Scheduled Medium Access Control in Mobile Ad Hoc Networks

by

Jonathan Lutz

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2013 by the
Graduate Supervisory Committee:

Charles J. Colbourn, Co-Chair
Violet R. Syrotiuk, Co-Chair
Goran Konjevod
Errol L. Lloyd

ARIZONA STATE UNIVERSITY

December 2013

ABSTRACT

The primary function of the medium access control (MAC) protocol is managing access to a shared communication channel. From the viewpoint of transmitters, the MAC protocol determines each transmitter's persistence, the fraction of time it is permitted to spend transmitting. Schedule-based schemes implement stable persistences, achieving low variation in delay and throughput, and sometimes bounding maximum delay. However, they adapt slowly, if at all, to changes in the network. Contention-based schemes are agile, adapting quickly to changes in perceived contention, but suffer from short-term unfairness, large variations in packet delay, and poor performance at high load. The perfect MAC protocol, it seems, embodies the strengths of both contention- and schedule-based approaches while avoiding their weaknesses.

This thesis culminates in the design of a Variable-Weight and Adaptive Topology Transparent (VWATT) MAC protocol. The design of VWATT first required answers for two questions: (1) If a node is equipped with schedules of different weights, which weight should it employ? (2) How is the node to compute the desired weight in a network lacking centralized control? The first question is answered by the Topology- and Load-Aware (TLA) allocation which defines target persistences that conform to both network topology and traffic load. Simulations show the TLA allocation to outperform IEEE 802.11, improving on the expectation and variation of delay, throughput, and drop rate. The second question is answered in the design of an Adaptive Topology- and Load-Aware Scheduled (ATLAS) MAC that computes the TLA allocation in a decentralized and adaptive manner. Simulation results show that ATLAS converges quickly on the TLA allocation, supporting highly dynamic networks. With these questions answered, a construction based on transversal designs is given

for a variable-weight topology transparent schedule that allows nodes to dynamically and independently select weights to accommodate local topology and traffic load. The schedule maintains a guarantee on maximum delay when the maximum neighbourhood size is not too large. The schedule is integrated with the distributed computation of ATLAS to create VWATT. Simulations indicate that VWATT offers the stable performance characteristics of a scheduled MAC while adapting quickly to changes in topology and traffic load.

To my children:

Benjamin, Jack, Hannah, and Isaac.

It's done. What shall we do tomorrow?

And to Sarah,

my wife and best friend,

for everything, and then some.

ACKNOWLEDGEMENTS

There are many who have journeyed with me through these doctoral years, and I am deeply grateful to each.

Violet Syrotiuk and Charlie Colbourn, thank you for taking me on as your student. You taught me how to research, how to write, and what it means to be precise. At the start, I chose advisers rather than an area of study. It was a better choice than I knew at the time.

Thank you, Goran Konjevod and Errol Lloyd, for serving on my committee. I appreciate the time and input you provided.

I am grateful to my colleagues, past and present. Alfonso Íñiguez, James McDowell, Bill Oh, Brian Kaufman, Jesse Krigelman, Jay Robinette, and Paul Stoaks, thank you for humouring me with feigned interest in my research; but especially, I appreciate the engaging topics covering everything except that.

Having a full time job while being a student was never easy, however the load was lessened because of understanding and supportive supervisors: Tom Tkacik, Greg Schmidt, Mark Hemingway, William Clark, Scott Chuprun, and Bahaa Osman.

Two men have been undeservedly kind to me. Jeff LaVell and Steve Tugenberg, thank you for your wisdom, your kindness, and your long suffering. My respect and admiration for you is unparallelled.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

Once dominant, the role of the personal computer has declined in recent years, giving way to mobile devices such as smart phones and tablets. The increasing computational power of these devices, paired with the improved battery life and smaller size of newer laptops and convertibles [10, 48], blur the distinction between the personal computer and the mobile device. At the same time, the decreasing size and cost of electronic devices is enabling advancements in the areas of wearable computing [39] and home automation [38]. Ubiquitous (anywhere and everywhere) [93] computing is coming of age. Computing devices, once relegated to the home or office, have become tools, with lower profile yet more pervasive, assisting in day-to-day tasks. Implicit in this paradigm shift is the need for wireless connectivity between devices. As the number of connected devices grows, and the Internet of Things [11] becomes a reality, the task of efficient and flexible connectivity becomes more challenging.

