployed for all of the nodes in the network, the probability distribution of $\tilde{F}_{i,t}$ depends only on the network graph at time t and node n_i's position in the graph. Note that because the physical movement is very slow in comparison to the speed of frame transmission, we are justified in using a "static network" model. Then, we define the "individual node's short-term throughput" at time t, denoted by $F_{i,t}$, as the conditional expectation of the node's instantaneous throughput, namely as $F_{i,t} = \mathrm{E}[\tilde{F}_{i,t} | G_t]$. The "node's average throughput", $\bar{F}_i$, is defined as the time average of the node's short-term throughput, namely as $\bar{F}_i = \frac{1}{T_o} \int_0^{T_o} F_{i,t} \mathrm{d}t$. Finally, we define the "per-node average throughput" of the network[1], $\bar{F}$, as the average of the throughput of all the nodes in the network, namely as $\bar{F} = 1/N \sum_{i=1}^{N} \bar{F}_i$.

[1]Other metrics or weighted combinations thereof may be substituted while retaining this framework.

Putting these together, we have

$$\bar{F} = \frac{1}{T_o} \int_0^{T_o} \mathrm{d}t \left(\frac{1}{N} \sum_{i=1}^{N} F_{i,t} \right) \tag{5.3}$$

Now, we use the fact that $F_{i,t} = \mathrm{E}[\tilde{F}_{i,t}|G_t] \approx \mathrm{E}[\tilde{F}_{i,t}|G_{i,t}]$, where G_t is the network graph at time t and $G_{i,t}$ is "the centered subgraph around node n_i at time t", which means that the knowledge of the local network around node n_i suffices in computing the expectation. (This is due to the locality of a MAC channel.) Thus, we have

$$\begin{align} \bar{F} &= \frac{1}{T_o} \int_0^{T_o} \mathrm{d}t \left(\frac{1}{N} \sum_{i=1}^{N} \mathrm{E}[\tilde{F}_{i,t}|G_t] \right) \tag{5.4} \\ &\approx \frac{1}{T_o} \int_0^{T_o} \mathrm{d}t \left(\frac{1}{N} \sum_{i=1}^{N} \mathrm{E}[\tilde{F}_{i,t}|G_{i,t}] \right) \tag{5.5} \end{align}$$

We define the "subgraph counting function" $\mathrm{A}((\mathcal{C}_G)_{\mathrm{key}}, t)$ as

$$\mathrm{A}((\mathcal{C}_G)_{\mathrm{key}}, t) \equiv |\{G_{i,t}|G_{i,t} \cong (C_G)_{\mathrm{key}}, 1 \le i \le N\}| \tag{5.6}$$

which counts the number of the subgraphs $G_{i,t}$ at time t, which are isomorphic ("$\cong$" means "graph isomorphism") to the codebook key $(C_G)_{\mathrm{key}}$. Recall that $\tilde{F}_{i,t}$ is the throughput of node n_i in the centered subgraph $G_{i,t}$. (Note that $\tilde{F}_{i,t}$ is a random variable.) Note that $\tilde{F}_{i,t}$ and $\tilde{F}_{j,t'}$ ($n_i, n_j \in V$, and $0 \le t, t' \le T_O$)

are identically distributed if $G_{i,t} \equiv (\mathcal{C}_G)_{\text{key}} \equiv G_{j,t'}$ for the same key *key*. We let $\tilde{F}_{\text{key}}$ be a random variable with this identical distribution. Let $\mathcal{K}(\mathcal{C}_G)$ denote the set of all the keys of $\mathcal{C}_G$. Then, we assume that the process $\mathrm{A}((\mathcal{C}_G)_{\text{key}}, t)$ is a strongly ergodic process in the sense that the time average of $\mathrm{A}((\mathcal{C}_G)_{\text{key}}, t)$ converges, for every $(\mathcal{C}_G)_{\text{key}}$, to its ensemble average for every realization of the dynamic network's evolution. Then,

$$\bar{F} = \lim_{T_o \to \infty} \frac{1}{T_o N} \int_0^{T_o} \mathrm{d}t \sum_{(\mathcal{C}_G)_{\text{key}} \in \mathcal{K}(\mathcal{C}_G)} \mathrm{E}[\tilde{F}_{\text{key}} | (\mathcal{C}_G)_{\text{key}}] \cdot \mathrm{A}_\omega((\mathcal{C}_G)_{\text{key}}, t) \tag{5.7}$$

for every realization ω of the network evolution. (Above, $\mathrm{A}_\omega((\mathcal{C}_G)_{\text{key}}, t)$ denotes the (unique) realization of the process $\mathrm{A}((\mathcal{C}_G)_{\text{key}}, t)$ that corresponds to the realization ω of the network's evolution.) Then,

$$\bar{F} = \sum_{(\mathcal{C}_G)_{\text{key}} \in \mathcal{K}(\mathcal{C}_G)} \mathrm{E}[\tilde{F}_k | (\mathcal{C}_G)_{\text{key}}] \lim_{T_O \to \infty} \frac{1}{T_o} \int_1^{T_o} \mathrm{d}t \left(\frac{\mathrm{A}((\mathcal{C}_G)_{\text{key}}, t)}{N} \right) \tag{5.8}$$

Because $\mathrm{A}((\mathcal{C}_G)_{\text{key}}, t)$ is strongly ergodic in the sense defined above,

$$\lim_{T_O \to \infty} \frac{1}{T_O} \int_0^{T_O} \mathrm{d}t \left(\frac{\mathrm{A}((\mathcal{C}_G)_{\text{key}}, t)}{N} \right) = \mathrm{P}[(\mathcal{C}_G)_{\text{key}}] \tag{5.9}$$

It is worthwhile to note that this is the only assumption that we make about the network dynamics; note that, the nodes' movements are not restricted to any particular mobility model.

Based on the above assumption, we have

$$\bar{F} = \sum_{(\mathcal{C}_G)_{\text{key}} \in \mathcal{K}(\mathcal{C}_G)} \mathrm{E}[\tilde{F}_{\text{key}} | (\mathcal{C}_G)_{\text{key}}] \mathrm{P}[(\mathcal{C}_G)_{\text{key}}] \tag{5.10}$$

For simplicity, we write $\mathrm{P}[(\mathcal{C}_G)_{\text{key}}]$ as $\phi_{\text{key}} \in \Phi$, where Φ is the probability distribution on $\mathcal{K}(\mathcal{C}_G)$. Then,

$$\bar{F} = \sum_{(\mathcal{C}_G)_{\text{key}} \in \mathcal{K}(\mathcal{C}_G)} \mathrm{E}[\tilde{F}_{\text{key}} | (\mathcal{C}_G)_{\text{key}}] \cdot \phi_{\text{key}} \tag{5.11}$$

We see in (5.10) that the per-node average throughput is equal to the average of the throughputs of the subgraphs, weighted by their frequency of occurrence. We also note that a dynamic wireless network can be efficiently represented by the pair $< \mathcal{C}_G, \Phi >$, i.e., the subgraph codebook.

In order to demonstrate the subgraph codebook intuitively, we give a simple example in Fig. 5.2: This example uses a network with periodic dynamics; however, as can be seen in the development that leads up to (5.11), no periodicity is necessary for (5.11) to hold. From time $t = t_0$ to $t = t_0 + 1$, ($t_0 > 0$), n_i is moving from the left to the right while breaking the links on its left and form-

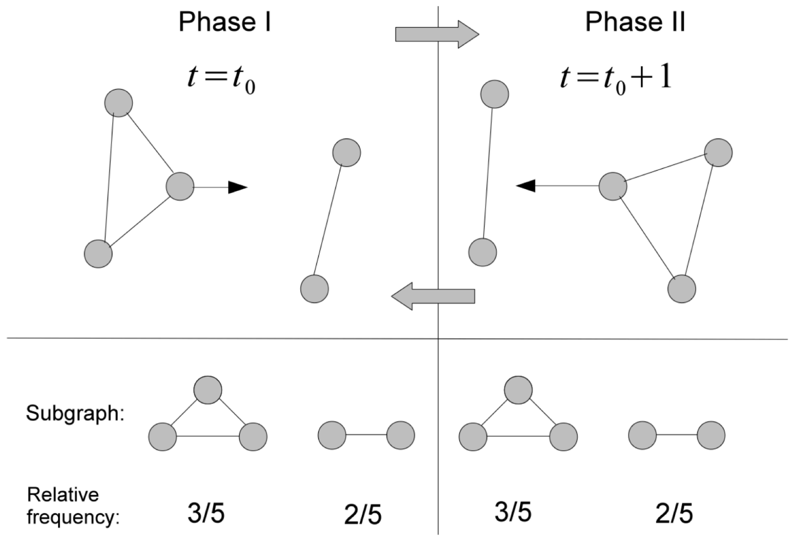


Figure 5.2: A simple example that illustrates a periodic dynamic network and its corresponding codebook

ing new links on its right. The probability distribution (computed as a relative frequency) on the subgraphs remains the same, despite the fact that the global topology oscillates between the two shown topologies.

5.2 Optimal MAC Protocol Design for Dynamic Topologies

Based on the above probabilistic model of a dynamic network, we solve the automated MAC protocol generation problem by using the symbolic Monte Carlo method [47]. The objective function we use in this chapter is the "per-node average throughput" and is equal to the Φ-weighted sum of the individual node's throughput given, as described in the previous section, by

$$\bar{F} = \sum_{(\mathcal{C}_G)_{\mathrm{key}} \in \mathcal{C}_G} \phi_{\mathrm{key}} \cdot \mathrm{E}[\tilde{F}_{\mathrm{key}} | (\mathcal{C}_G)_{\mathrm{key}}] \tag{5.12}$$

where $\phi_{\mathrm{key}} \in \Phi$ and $(\mathcal{C}_G)_{\mathrm{key}} \in \mathcal{C}_G$ in codebook $< \mathcal{C}_G, \Phi >$. Thus, this maximizes the expected value of the objective function (e.g., throughput) over this probabilistic model of the network. Note that a *single* protocol is generated.

The MAC model we use in this chapter is similar to the one in the previous chapter, so only a brief description shall be given here. In the MAC model, only a minor difference exists, namely that we have disabled the acknowledgment for the received data frame, because it was found that this was never used in the optimal solution in the case of throughput maximization (see the results in [47]). Thus, each node takes any one of the five actions in every slot: (1) n; the node remains silent, (2) d; the node sends a data frame, (3) a; the node sends an acknowledgment, (4) c; the node sends a control frame that is not an acknowledgment, by which it tells (1-hop) neighbors that it will send a data frame for the following W slots, where W is the "control lifetime", i.e., the number of slots for which the effect of the control frame on the receiving node lasts, (5) c_a, which has the same meaning as c but which requires that the receiver node send back an acknowledgment a right after it receives c_a. Note that a is an acknowledgment only for a control frame; there are no acknowledgments for data frames. Once the node receives a, it takes hold of the channel if a is destined to it and keeps silent if it is not destined to it. Thus, the actions that a node can take form the set of actions denoted by $\mathcal{A} = \{n, d, c, c_a, a\}$. We define the "initial set of actions", as a subset of this action set, as $\mathcal{A}_I \equiv \{n, d, c, c_a\}$.

We define the state of a node as $< s, w >$, where $s \in \mathcal{S}$ is the "knowledge state", which encodes that node's knowledge regarding its control information exchanges (namely, this encodes the control frames that this node has sent, and the control frames that it has received in the last W slots), and $w \in \{1, 2, ..., W\}$ is the state timer which counts down and records how many slots the node will still be in this state. We assume unicast data transmission and broadcast control transmission, i.e., only the destined receiver node decodes successfully the data frame d that it receives, while each of the neighbor nodes may decode all of the control frames that it receives. Thus, the set of knowledge states is denoted by $\mathcal{S} = \{\phi, s_c, s_{ca}, s_a, q_c, q_{ca}, q_a, l_c, l_{ca}, l_a\}$, where ϕ is the null state, in which no control information has been sent or received by the node during the last W slots; s_c, s_{ca}, s_a mean that c, c_a or a (respectively) has just been sent in the last slot; q_c, q_{ca} and q_a mean that c, c_a or a (respectively) has been received in the last slot and the frame was destined to this node; l_c, l_{ca} and l_a mean that c, c_a or a has been

state s	action x	$P(x\|s)$	timer $w(x\|s)$
ϕ	n, d, c, c_a	θ_n, θ_d, θ_c, θ_{ca}	1, 1, 1, 1
s_c	d	1	W (control lifetime)
s_{ca}	n	1	1
s_a	n	1	W
q_c	n	1	W
q_{ca}	a	1	W
q_a	d	1	W
l_c	n	1	W
l_{ca}	n	1	$W+1$
l_a	n	1	W

Table 5.1: Transition rules of a node upon arriving in state s

received in the last slot but was destined to another node (such a case is due to the broadcast nature of control transmission).

Upon arriving at one of these states, the node chooses an action $x \in \mathcal{A}$ with probability $P(x|s)$, according to the transition rule of this state, and starts the timer with $w = w(s)$ accordingly, and the timer counts down each slot until it triggers another state transition when it times out. The transition rules of this model are summarized in Table 5.1. In this table, θ_n, θ_d, θ_c, θ_{ca} ($\sum \theta = 1$) are the *design variables* (a.k.a. decision variables) of the optimization. Note that this table is not a complete transition table for the finite state machine (FSM) of a node; for simplicity, it describes only the transition rules of phase 2 for a 2-phase FSM (see [47]). The phase 1 transition rules (from actions to next state) are rather straightforward, and are not shown.

In order to optimize the performance of the MAC protocol (recall that our chosen metric is throughput in this chapter), we utilize the *Symbolic Monte Carlo* method [47] to explore the global state space of the network; we collect symbolic terms, and accumulate the symbolic expressions for the two metrics of interest that help us compute the average throughput. These two metrics are $E[F]$, namely, the average number of successful transmissions per cycle, and $E[T]$, the average length of the cycle, where a cycle is defined as the time between two subsequent visits to the same recurrent state. The long-term average throughput can be computed as $E[F]/E[T]$. The key aspect of the Symbolic Monte Carlo method is that the whole state space need not be explored; this would be a computationally intensive task even for a small network with multiple neighborhoods. Instead, the state space is sampled by running a symbolic Monte-Carlo simulation in which symbolic expressions are collected via only the sampled routes through the state space. In the end, an approximation of the objective function is obtained. Finally, after the objective function has been computed by Symbolic Monte Carlo, the resulting non-linear optimization program is solved by using the *openopt* package available for Python.

5.3 Simulations

The simulation set-up is as follows: Ten moving nodes are randomly deployed, according to a uniform distribution, on a 600m$\times$600m square area. Each node is identical and moves according to the Random Waypoint Model, with a constant velocity $v = 80$m/min (for simplicity in this simulation, but the approach would work for any dynamic model). The "pause time" at each waypoint is zero minutes. Each node is equipped with a 802.11b/g radio and has a transmission range of 95m outdoors (following a circular transmission model). To simplify the model, we assume that the interference range is the same as the transmission range. The time span of the simulation is 100 minutes, and the network topology is computed every minute, i.e., 100 graphs are generated in the simulation.

In order to evaluate the proposed automated MAC generation framework for dynamic topologies, we built a simulation environment in Python, incorporated with three main packages: *pylab*, which enables a MATLAB-like working environment; *networkx*, which provides powerful graph-related functions; and *openopt*, which is able to solve the resulting non-linear optimization problem efficiently. All the simulations were run on a Lenovo ThinkPad SL410 laptop, with no other concurrent computationally intensive processes.

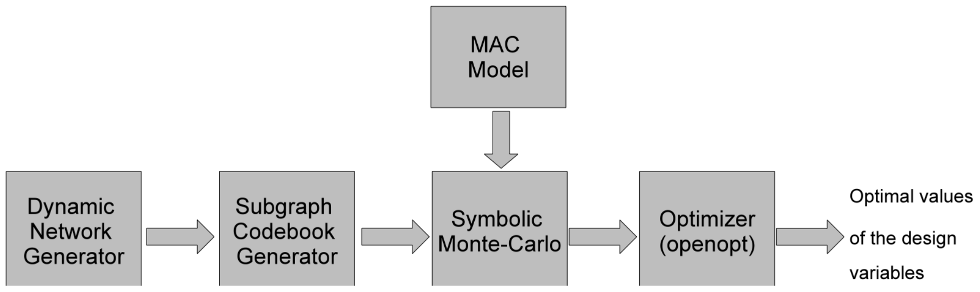

MAC Model
Dynamic Network Generator
Subgraph Codebook Generator
Symbolic Monte-Carlo
Optimizer (openopt)
Optimal values of the design variables

Figure 5.3: Simulation structure

The simulation structure is illustrated in Fig. 6.4: the dynamic network generator randomly generates the traces of the nodes for a dynamic network. The subgraph codebook generator takes in the traces of the nodes and builds the corresponding subgraph codebook for this dynamic network. Based on the codebook and the given MAC model, symbolic Monte Carlo simulation explores the design space and generates an approximate symbolic expression for the objective function. At the end, the values of the optimal design variables are generated by the optimizer.

We describe a refinement of the original symbolic Monte Carlo method in order to speed up its convergence. This approach accelerates the symbolic Monte Carlo method by intelligently choosing the samples which are never duplicates of each other. In the first stage, we exhaustively search the design space using depth-first search for the first D_{sd} steps, starting from the null state ϕ, where D_{sd} is chosen dynamically and implicitly: once the depth-first search collects enough sample

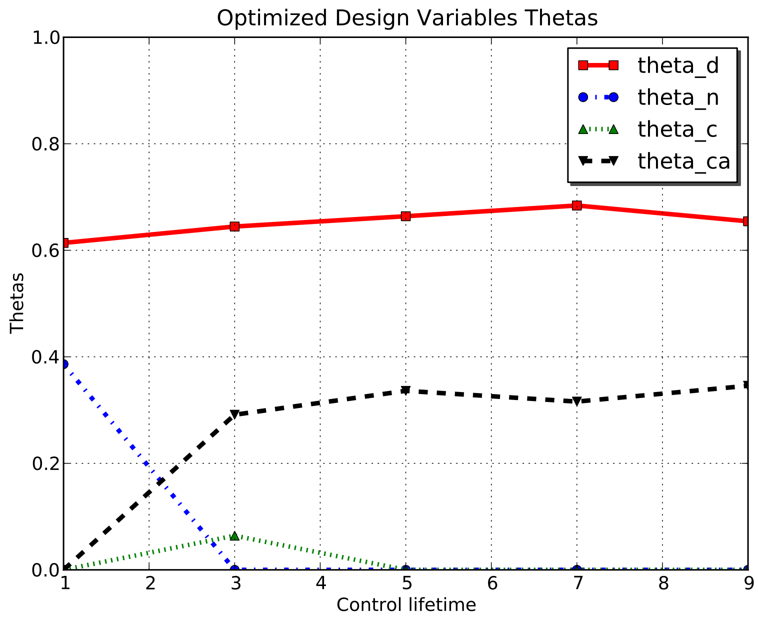


Figure 5.4: Optimized design variables

paths (> 1000 in this simulation) at the l-th step, D_{sd} is set to l and we finish the first stage. In the second stage, we continue each of the sample paths built in the first stage and run the original symbolic Monte Carlo method, in which each next step is chosen randomly until the path loops back to the null state. (If the path loops back to ϕ within the first D_{sd} steps, we terminate that path and keep it as one of the sample paths.)