Today, most wireless devices connect through a single-hop wireless link to a wired infrastructure, *i.e.*, Wi-Fi access point or cell tower. Looking forward, with the number of connected devices expected to reach 50 billion by the year 2020 [26], this approach to wireless connectivity may be insufficient to support the massive demand for communication. In light of this, future network architectures may need to (1) support multi-hop wireless communication to lessen the requirements for fixed infrastructure, (2) self organize to simplify the design and deployment of the network, and (3) adapt in real time to accommodate devices entering, leaving, or moving through the network. These three features are properties intrinsic to a mobile ad hoc network (MANET). While future networks are likely to retain a wired component,

mobile ad hoc networks have the potential to augment the network and achieve the desired goals [13, 15].

A MANET is a collection of autonomous nodes, each equipped with a wireless transceiver, power supply, and host processor. Without centralized control, the nodes self organize to form a network, relying only on the availability of electromagnetic spectrum for communication. Each node originates and receives network traffic, but can also act as a router to facilitate multi-hop communication between otherwise disconnected nodes. The network accommodates mobile nodes by automatically adjusting to changes in the topology. Although design for mobile ad hoc networking affects all layers of the network stack, it represents a particular challenge to the task of medium access control. The focus of this thesis is on medium access control over a single channel with access multiplexed in the time domain.[1]

The medium access control (MAC) layer is responsible for managing access to a shared communication channel. From the perspective of a transmitter, this entails answering the question: When can I transmit next? In a wireless network, answering this question becomes less than straightforward because of a number of complications:

1. *The hidden terminal problem* [86]: Due to signal attenuation over the wireless medium, nodes can form disconnected topologies. In these topologies, it is possible for a pair of nodes to be out of range of one another and yet compete for the channel at a common receiver. These nodes remain "hidden" from each other, making the traditional carrier sense approach to collision avoidance [44] ineffective.

[1] Alternatives are Code Division Multiple Access and Frequency Division Multiple Access [72] which are not discussed here.

2. *The need for distributed computation*: To qualify as "ad hoc," the decision to transmit must be made in a distributed fashion without the aid of centralized control.

3. *The chicken and the egg*: Any distributed method for deciding when to transmit must not require extensive communication between nodes. The MAC layer enables communication; it cannot presume use of the service it is expected to provide.

4. *Interference from broadcast transmissions*: Due to the broadcast nature of wireless communication, transmissions interfere with the intended receivers as well as all receivers within transmission range. The decision to transmit must consider the potential for collisions beyond the intended receiver.

5. *The need for adaptation*: The decision to transmit must accommodate changes in both topology and local demand for the channel.

Generally speaking, MAC protocols can be categorized according to their underlying channel allocation mechanism, specifically whether they are contention-based or schedule-based. Nodes in contention-based protocols such as Aloha [9] transmit messages whenever there is one to send. If unsuccessful, an attempt to retransmit is made at a later time. MACAW [14] and IEEE 802.11 [45] build on Aloha by increasing the expected time between retransmissions following each unsuccessful attempt. In contrast, schedule-based protocols rely on predetermined schedules to coordinate access to a common channel. A primitive example of scheduled allocation is found in Time Division Multiple Access (TDMA) where time is divided into frames, then slots; each node is assigned a unique transmission slot in a frame. Other examples include the family of (k, v)-scheduled protocols in which nodes are assigned

k transmission slots from frames of v slots. The k slots can be selected at random [29] or can be chosen deterministically. Another example, topology transparent schedules [16, 30, 83], designed to be independent of the detailed network topology, rely on only two design parameters: N, the number of nodes in the network, and D_{max}, the maximum supported neighbourhood size. These schedules guarantee each node a collision-free transmission opportunity from each of its neighbours at least once per frame, provided the node's neighbourhood size does not exceed D_{max}. Though the schedules are robust to violations to D_{max} [84], they do not adapt.