In Fig. 5.4, we show the optimized design variables $\{\theta_i\}$ under different control lifetimes, from 1 to 9. In order to make a fair comparison among different control lifetimes, we use the same nodes' traces, generated by the dynamic network generator. We can see in this figure that for different control lifetimes the optimal design variables are different, and that the figure can be divided into three main regimes: 1) the control lifetime is 1, where no control frame (c or c_a) is sent because the benefit of successfully transmitting control frames is too small; 2) the control lifetime is 2 or 3, where all the node actions are used with different weightings; 3) the control lifetime is larger than 3, where only the control frame c_a is sent. Each of these regimes corresponds to a protocol that is structurally different from the others; hence, this provides an example as to how this framework generates structurally different protocols under the same umbrella. In the traditional hand-design of protocols, the human designer makes a (usually implicit) guess at what the underlying control lifetime is (this may be governed, for example, by the coherence time of traffic generation under dynamic traffic conditions, which is not addressed in this chapter). Based on this implicit guess, a protocol is intuitively designed to include or exclude a control frame such as c_a, in a hand-designed protocol. Moreover, most hand-designed protocols comprise only very elementary randomness in the choice of node actions; in contrast, above, optimal weightings according to which nodes take actions from the action set are specified and become part of the protocol description. The main reason that this can be done is that optimality that incorporates control information exchanges has been formulated a priori in our framework, whereas no such rigorous optimality framework that includes control information generation exists in

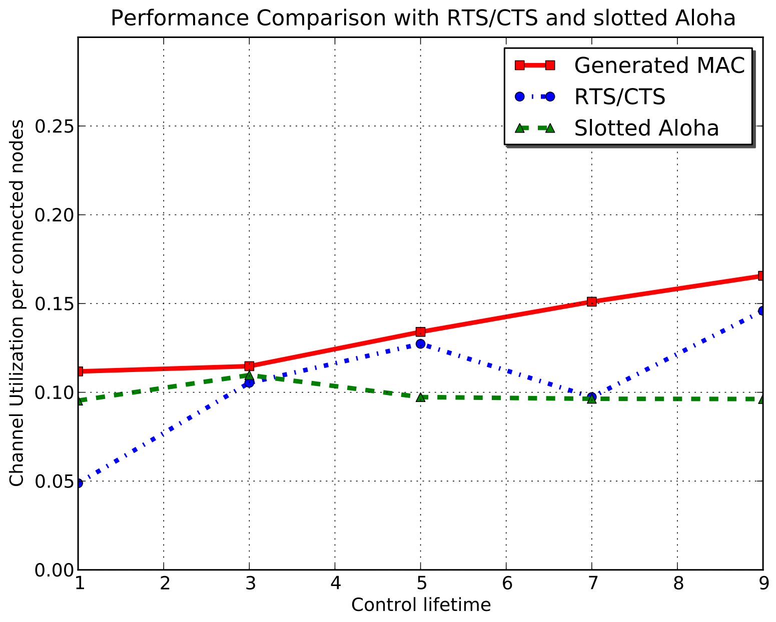


Figure 5.5: Per-node throughput

the hand-design of protocols. Numerical optimizations exist for protocols whose control structure has been fixed; however, the above framework differs from these in that structurally different protocols are subsumed under the same umbrella.

In Fig. 5.5, we display the per-node throughput under different control lifetimes, where the control lifetime W ranges from 1 to 5. When we compare the designed MAC protocol with the widely used RTS/CTS protocol, we see that the designed MAC protocol outperforms RTS/CTS for all Ws in this range. The reason is that RTS/CTS does not utilize the distribution with which local subgraphs appear in the global network. This is both a strength and a weakness of RTS/CTS. The graph shows the extent to which this information, when available, can be utilized to improve the throughput, while relying only on local information. This type of information, as displayed in this graph, also provides a partial answer to the more general question that we raised earlier, namely, a definition of the optimality of protocols with respect to differing levels of side information. While RTS/CTS is not provably optimal for the case without the side information on the distribution of local topologies, because it is a feasible solution, it provides a lower bound for the performance of protocols without this side information. Thus, the graph displays the performance difference between this feasible solution and the optimal solution (i.e., protocol) with the side information on the probability distribution of the local topologies. In Fig. 5.5, we can also see that slotted ALOHA performs the worst except for when the control lifetime is less than 3, in which case RTS/CTS performs worse than slotted ALOHA. THe reason is that for the smaller control lifetime, RTS/CTS has a relatively large control overhead whereas slotted ALOHA does not.

Next, we analyze the computational complexity of the optimal protocol obtained for the same dynamic network. In Fig. 6.8, the computational complexities (characterized by the execution times) for the three major parts of the framework are shown. It can be seen that expression generation (i.e., state space exploration with Symbolic Monte Carlo) accounts for most of the complexity. While Python may be replaced by a more efficient language, this result points to a general re-

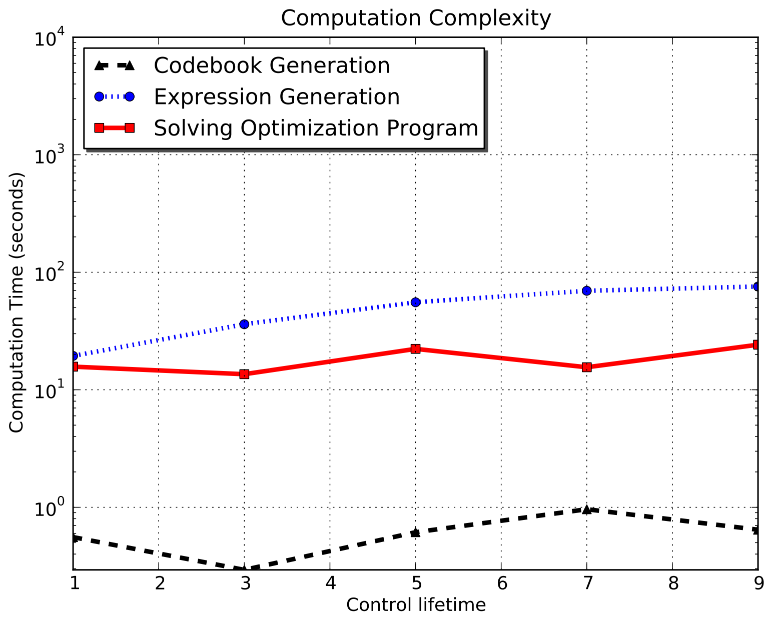


Figure 5.6: Computational complexity

sult that we have seen so far, namely that the time to generate the optimization program dominates over the time to solve the program using non-linear state-of-the-art solvers.

5.4 Summary

In this chapter, we have presented a framework for automated MAC protocol generation for dynamic topologies. The key idea in handling network dynamism is to model the network via a collection of local topologies (a.k.a. subgraphs) and their probability distribution. Empirical frequencies of these local topologies collected during simulation serve to approximate this probability distribution. An optimal protocol for a dynamic topology is defined with respect to this model. In this chapter, we have solved this optimal protocol generation problem for dynamic topologies by generating a mathematical program using the symbolic Monte Carlo method. We have also quantified the value of side information on the distribution of local topologies by comparing the performance of the optimal protocol with RTS/CTS. The results on computational complexity show that the problem can be solved in a reasonable time, and that the time to generate the approximate program dominates the time to solve the generated optimization program.

One of the key assumptions in this chapter has been that the frequency distribution with which topologies occur is available to the protocol designer off-line. The question arises as to how time-varying frequencies of subgraphs can be addressed in an on-line fashion. Even though this question is beyond the scope of this dissertation, we note here that the frequency of subgraphs can be maintained by nodes in a distributed fashion, and this information can be fed back to a protocol optimization engine that can re-run the optimization program with updated frequencies. If this can be done within the coherence time of the network, then this method can be used in an on-line fashion. The design of *online* optimal protocol generators is beyond the scope of this thesis.

Chapter 6

Optimal MAC Protocol Design for Dynamic Traffic Conditions

The main goal of this chapter is to extend our methodology for MAC protocol generation to address *dynamic* traffic demands, that is, traffic demands that are different at each node and that vary with time. In the previous chapters, the nodes were assumed to have an infinite amount of data to send in their buffers. However, realistic networks have bursty traffic patterns, and significant differences in the long-term traffic demands of the nodes can exist. In this case, a methodology has to be developed to address both long-term traffic needs as well as short-term bursty traffic patterns (which we assume are known only at the transmitting node at a given time). A major challenge is to weave these long-term and short-term dynamic and variable demands into a single optimization framework that can generate a MAC protocol.

In this chapter, we model stationary networks with multiple MAC neighborhoods in which nodes can exchange advertisements and acknowledgments whose effects are fully modeled, under dynamic traffic demands.

6.1 Assumptions and Preliminaries

We assume that N identical nodes, $V = \{1, 2, ..., N\}$, have been deployed onto a two-dimensional deployment region. We assume that the transmission range, R_{TX}, and the interference range, R_{ITF}, are much smaller than the size of the deployment region, which allows for a multi-neighborhood MAC protocol. A bidirectional link l is formed as $l = \{i, j\}$ iff nodes i and j are located within transmission range of each other. As a result, the N nodes form an undirected transmission graph $G = (V, E)$ with vertex set V and edge set $E = \{\{i, j\}|1 \leq i, j \leq N\}$.

In this chapter, we consider only unicast data transmission. We further assume that at each time, a node has a single radio that allows the transmission of a MAC frame to only one neighbor. Except for possibly different traffic demands that they will see at different times, we assume that all of the nodes are identical. We now switch to the perspective of any one of these nodes. At any given time, if there is an outgoing transmission of a MAC frame at this node, we can examine the traffic stream with which this MAC frame is associated, and record the traffic demand associated with that stream outgoing to that neighbor. We denote the rate of this traffic demand by a continuous function, $\mu(t)$ (in bits per second). (Note that this notation does not require the index of either the sending node or the receiving node. We shall see in the next section that the behavior of each node is the same for a given traffic demand at that time. This is possible by virtue of the MAC layer abstraction, which separates it from routing issues.) Because the traffic in a wireless network is bursty, we model the current traffic demand of this node for this traffic stream as $\mu(t) = \sum_m h_m \delta(t - t_m)$, where m is the index for the mth traffic "spike", h_m is the number of frames in the mth "spike", and t_m is when the m-th spike occurs. (Note that this definition is for the traffic stream associated with the current outgoing MAC frame at this node. We fix that stream and examine its traffic demand pattern over time.) An example is given in Fig. 6.1. Because the traffic stream that has been singled out belongs to a single application, which might have an end-to-end delay requirement, we incorporate soft delay guarantees into our framework by imputing an average target link delay of D_m seconds to each spike based on the average remaining time for the delivery of the frames in that spike. (This assumption on the higher-layer routing layer is invoked only to give one example as to how a delay constraint can arise at the MAC Layer.)

In order to create a mathematical model on which the long-term averages of the next section is based, we define the "effective traffic demand" as $\bar{\mu}(t) = \sum_m \frac{h_m}{D_m} s_m(t-t_m)$, where $s_m(t)$ is the rectangle function with unit height and duration D_m, as shown in Fig. 6.1. Considering that

$$\int_t^{t+D_m} \frac{h_m}{D_m} s_m(t)dt = \int_t^{t+D_m} h_m \delta(t)dt \tag{6.1}$$

$\bar{\mu}(t)$ is a smoothed version of $\mu(t)$. This effective traffic demand, $\bar{\mu}(t)$, sets the minimal average throughput around time t, which needs to be supported by the MAC protocol such that each frame meets the soft delay guarantee. (We assume that the retransmissions from the Data Link Layer are incorporated into the traffic demands; however, we do not model the Data Link Layer in this chapter.) Let $F(\mu(t))$ be the achieved throughput around t, under traffic demand $\mu(t)$. Then,

$$\mathrm{E}[F(\mu(t))] \geq \beta\bar{\mu}(t), \tag{6.2}$$

is our requirement on the average throughput, where $\beta \geq 1$ is a margin specified by the soft-delay guarantee (In the simulation of this chapter, we let $\beta = 1$).

We assume that the nodes are slot-synchronized and that the slot size is equal to the duration of a frame. Thus, the slot duration is assumed to be much smaller than the coherence time, T_c, of the effective traffic demand, which is the duration

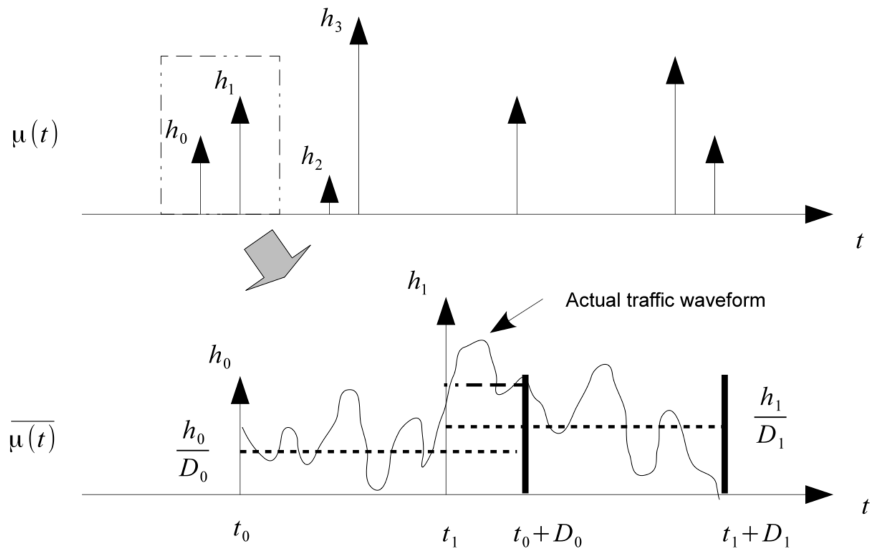


Figure 6.1: Network traffic pattern and "effective coherent time of traffic"

over which the effective traffic demand remains roughly constant. Then, $\bar{\mu}(t)$ can be discretized into $\bar{\mu}_k$, as $\bar{\mu}_k \equiv \bar{\mu}(kT_c)$.

We assume that only a single (data or control) frame can be successfully transmitted by a node, or received by a node in each time slot. A collision occurs at a time slot at the receiver when at least two frames overlap in that time slot, in which case no frame can be decoded correctly for that time slot. We assume that each node has no knowledge about any other node unless it obtains control information through a successfully received control frame.

6.2 Formulation of MAC Protocol Optimization Problem For Dynamic Traffic Conditions

The first key idea in this section is that a MAC protocol is uniquely characterized by the choice of the design functions that are functions of the local effective traffic demand at the current time. We shall write a *functional* optimization program that optimizes over these functions over the entire set of links in the network. Even though this functional optimization program will be global, the information that is assumed to be known to each node is only its own $\bar{\mu}$ at the current time, which is locally available. Because each node acts identically, the goal is to obtain *off-line*, a look-up table for the probability that a node takes a given action given its knowledge state *as a function of* $\bar{\mu}$.

We can describe our design space generally by defining the design functions in vector form, as $\boldsymbol{\theta}(\cdot) = [..., \theta_x(\cdot), ...]^T$ with $x \in \mathcal{A}$, where $\mathcal{A}$ is the set of actions that a node can take given its knowledge state. (See [47].) Thus, the goal of automatic MAC protocol generation for dynamic traffic is to find the optimal $\boldsymbol{\theta}(\cdot)$ as a vector function of the local effective traffic demand $\bar{\mu}_k$.

Let $\bar{\boldsymbol{\mu}}_k$ denote the *vector* of effective traffic demands over all of the links in the network. Then, let $C(\bar{\boldsymbol{\mu}}_k; \boldsymbol{\theta}(\cdot))$ and $F(\bar{\boldsymbol{\mu}}_k; \boldsymbol{\theta}(\cdot))$ denote the objective function (which will be a cost function in this case) and the constraint function, respectively. Then the general form of the functional optimization problem is as follows:

$$\min_{\boldsymbol{\theta}(\cdot)} \frac{1}{M} \sum_{k=1}^{M} C(\bar{\boldsymbol{\mu}}_k; \boldsymbol{\theta}(\cdot)) \tag{6.3}$$

$$\text{s.t. } F(\bar{\boldsymbol{\mu}}_k; \boldsymbol{\theta}(\cdot)) \geq \bar{\boldsymbol{\mu}}_k, k \in \{1, ..., M\} \tag{6.4}$$

where M is the number of time slots over which the optimization is performed. Thus, $\frac{1}{M}\sum_{k=1}^{M} C(\bar{\boldsymbol{\mu}}_k; \boldsymbol{\theta}(.))$ is the time average of the cost function.

Finding the optimal *function* $\boldsymbol{\theta}(.)$ by directly solving the functional optimization program in (6.3) and (6.4) is particularly difficult, if it is possible at all. In order to make the optimization program computationally viable, the second key idea in this section is to quantize the vector $\bar{\boldsymbol{\mu}}$ of effective traffic demands over all the links in the network, with a vector quantizer, which we denote as $Q(\bar{\boldsymbol{\mu}}) = \boldsymbol{y}_j$ with $j \in \{1, ..., L\}$. Here, the vector $\boldsymbol{y}_j$ is the output of the quantization, and L is the number of possible outputs. The partition for $\boldsymbol{y}_j$ is given as $\mathcal{R}_j = \{\bar{\boldsymbol{\mu}} \in (\mathcal{R}^+)^N : Q(\bar{\boldsymbol{\mu}}) = \boldsymbol{y}_j\}$. The quantization outcomes and the partition can be chosen properly, based on the statistical characteristics of $\bar{\boldsymbol{\mu}}$ and the computational power available[1]. We form a codebook for the output of the quantizer, denoted by $\mathcal{C}_{\bar{\boldsymbol{\mu}}} = \{\boldsymbol{y}_j\}$, with $|\mathcal{C}_{\bar{\boldsymbol{\mu}}}| = L$. For simplicity, we write $P[\bar{\boldsymbol{\mu}} \in \mathcal{R}_j]$ as ψ_j. Let $y_j^{(i)}$ be the ith element of the vector

[1]A thorough analysis of vector quantizer $Q(\bar{\boldsymbol{\mu}})$ is beyond the scope of this thesis, but will be studied in our future work.