The pros and cons of contention-based and schedule-based MAC protocols appear to complement one another. Contention-based schemes are agile but fail to bound maximum delay or allocate channel capacity fairly between neighbouring nodes [14]. On the other hand, scheduled protocols exhibit stable delay characteristics and, in some cases, provide a maximum delay guarantee [16]. But unlike contention-based allocation, they adapt slowly, if at all, to changing network conditions. A desirable outcome is to combine the strengths of each type of protocol, the adaptability of contention-based schemes and the stable delay characteristics of schedule-based schemes. The contributions of this thesis work toward this goal.

1.2 Contributions

This thesis presents three primary contributions to the area of medium access control in mobile ad hoc networks: (1) The Topology- and Load-Aware (TLA) allocation is proposed and evaluated as a channel allocation strategy. (2) A distributed method for computing the TLA allocation is proposed and integrated into an Adaptive Topology- and Load-Aware Scheduled (ATLAS) MAC protocol. (3) A variable-weight topology transparent schedule is constructed and integrated into ATLAS to create the Variable-

Weight and Adaptive Topology Transparent (VWATT) MAC protocol. Contributions are further clarified in the following subsections.

1.2.1 The Topology- and Load-Aware Allocation

Most MAC protocols treat specific MAC challenges such as collision avoidance, maximization of spatial re-use, or minimization of delay. Some protocols target persistences, the percentage of time nodes are permitted to transmit, without regard to network topology. Regardless of whether it is contention-based, schedule-based, randomized, or deterministic, the MAC protocol directly influences persistences. Hence, we start with a more general question: At what persistence should a node be permitted to transmit given the network topology and demand for the communication channel? The answer to this question has a direct impact on network performance. The channel may be underutilized when persistences are too low. On the other hand, numerous collisions can result if persistences are too high. Even when long-term persistences are appropriate, high variation in persistence can degrade the quality of service by increasing the maximum packet delay. The goal of the MAC layer is to increase persistences when possible, but avoid over-provisioning. More specifically, persistences should satisfy the following properties:

1. No receiver is overrun, *i.e.*, the combined persistence of a neighbourhood is not too large.

2. No transmitter is given a persistence greater than it can use.

3. No transmitter is permitted to monopolize the channel, *i.e.*, persistences are distributed fairly among the transmitters in a neighbourhood.

4. Persistences are maximized subject to the constraints of the first three properties.

The lexicographic max-min allocation satisfies all four properties, reflecting desirable properties of channel access without limiting the MAC protocol employed. When applied to transmitter persistences, the lexicographic max-min allocation defines the Topology- and Load-Aware (TLA) allocation, emphasizing that persistences are selected to accommodate both topology and load.

The performance enabled by the TLA allocation is evaluated through simulation; results are compared against those of IEEE 802.11, a well-known point of reference. Simulations show the TLA allocation to achieve

- lower expected delay for all but the most lightly loaded networks,
- higher expected throughput,
- a dramatic reduction in dropped packets, and
- extremely low variation in delay and throughput.

Centralized and distributed methods for computing the TLA allocation are given. Although this initial distributed method does not adapt to changes in the network, it lays the framework for a fully adaptive computation, which is the second contribution of this thesis.

1.2.2 Adaptive Topology- and Load-Aware Scheduling

To be used as a channel allocation strategy, the TLA allocation must be computed in a decentralized and adaptive way. To achieve this, we propose an Adaptive Topology- and Load-Aware Scheduling (ATLAS) MAC protocol. In ATLAS, a distributed auction runs *continuously*. It piggybacks offers and claims onto existing network traffic to compute the TLA allocation. Each node's schedule is selected at random to realize

the persistence informed by its allocation. Schedules are updated whenever a change in topology or load results in a change in allocation. The slots of the schedule are grouped into frames. However, this is done only to reduce the variance in delay [29]; there is no need to wait for a frame boundary to update the schedule. While ATLAS is used here to compute TLA persistences, its underlying auction-based algorithm computes a solution to the general resource allocation problem. As such, it may prove useful in applications beyond medium access control.