$\boldsymbol{y}_j \in (\mathcal{R}^+)^N$. Let U^Q denote the total number of output levels, namely, the number of elements of the set $\{y_j^{(i)} | 1 \le j \le L, 1 \le i \le N\}$. We shall sometimes switch to a single-dimensional index notation for $y_j^{(i)}$, and denote the lth output level by $y[l]$, $1 \le l \le U^Q$. Define the vector of quantized design functions by $\overrightarrow{\boldsymbol{\theta}} \equiv [\boldsymbol{\theta}(y[1])^T, \boldsymbol{\theta}(y[2])^T, ..., \boldsymbol{\theta}(y[l])^T, ..., \boldsymbol{\theta}(y[U^Q])^T]^T$. Then, the quantized version of the optimization program can be written as

$$\min_{\overrightarrow{\boldsymbol{\theta}}} \sum_{\boldsymbol{y}_j \in \mathcal{C}_{\bar{\boldsymbol{\mu}}}} C(\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}) \cdot \psi_j \tag{6.5}$$

$$\text{s.t. } F(\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}) \ge \boldsymbol{y}_j, j \in \{1, ..., L\} \tag{6.6}$$

A simple example is given in Fig. 6.2 to illustrate the typical scenario we consider and to show intuitively the construction of a traffic pattern codebook: node 1, node 2 and node 3 are deployed as shown in the figure. The traffic links are shown as arrows and the traffic demands on each link, $\mu_1(t)$, $\mu_2(t)$ and $\mu_3(t)$, as functions of time, are also shown below the network. For simplicity, we assume that the traffic demands on all the links are periodic with period $T = 20$. Within each period T, 200 frames at node 1, and 100 frames at node 2 are generated at $t_1 = kT + 1$ and at $t_2 = kT + 7$, respectively. The maximal allowed delays D_1 and D_2 are both equal to 10. With the above assumptions, the keys of the effective

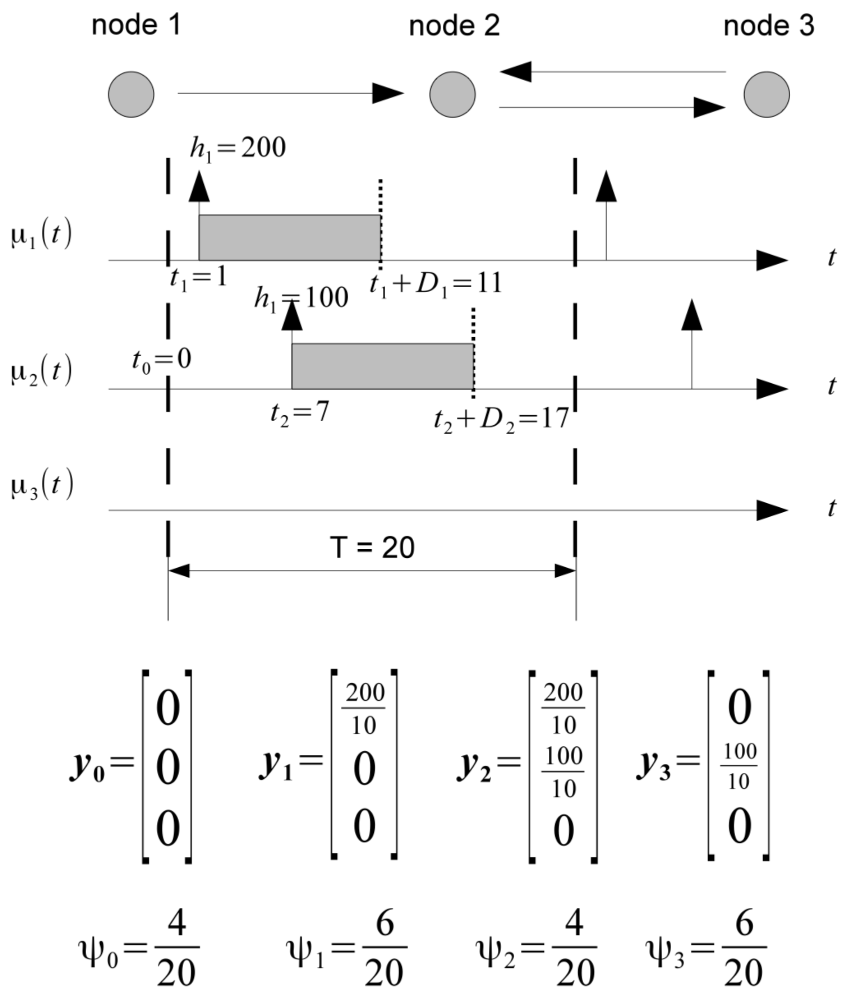

node 1
node 2
node 3
$h_1=200$
$\mu_1(t)$
$t_1=1$
$t_1+D_1=11$
$h_1=100$
$\mu_2(t)$
$t_0=0$
$t_2=7$
$t_2+D_2=17$
$\mu_3(t)$
t
T = 20
$y_0=\begin{bmatrix}0\\0\\0\end{bmatrix}$
$y_1=\begin{bmatrix}\frac{200}{10}\\0\\0\end{bmatrix}$
$y_2=\begin{bmatrix}\frac{200}{10}\\\frac{100}{10}\\0\end{bmatrix}$
$y_3=\begin{bmatrix}0\\\frac{100}{10}\\0\end{bmatrix}$
$\psi_0=\frac{4}{20}$
$\psi_1=\frac{6}{20}$
$\psi_2=\frac{4}{20}$
$\psi_3=\frac{6}{20}$

Figure 6.2: A simple example of the "traffic pattern codebook"

traffic demand codebook, $\boldsymbol{y}_j$, $j = \{0, 1, 2, 3\}$ as well as the associated probability distribution, ψ_j, $j = \{0, 1, 2, 3\}$, can be calculated as shown in Fig. 6.2.

6.3 MAC Protocol Generation Model

We use a MAC protocol model that is similar to those in [47] and [48]: each node takes an action from the set of actions denoted by $\mathcal{A} = \{n, d, c, f, a\}$ (each action occupies one time slot), where n means "listen to the channel", d means "send data frame", c means "send control frame", f means "send control frame and require acknowledgment for the control frame from the destined receiver," and a means "send acknowledgment". Based on these action definitions, we define the "knowledge state", $\mathcal{S}$, as the control information that is "owned" by a node, namely, the control information frames it has sent and the control information frames that is has received in the last W slots. (As in [47] and [48], W is the control information lifetime, namely, how long the effect of each control frame was designed to last.) The set of knowledge states of a node is denoted by $\mathcal{S} = \{\phi, \phi_0, s_c, s_f, s_a, q_c, q_f, q_a, l_c, l_f, l_a\}$, where ϕ is the null state, in which no control information has been sent or received by the node during the last W slots but with data waiting in the network layer buffer to be sent, and ϕ_0, a newly introduced state which has the same definition as ϕ except that there is *no* data in the network layer buffer to be sent; s_c, s_f, s_a mean that c, f or a, respectively, has just been sent in the last slot; q_c, q_f and q_a mean that c, f or a, respectively, has just been received in the last slot and the frame was destined to this node; l_c, l_f and l_a mean that c, f or a, respectively, has been received in the last slot but was destined to another node. (A more detailed explanation of the model can be found in [47] and [48].)

We define the design functions (a.k.a. decision functions), $\theta_n(\cdot)$, $\theta_d(\cdot)$, $\theta_c(\cdot)$, $\theta_f(\cdot)$ ($\sum \theta(\cdot) = 1$) as the probability, $\mathrm{P}(x|s)(\cdot)$, with which the node chooses an action $x \in \mathcal{A}$, upon arriving at one of the states, $s \in \mathcal{S}$, as a function of the effective traffic demand at the current time at that node. As was described for the general case in the previous section, we denote by $\boldsymbol{\theta}(\cdot) = [\theta_n(\cdot), \theta_d(\cdot), \theta_c(\cdot), \theta_{ca}(\cdot)]^T$ the vector of design functions, to represent all the design functions collectively. The transition rules of this model are summarized in Table 6.1. [2]

In this chapter, we pick a particular objective function of interest, the average transmission power, which we shall minimize. In order to obtain the optimization program expressions, we utilize the *Symbolic Monte Carlo* method [47] to explore the global state space of the network. Let C, R and T denote the power consumption of a node, the network throughput, and the length of a cycle, re-

[2]Note that this table is not a complete transition table for the finite state machine (FSM) of a node; for simplicity, it describes only the transition rules for phase 2 of a 2-phase FSM (see [47]). The phase 1 transition rules (from actions to next state) are straightforward, and are not shown.

state s	action x	$P(x\|s)$	timer $w(x\|s)$
ϕ	n, d, c, f	$\theta_n(\cdot)$, $\theta_d(\cdot)$, $\theta_c(\cdot)$, $\theta_f(\cdot)$	1, 1, 1, 1
ϕ_0	n	1	1
s_c	d	1	W
s_f	n	1	1
s_a	n	1	W
q_c	n	1	W
q_f	a	1	1
q_a	d	1	W
l_c	n	1	W
l_f	n	1	1
l_a	n	1	W

Table 6.1: Transition rules for a node upon arriving in state s

spectively, where a "cycle" is defined as the time between two subsequent visits to the same recurrent state. Let F_r and F_c denote the successful transmissions per cycle and the energy consumed by a node per cycle, respectively. We collect symbolic terms, and accumulate the symbolic expressions for the three metrics of interest that help us compute the average throughput $\mathrm{E}[R]$ and the average power consumption $\mathrm{E}[C]$. Thus, the three metrics of interest are $\mathrm{E}[F_r]$, $\mathrm{E}[F_c]$ and $\mathrm{E}[T]$. The long-term average throughput of the network and power consumption of a node can be computed as $\mathrm{E}[R] = \mathrm{E}[F_r]/\mathrm{E}[T]$ and $\mathrm{E}[C] = \mathrm{E}[F_c]/\mathrm{E}[T]$. The key aspect of the Symbolic Monte Carlo method is that the whole state space need not be explored; this would be a computationally intensive task even for a small network with multiple neighborhoods. Instead, the state space is "sampled" by running a symbolic Monte Carlo simulation in which symbolic expressions are collected via only the sampled routes through the state space. In the end, an approximation of the objective function is obtained. Finally, after the objective function and the constraints (based on the optimization program in (6.5) and (6.6)) have been computed by the Symbolic Monte Carlo method, the resulting non-linear optimization program

$$\min_{\overrightarrow{\boldsymbol{\theta}}} \sum_j \mathrm{E}[C|\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}] \cdot \mathrm{P}[\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}] \tag{6.7}$$

$$\text{s.t. } \mathrm{E}[R|\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}] \geq \boldsymbol{y}_j, \forall j \in \{1, ..., L\} \tag{6.8}$$

with

$$\mathrm{E}[C|\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}] = \frac{\mathrm{E}[F_c|\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}]}{\mathrm{E}[T|\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}]} \tag{6.9}$$

$$\mathrm{E}[R|\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}] = \frac{\mathrm{E}[F_r|\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}]}{\mathrm{E}[T|\boldsymbol{y}_j; \overrightarrow{\boldsymbol{\theta}}]} \tag{6.10}$$

(under the further constraints that all the probabilities are between 0 and 1, and sum to 1 over the set of actions) is solved by using the *openopt* package available for Python.