The correctness of ATLAS is proven and its performance is evaluated through simulation. Results show the protocol to adapt quickly—converging on the TLA allocation within 0.1 seconds following most network changes—and to scale well to large multi-hop networks. The protocol's agility is further demonstrated in mobile networks where it maintains throughput, even when node speeds are high. ATLAS realizes impressively small variance in delay and throughput to implement a stable MAC environment over which TCP can support 4- and 5-hop flows, something that IEEE 802.11 networks struggle to do. While ATLAS realizes superior delay variance compared to IEEE 802.11, its randomized schedules do not provide a guaranteed maximum on delay. An adaptive MAC, capable of providing a delay guarantee, is the third contribution of this thesis.

1.2.3 Variable-Weight and Adaptive Topology Transparent Scheduling

The main obstacle to using topology transparent schedules is their inability to adapt to variations in traffic load and neighbourhood size. Existing schemes employ constant-weight schedules that allocate the same number of transmission slots, providing every node in the network with the same persistence. The result is an unnecessary constraint on throughput in neighbourhoods smaller than D_{max}, especially for nodes with large

traffic demands, and the loss of the delay guarantee in neighbourhoods larger than D_{max}.

Here, we propose variable-weight topology transparent scheduling as a means to achieve a guarantee on maximum delay *without* paying a high cost in throughput. We construct variable-weight schedules from transversal designs, provisioning each node in the network with multiple schedules, each with a distinct weight. Nodes dynamically select schedules with weights following the lexicographic max-min allocation, accommodating local topology and traffic load. The guarantee on maximum delay is maintained provided the weight of schedules is not too large. The schedule construction extends the constant-weight constructions of [16] and [83] by adding higher weight schedules. Surprisingly, the variable-weight schedules are included at no cost to frame length, D_{max}, or N.

The schedules are integrated into the Variable-Weight and Adaptive Topology Transparent (VWATT) MAC protocol, which employs a modified version of the distributed computation in ATLAS to compute lexicographic max-min schedule weights. The performance of VWATT is evaluated through simulation. Results show VWATT to improve throughput while maintaining the delay characteristics of constant-weight topology transparent schedules. Compared to ATLAS, VWATT reduces the maximum delay as well as the 75^{th} and 95^{th} percentiles in delay. Dropped packets are eliminated in networks whose neighbourhood size does not exceed D_{max}. Surprisingly, VWATT also achieves better throughput compared to ATLAS in all simulated networks except those sparsely loaded with large demands.

VWATT represents the first topology transparent MAC that adapts to both topology and load by selecting schedules with appropriate weight while maintaining a guarantee on maximum delay.

1.3 Organization

The remainder of this thesis is organized as follows. Chapter 2 reviews the necessary background material and surveys work related to medium access control in multi-hop wireless networks. Chapter 3 presents the TLA allocation evaluating it for use as a channel access strategy. The asynchronous and distributed computation of the TLA allocation via ATLAS is presented and evaluated in Chapter 4. The variable-weight topology transparent schedules are described and integrated into the VWATT MAC in Chapter 5. We conclude with Chapter 6, summarizing contributions and discussing potential avenues for future work. Supplemental material is provided in Appendix A and a list of publications is provided in Appendix B.

Much of the content in this thesis has been (or is in the process of being) published in journals or conference proceedings. Chapter 3 is contained in [64], Chapter 4 in [63], and Chapter 5 in [65]. The material presented in Appendix A.1 and Appendix A.2 appears in [61] and Appendix A.3 in [62].

Chapter 2

BACKGROUND

This chapter contains background material necessary to support the contributions in Chapter 3, Chapter 4, and Chapter 5. In Section 2.1, our network model is introduced and the challenges of wireless medium access control are reviewed. Definitions are provided in Section 2.2. In Section 2.3, the connection between medium access control and transmitter persistences is discussed. Section 2.4 provides a high-level review of existing approaches to medium access control in wireless networks. The approaches are organized in a continuum, ranging from pure contention-based to schedule-based schemes. Existing work related to single channel wireless medium access control is reviewed in Section 2.5. Finally, the `ns-2` simulator is introduced in Section 2.6.

2.1 Preliminaries

Let a *node* be a computing device equipped with a single half-duplex transceiver. A *network* is a collection of N nodes, labelled with $\{1, \ldots, N\}$, operating over a common channel. The network topology is captured by the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the vertex set $\mathcal{V}$ contains the N nodes. Edge (i, j), for $1 \leq i, j \leq N$ and $i \neq j$, is in the edge set $\mathcal{E}$ if the transmissions of node i can be heard by node j. Node i is a *neighbour* of node j if $(i, j) \in \mathcal{E}$. The *neighbourhood* of node j contains the neighbours of node j and node j itself. The directed edges of $\mathcal{G}$ accommodate asymmetric hearing between nodes. However, we assume a symmetric hearing matrix, implying that $(i, j) \in \mathcal{E}$ if, and only if, $(j, i) \in \mathcal{E}$.

The transmissions of neighbours of node j have the potential to interfere with one another at node j; any concurrent transmissions *collide* at node j. Depending on the signal strength of the competing transmissions, it is possible for node j to correctly

decode, or *capture*, one of the transmissions in spite of the interference caused by the other transmissions. Within the context of this thesis, we make the pessimistic assumption that collisions prevent successful reception of all transmissions involved, regardless of relative signal strengths; *i.e.*, we assume capture does not occur.

Contention occurs when two or more neighbours of node j have messages to send and at least one message is destined for node j. (If transmitted concurrently, these messages collide at node j.) The magnitude of the contention for the channel at node j is proportional to the number of neighbours that have messages to send. A MAC protocol is *contention-based* if it decides when and how frequently a node can transmit based on (perceived) contention levels in the network.

The limited range of wireless transmissions can result in sparsely connected networks in which pairs of nodes are unable to communicate directly. Figure 2.1 shows an example of such a network. Node 6 can only communicate with node 5 and is disconnected from nodes 1, 2, 3, and 4. Such a network is said to be *multi-hop* because communication between disconnected nodes is enabled by forwarding messages along a multi-hop path. For instance, communication between node 1 and node 6 is facilitated by nodes 2, 3, and 5 acting as intermediate routers.

Sufficiently sparse networks may allow for *spatial reuse* of the channel. Nodes can transmit concurrently without risk of collision provided they do not share a common neighbour. In Figure 2.1, nodes 1 and 5 can transmit concurrently without risk of collision. Efficient spatial reuse can improve the performance (in terms of throughput and delay) of a multi-hop wireless network. However, multi-hop networks pose a challenge to medium access control, namely in the form of *hidden terminals* [86]. Consider packet transmissions A and B of nodes 1 and 3, respectively, in Figure 2.1. Nodes 1 and 3 are out of range from each other and cannot hear one another's