6.4 Simulations

The simulation set-up is as follows: 4 nodes are deployed as shown in Fig. 6.3, with the shown neighbor relationships. We set the control lifetime $W = 5$. We assume that each node generates the same type of bursty traffic with the same distribution. In this simulation, we assume a convergecast scenario: All the traffic converges onto node 2; node 0 and node 3 generate data. Node 0 sends to node 2 through a relay node 1. Relay node 1 generates no traffic of its own. In order to simplify the problem while keeping its essential features, the following assumptions are made: (1) Each burst has the same deterministic number of frames. (2) The maximal allowed delay on each link is $D = 1$ second, which is the same for all the traffic streams. (3) Each node is equipped with only one half-duplex radio. (4) The radio on each node is able to finish transmission/reception of a maximum number c of bursts within the average target delay D seconds, i.e., the radio capacity is c (bursts/target delay). In our simulations, we set $c = 20$. (5) The arrival process of bursts at each node is an independent Poisson process with arrival rate λ, i.e., $\lambda_1 = \lambda_2 = \lambda$ (arrivals/second).

In order to evaluate the proposed automated MAC generation framework for dynamic traffic, we built a simulation environment in Python, incorporated with three main packages: *pylab*, which enables a MATLAB-like working environment;

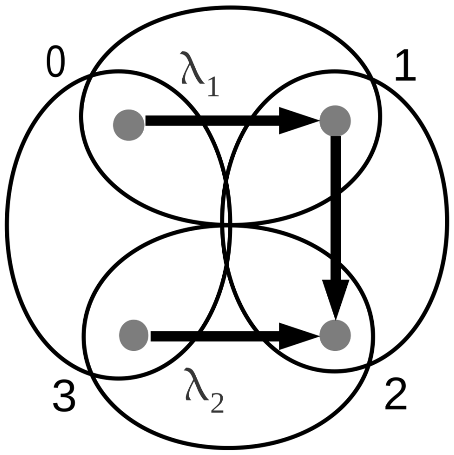


Figure 6.3: Simulation scenario

networkx, which provides powerful graph-related functions; and *openopt*, which is able to solve the resulting non-linear optimization problem efficiently (in this simulation we use the NLP non-linear solver within the *openopt* package). All the simulations were run on a Dell Studio 540 Mini-Tower, with Intel Core 2 Quad Processor Q9550 (2.83GHz, 1333MHz FSB and 12MB cache) with no other concurrent compute-intensive processes.

The simulation structure is illustrated in Fig. 6.4: the Dynamic Traffic Generator randomly generates bursty traffic according to the Poisson distribution. The traffic Codebook Generator takes in the bursty traffic and builds the corresponding traffic codebook for this dynamic traffic. Based on the given MAC model, symbolic Monte Carlo simulation explores the design space and generates

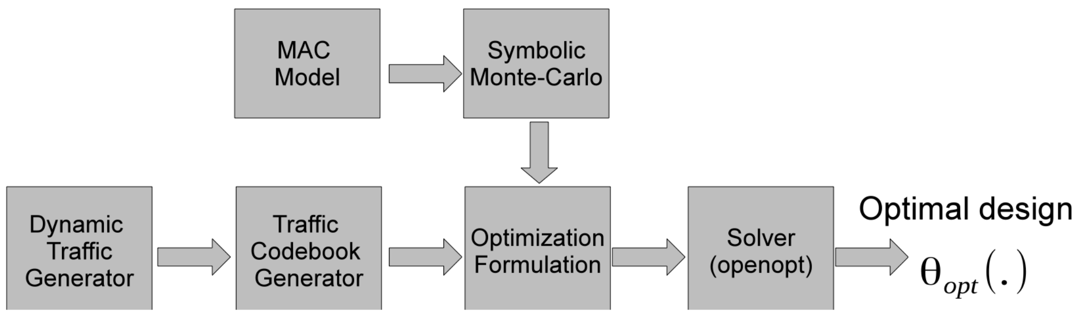


Figure 6.4: Simulation structure

approximate symbolic expressions which are needed for formulating the optimization. With the expressions from the Symbolic Monte Carlo method and the traffic codebook from the Codebook Generator, the Optimization Formulation stage generates a non-linear optimization program which can be solved by the NLP Solver provided by the *openopt* package. At the end, the values of the optimal design functions are output by the Solver.

In this simulation, we use the “two-stage” symbolic Monte Carlo method that was described in the simulation section of the Chapter 5.

In Fig. 6.5, we show the optimized design functions $\{\theta_i\}(\cdot)$ as a function of the effective traffic demand, for the low traffic load network (arrival rate $\lambda = 0.1$, characterizing the traffic generator, and is constant during the simulation). We vary the local effective traffic demand from 0.00 to 0.10. We can see that when the node has no frames to send, the node chooses to listen (denoted by action

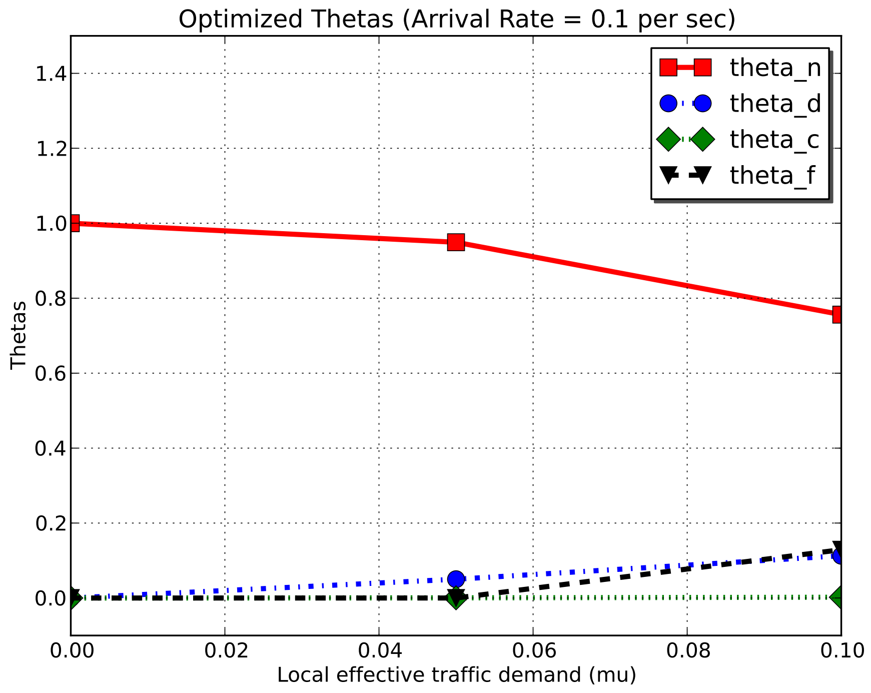


Figure 6.5: Optimized design functions

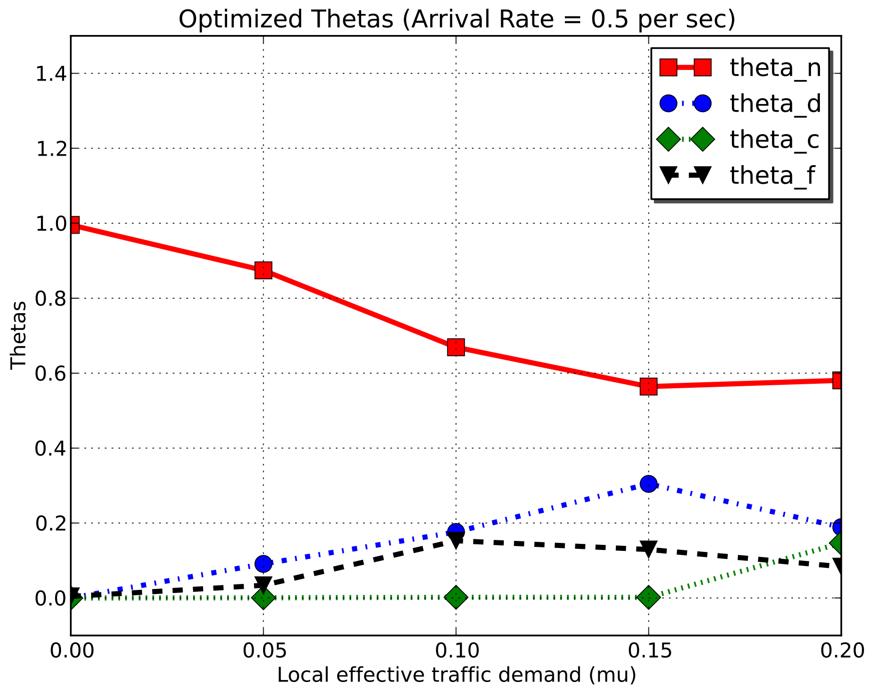


Figure 6.6: Optimized design functions

n) almost all the time ($\theta_n \approx 1.0$). (There is a small discrepancy, not completely visible in the graph, which is due to the randomness inherent in the generation of the approximate objective function in the Symbolic Monte Carlo method.) As the traffic demand increases, θ_n decreases and the other θs decrease, which means that the node chooses to send frames more often in order to meet the average traffic demand. When the local effective traffic demand reaches 0.10, instead of choosing only one of the sending methods, i.e. d, c or f, the optimal MAC protocol chooses a *combination* of them, with different probabilities. These values in the figures characterize the optimal MAC protocol for dynamic traffic, which can be stored at each node during the operation, and do not need to be recomputed.[3] Each time that the local traffic load changes, the node needs to look up only θ in its lookup table and set the optimal θs for the current traffic condition at almost no computational cost. Furthermore, this "hybrid" protocol (which uses d, c, and f) results from the optimization, and is very difficult to be found as a hand-designed protocol.