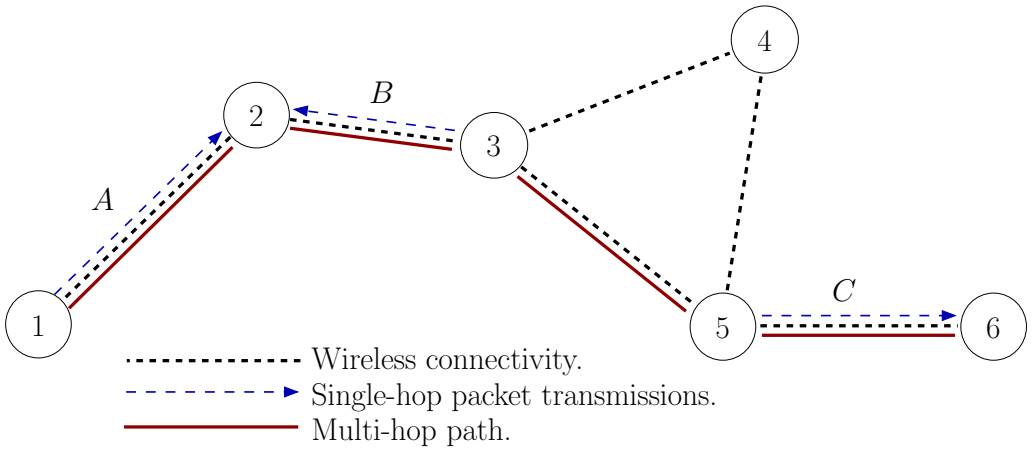


Figure 2.1: An example multi-hop network with a multi-hop path transversing the network from node 1 to node 6 and three unicast, single-hop packet transmissions labelled A, B, and C originating from nodes 1, 3, and 5, respectively.

transmissions. Yet, if they transmit concurrently, their transmissions collide at node 2. Here, node 1 is "hidden" from node 3, and vice versa. Carrier sense multiple access (CSMA) does not prevent collisions of this form because the hidden nodes cannot sense the transmissions of one another.

Another shortcoming of CSMA is its creation of *exposed terminals*. Consider packet transmissions B and C originating from nodes 3 and 5, respectively, in Figure 2.1. The destination of packet B is node 2 and the destination of packet C is node 6. Because node 2 is out of range of node 5, and node 6 is out of range of node 3, packets B and C can be transmitted concurrently without colliding at either intended receiver. However, carrier sensing prevents concurrent transmission of the two packets. Here, nodes 3 and 5 are "exposed" to each other. Exposed terminals can reduce the performance of a network by limiting efficient spatial reuse.

2.2 Definitions

2.2.1 Resource Allocation

The lexicographic max-min allocation is a foundational concept for the TLA allocation discussed in Chapter 3, the ATLAS MAC described in Chapter 4, and the practical use of the variable-weight topology transparent schedules of Chapter 5. We define it here.

Let R be a set of N *resources*, with *capacity* $\boldsymbol{c} = (c_1, \ldots, c_N)$. Let D be a set of M *demands*, with magnitudes $\boldsymbol{w} = (w_1, \ldots, w_M)$. $D_j \subseteq D$ is the set of demands that require resource $j \in R$; resources $R_i \subseteq R$ are required by demand $i \in D$. Each demand $i \in D$ utilizes the capacity of all resources in R_i equally and simultaneously. A *resource allocation* is a vector $\boldsymbol{s} = (s_1, \ldots, s_M)$, with $s_i \geq 0$ for $1 \leq i \leq M$. It is *feasible* if

$$\sum_{i \in D_j} s_i \leq c_j$$

for all $j \in R$ and $s_i \leq w_i$ for all $i \in D$. Demand $i \in D$ is *satisfied* if $s_i \geq w_i$. Resource $j \in R$ is *saturated* if

$$\sum_{i \in D_j} s_i \geq c_j.$$

Definition 2.1 and Definition 2.2 define the lexicographic max-min solution to the resource allocation problem.

Definition 2.1. *[71] Let* $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ *satisfy* $x_1 \leq x_2 \leq \ldots \leq x_n$*, and* $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ *satisfy* $y_1 \leq y_2 \leq \ldots \leq y_n$*. Then* $\boldsymbol{x}$ *is lexicographically greater than* $\boldsymbol{y}$ *if there exists an index* k*,* $1 \leq k \leq n$*, such that* $x_i = y_i$ *for all* $1 \leq i < k$ *and* $x_k > y_k$*. An allocation* $\boldsymbol{s} = (s_1, \ldots, s_M)$ *is* lexicographically max-min *if the vector is lexicographically greatest among all feasible allocations when each is sorted in non-decreasing order.*

Definition 2.2. *[71] A feasible allocation* $\boldsymbol{s}$ *is* lexicographically max-min *if, for every demand* $i \in D$*, either the demand is satisfied, or there exists a saturated resource* j *with* $i \in D_j$ *where* $s_i = \max\{s_k : k \in D_j\}$.