In Fig. 6.6, we show the optimized design functions $\{\theta_i\}(\cdot)$, as a function of the effective traffic demand, for the high traffic load (arrival rate $\lambda = 0.5$). We vary the effective traffic demand from 0.00 to 0.20. In this figure, we can see a trend

[3]We note that in this thesis, even though the traffic conditions are dynamic, the parameters of the random processes that generate them (only λ in the current simulation) are assumed to be static.

similar to that in Fig. 6.5; namely, the node listens less and sends more frames into the channel as the traffic demand increases. However, because the network is more loaded ($\lambda = 0.5$), the node chooses to send d, c and f more aggressively (for the same traffic demands, the values of θ_d, θ_c and θ_f are larger than in Fig. 6.5).

When we compare Fig. 6.5 and Fig. 6.6 more carefully, we can see that they are very similar in the range from $\mu = 0.0$ to $\mu = 0.1$, except for a slightly smaller θ_n and slightly larger θ_d and θ_f in Fig. 6.6 [4]. This recurring pattern in the two figures implies that the "irregularity" of the theta function $\theta(\cdot)$ does not come from randomness, but rather from the complex nature of this optimization problem.

In Fig. 6.7, we display the average power consumption of a node under different arrival rates, where the arrival rate λ ranges from 0.0 to 0.7. We see that the average power consumption increases as the traffic load increases, but the average power consumption is not a linear function of the traffic load because as λ increases, the nodes have different optimal combinations of actions and thus more complex power consumption patterns, which are difficult to derive analytically.

In Fig. 6.8, the computational complexities (characterized by the execution times) for the four major parts of the framework are shown. It can be seen that

[4]In Fig. 6.5, the local effective traffic demand ranges only between 0.0 and 0.10 because the generated traffic codebook does not contain entries with local traffic higher than 0.1; thus, no design function $\theta(\cdot)$ can be evaluated beyond 0.1. Similarly, for Fig. 6.6, the design function $\theta(\cdot)$ is not evaluated for values greater than $\bar{\mu} = 0.2$.

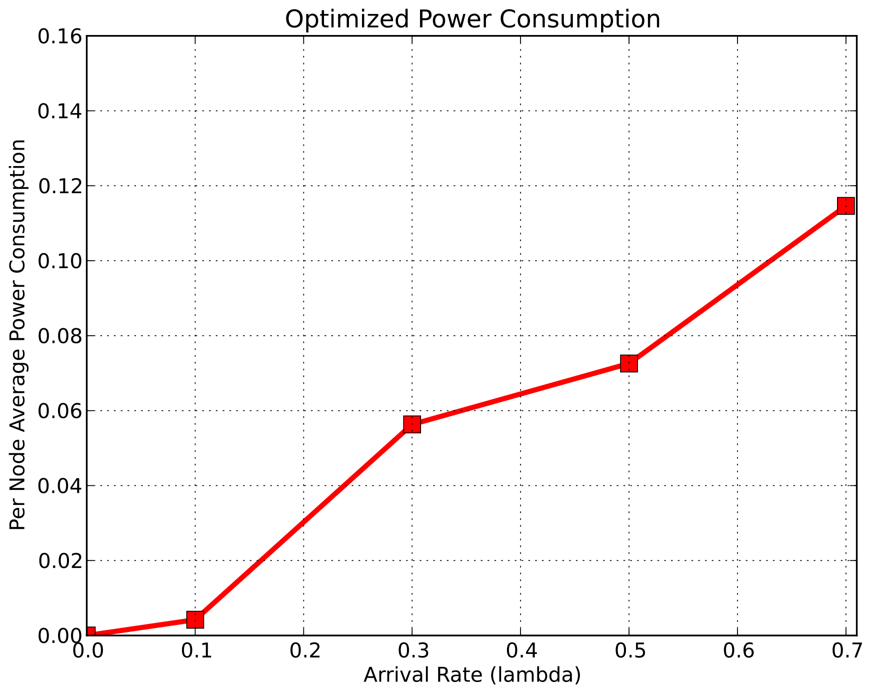


Figure 6.7: Optimized power consumption

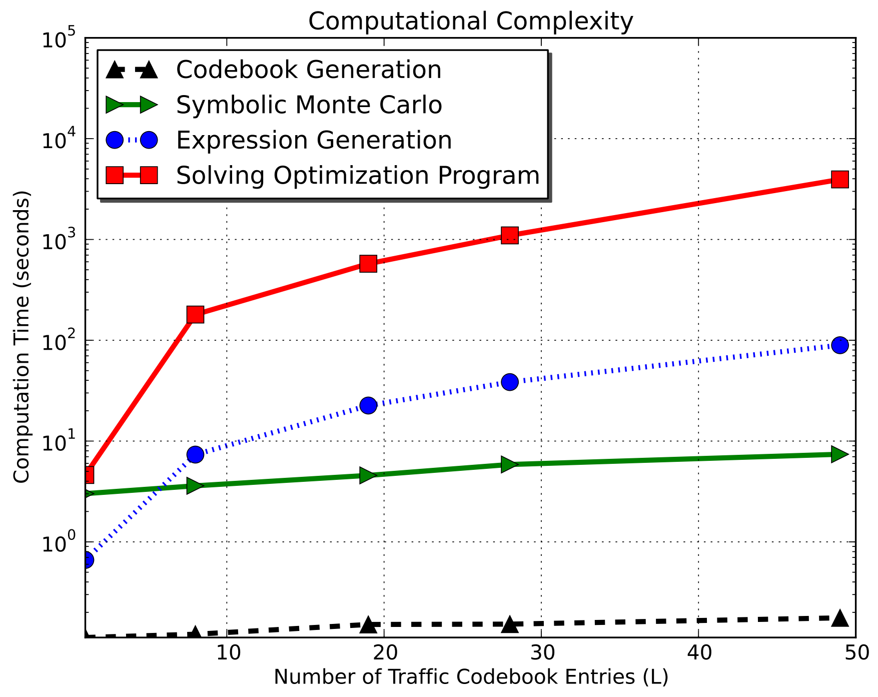


Figure 6.8: Computational complexity

solving the optimization program accounts for most of the complexity, which is different from the result obtained in [46] and [47], where the computation time for the Symbolic Monte Carlo method and expression generation was dominant. This is due to the fact that the objective function in this chapter is a weighted sum of multiple *identical* symbolic expressions (one for each value of effective traffic demand), which does not incur much additional computational time for the *generation* of symbolic expressions, but which does result in much higher computational complexity for solving the resulting optimization program.

6.5 Summary

We have demonstrated the viability of automated MAC protocol generation under dynamic traffic conditions for multiple neighborhoods, in the presence of acknowledgments. The functional optimization program that we generate has design functions, which are the probabilities with which a node takes a given action in a given knowledge state, and for a given current effective traffic demand at that node. By discretizing this program, we have presented a method by which an optimal MAC protocol can be generated off-line.

Chapter 7

Conclusions and Future Work

The conclusions of this work can be summarized as follows:

1. The incorporation of the impact of control information into optimization enables optimization programs to generate optimal protocols.

2. Protocol optimality, that takes into account the cost and impact of control information, is a key concept for the design of protocols at all layers of the network protocol stack.

3. A protocol model in which decisions branch probabilistically for each node at each slot is a general framework by which to subsume structurally different protocols under a single umbrella, and optimize over them.

4. Symbolic Monte Carlo simulation is an effective method in generating an approximation to the objective function on a complex state space.

5. For the Medium Access Control (MAC) Layer, Symbolic Monte Carlo simulation can generate an optimal protocol, off-line, for small-scale multi-neighborhoods, for dynamic topologies, and for dynamic traffic conditions.

This work solves the following problems effectively:

1. The MAC Layer problem to find an optimal protocol for a single neighborhood for N nodes in polynomial-time in N.
2. The MAC Layer problem to find an optimal protocol for small-scale multi-neighborhoods (using Symbolic Monte Carlo simulation).
3. The MAC Layer problem to find an optimal protocol for a mobile network, which has a dynamic topology (using Symbolic Monte Carlo simulation, and information on the frequencies with which local topologies occur).
4. The MAC Layer problem to find an optimal protocol under dynamic traffic conditions, modeled by local traffic demand at a node.

The following are some indicated future directions by which the methods in this thesis can be made useful so that they will have an impact on networks:

1. Development of a "front end" for automated protocol generation, which is comprised of a high-level language in which these optimizable network

protocols can be expressed, and a Graphical User Interface (GUI) on which the different design specifications can be given.

2. Incorporation of this framework into Software Defined Networking (SDN), which virtualizes the network and eases network management issues. These optimized protocols can then be incorporated into a Software Defined Networking framework, such as OpenFlow [49].

3. Development of "back end" technologies that are polynomial-time in N even for multi-neighborhood topologies. Even though such scaling is not required for MAC protocol design that is local, it will be required if this framework is to have success within Software Defined Networking (SDN).

Bibliography

[1] H. Hartenstein and K. P. Laberteaux, “A tutorial survey on vehicular ad hoc networks,” in *IEEE Communications Magazine* Volume: 46, Issue: 6, June 2008.

[2] R. C. Daniels and R. W. Heath, “60 GHz wireless communications: emerging requirements and design recommendations,” in *IEEE Vehicular Technology Magazine*, Volume: 2, Issue: 3. Sept. 2007.

[3] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks (5th Edition)*, Prentice Hall, Oct 7, 2010

[4] J. D. Day and H. Zimmermann, “The OSI reference model,” in *Proc. of the IEEE* Volume:71, Issue:12, Dec. 1983.

[5] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, pp. 379-423 and 623-656, July and Oct., 1948.

[6] A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communication*, Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA 1995.

[7] R. v. Nee, *OFDM for Wireless Multimedia Communications*, Artech House, Inc. Norwood, MA, USA, 2000

[8] I. Demirkol, C. Ersoy and F. Alagoz,“MAC protocols for wireless sensor networks: a survey,” in *IEEE Communications Magazine*, Volume:44, Issue: 4, 2006.

[9] D. Camps-Mur and A. Garcia-Saavedra,“Device-to-device communications with Wi-Fi Direct: overview and experimentation,” in *Wireless Communications, IEEE*, Volume:20, Issue: 3, June 2013.

[10] J. Jubin and J. D. Tomow, “The DARPA packet radio network protocols,” in *Proc. of the IEEE*, Volume: 75, Issue: 1, Jan. 1987.

[11] P. E. Allen and D. R. Holberg, *CMOS Analog Circuit Design*, Oxford University Press, USA, Jan. 2002.

[12] B. Bing, *Wireless Local Area Networks: the New Wireless Revolution*, Wiley, 2002.

[13] B. Hofmann-Wellenhof, H. Lichtenegger and J. Collins, *Global Positioning System: Theory and Practice*, Springer, Wien (Austria), 1993.

[14] S. L. Goff, A. S. Glavieux and C. Berrou, "Turbo-codes and high spectral efficiency modulation," in *Proc. IEEE ICC '94*, 1994.

[15] C.S. Burrus, R.A. Gopinath and H. Guo, *Introduction to Wavelets and Wavelet Transforms: a Primer*, Prentice Hall, 1997.

[16] Y. Xiao, J. Rosdahl, "Throughput and delay limits of IEEE 802.11," in *IEEE Communications Letters*, 2002, vol. 6, no. 8, pp. 355-357.

[17] E. H. Ong, J. Kneckt, O. Alanen and Ch. Zheng, "IEEE 802.11ac: enhancements for very high throughput WLANs," in *2011 IEEE 22nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, 2011.

[18] V. Shrivastava, S. Rayanch, J. Yoon, and S. Banerjee, "802.11n under the microscope," in *Proc. IMC*, Oct. 2008.

[19] K. Pelechrinis, T. Salonidis, H. Lundgren, N. Vaidya, "Analyzing 802.11n performance gains," in *Proc. ACM MobiCom* 2009.

[20] V. Visoottiviseth, T. Piroonsith, and S. Siwamogsatham, "An empirical study on achievable throughputs of IEEE 802.11n Devices," in *Proc. WinMee*, June 2009.

[21] S. Shakkottai, T.S. Rappaport, “Cross-layer design for wireless networks,” in *IEEE Communications Magazine*, 2003, vol. 41 , issue 10, pp. 74 - 80.

[22] V. Srivastava, M. Motani, “Cross-layer Design: a survey and the road ahead,” in *IEEE Communications Magazine*, 2005, vol. 43, issue 12, pp. 112 - 119.

[23] E. Setton, T. Yoo, X. Zhu, A Goldsmith, “Cross-layer design of ad hoc networks for real-time video streaming,” in *IEEE Wireless Communications*, 2005, vol. 12 , issue 4, pp. 59 - 65, 2005.

[24] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, “Layering as optimization decomposition: a mathematical theory of network architectures,” in *Proceedings of IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[25] J.W. Lee, A. Tang, J. Huang, M. Chiang and A.R. Calderbank, “Reverse-engineering MAC: a non-cooperative game model,” in *IEEE Journal on Selected Areas in Communications*, Vol.25, No.6, August 2007.

[26] M. Casado, T. Koponen, S. Shenkder and A. Tootoonchian, “Fabric: a retrospective on evolving SDN,” in *Proceedings of the first workshop on Hot topics in software defined networks HotSDN'12*, New York, 2012.

[27] D. D. Gajski and R. H. Kuhn, “Guest editors' introduction: new VLSI tools,” in *IEEE Computer* Volume:16, Issue: 12, 1983.

[28] D. Macmillen, R. Camposano, D. Hill and T. W. Williams, "An industrial view of electronic design automation," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* Volume:19, Issue: 12, Dec 2000.

[29] N. Abramson, "Development of the ALOHANET," in *IEEE Transactions on Information Theory*, 1985.

[30] M. O. Rabin, "Probabilistic automata," in *Information and control*, Elsevier, 1963.

[31] G. Bianchi, L. Fratta and M. Oliveri, "Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LANs," in *IEEE Personal, Indoor and Mobile Radio Communications (PIMRC'96)*, 1996.

[32] C. Z. Mooney, *Monte Carlo Simulation*, Sage Publications, 1997.

[33] A. Perrig and D. Song, "A first step towards the automatic generation of security protocols," in *Symposium on Network and Distributed Systems Security (NDSS)*, Feb. 2000.

[34] D. Song, A. Perrig, and D. Phan, "AGVI - automatic generation, verification, and implementation of security protocols," in *Computer Aided Verification: Proc. 13th Intl. Conf. (CAV 2001)*, G. Berry, H. Comon, and A. Finkel, Eds.,

Lecture Notes in Computer Science, Vol. 2102, Berlin, Germany: Springer-Verlag, 2001, pp. 241-245.

[35] J. W. Lee, M. Chiang, and R. A. Calderbank, “Utility-optimal random-access control,” in *IEEE Transactions on Wireless Communications*, vol. 6, no. 7, pp. 2741-2751, July 2007.

[36] V. Rodoplu, A. Aminzadeh Gohari, and W. Tang, “Towards automated design of MAC protocols for underwater wireless networks,” in *Proc. of the 3rd ACM International Workshop on Underwater Networks (WUWNET’08)*, Sept. 2008.

[37] V. Rodoplu, and A. Aminzadeh Gohari, “Challenges: automated design of networking protocols,” in *Proc. of ACM International Conference on Mobile Computing and Networking (MobiCom’08)*, Sept. 2008.

[38] S. C. Ergen, P. Di Marco, C. Fischione, “MAC protocol engine for sensor networks,” in *Proc. of the IEEE Global Communications Conference (Globecom’ 09)*, Nov. 2009.

[39] A. Farago, A. D. Myers, V. R. Syrotiuk and G. V. Zaruba, “Meta-MAC protocols: automated combination of MAC protocols to optimize performance for unknown conditions,” in *IEEE Journal on Selected Areas in Communications*, Vol.18, No.9, Sept. 2000, pp. 1670.

[40] A. Farago, A.D. Myers, V.R. Syrotiuk and G.V. Zaruba, "Meta-MAC Protocols: automatic combination of MAC protocols to optimize performance for unknown conditions," in *IEEE Journal on Selected Areas in Comminications,* Vol. 18, No. 9, Sept 2000.

[41] C. Doerr, M. Neufeld, J. Fifield, T. Weingart, D.C. Sicker and D. Grunwald, "MultiMAC - an adaptive MAC framework for dynamic radio networking," in *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*, 2005.

[42] H. S. Lichte, S. V. and H. Karl, "Automated development of cooperative MAC protocols a compiler-assisted approach," in *Mobile Networks and Applications*, vol. 15, no. 6, pp. 769-785, DOI: 10.1007/s11036-009-0210-5.

[43] J. Ansari, X. Zhang and Petri Maehoenen, "A compiler assisted approach for component based reconfigurable MAC design," in *The 10th IFIP Annual Mediterranean Ad Hoc Networking Workshop*, 2011.

[44] J. Ansari, X. Zhang, A. Achtzehn, M. Petrova and P. Maehoenen, "Decomposable MAC framework for highly flexible and adaptable MAC realizations," in *New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on*, 2010.

[45] J. Ansari, X. Zhang, A. Achtzehn, M. Petrova and P. Maehoenen, “A flexible MAC development framework for cognitive radio systems,” in *IEEE Wireless Communication and Networking Conf.*, Quintana-Roo, Mexico, Mar. 2011.

[46] J. Zhen, F. Brewer, V. Rodoplu,“A methodology for optimal MAC protocol generation: case study of a synchronous MAC channel,” in *Proc. of the IEEE Global Communications Conf. (Globecom' 10)*, Dec. 2010.

[47] J. Zhen, F. Brewer, V. Rodoplu,“Automated MAC protocol generation with multiple neighborhoods and acknowledgments based on Symbolic Monte Carlo simulation,” in *Proc. of the IEEE Global Communications Conf. (Globecom' 11)*, Dec. 2011.

[48] J. Zhen, V. Rodoplu,“Automated MAC protocol generation for dynamic topologies,” in *Proc. of the IEEE Global Communications Conf. (Globecom' 12)*, Dec. 2012.

[49] N. McKeown, T. Anderson and H. Balakrishnan,“OpenFlow: enabling innovation in campus networks,” in *ACM SIGCOMM Computer Communication Review*, Volume 38, Issue 2, April 2008.