For MAC channel allocation, each transmitter is a demand, and each receiver is a resource. Transmitters in D_j are precisely those contending for access at receiver j. R_i contains the receivers that are within range of transmitter i. The sets D_j and R_i capture the network topology. When nodes are identical, receiver capacities can be normalized to a unit value so that $c_j = 1, 1 \leq j \leq N$. Each node is equipped with a single half-duplex transceiver, so $N = M$.

2.2.2 Topology Transparent Scheduling

Consider a schedule for which time is divided into equal length slots and grouped into frames of length n. Label the slots from 1 to n and let $S = \{1, \ldots, n\}$ be the set of all slots. Then, transmission schedule $F \subseteq S$ permits transmission within slot $x \in S$ if $x \in F$. Schedules are repeated cyclically.

Definition 2.3. *[28] The set of schedules* $\mathbb{F} = \{F_1, \ldots, F_N\}$ *is* topology transparent *with design parameters* N *and* $D_{max} \leq N$ *if, for any set of* $\nu < D_{max}$ *schedules* $\mathbb{F}^{(\nu)} \subseteq \mathbb{F}$*, each schedule not in* $\mathbb{F}^{(\nu)}$ *contains at least one slot not in any of the schedules in* $\mathbb{F}^{(\nu)}$.

In Chapter 5, we work with variable weight schedules.

Definition 2.4. *The* weight *of schedule* F*, denoted* $\mathrm{wt}(F)$*, is* $|F|$.

2.2.3 Relevant Combinatorial Structures

In the language of cover-free families [27], the slots in S form the point set and the schedules in $\mathbb{F}$ form the blocks of an r-cover-free family where $r = \mathrm{D}_{\max} - 1$. The points in each block are the transmission slots in the corresponding schedule. The block size (*i.e.*, the number of points in the block) determines the weight of the schedule.

Definition 2.5. *[27] Let $\boldsymbol{X}$ be an n-set of points and $\mathbb{B}$ be a family of sets (called blocks) on $\boldsymbol{X}$. The set family $(\boldsymbol{X}, \mathbb{B})$ is* r-cover-free *if for any r blocks $B_1, \ldots, B_r \in \mathbb{B}$, and any other block $B_0 \in \mathbb{B}$, $B_0 \not\subseteq \bigcup_{i=1}^{r} B_i$.*

There are a number of well known constructions of cover-free families. We define a few here and relate them to the topology transparent schedules that they generate.

Definition 2.6. *[42] A $v^t \times k$ array A with entries from a v-set V is an* orthogonal array*, denoted $OA(t, k, v)$, with v levels and strength t, $1 \leq t \leq k$, if every $v^t \times t$ sub-array of A contains each t-tuple based on V exactly once as a row.*

For the orthogonal arrays discussed here, index $\lambda = 1$.

Definition 2.7. *[79] A* transversal design *of strength t, block size k, and order v, denoted $TD(t, k, v)$, is a triple $(\boldsymbol{X}, \mathbb{G}, \mathbb{B})$, where (1) $\boldsymbol{X}$ is a set of kv points; (2) $\mathbb{G}$ is a partition of V into k groups, each of size v; (3) $\mathbb{B}$ is a collection of k-subsets of $\boldsymbol{X}$ called blocks; and (4) any t points taken from t distinct groups is contained in exactly one block.*

Again, we presume an index $\lambda = 1$.

A TD(t, k, v) is equivalent to an orthogonal array with strength t, k factors, and v levels [42]. We use the language of transversal designs because it maps readily to cover-free families. A TD(t, k, v) contains v^t blocks. Any block intersects another in no more than $t-1$ points; indeed, an intersection of t or more points identifies a t-set that is contained in more than one block, violating the fourth requirement of Definition 2.7. Any d blocks intersect another in at most $d(t-1)$ points. The block remains uncovered if the number of intersecting points is at least one smaller than the block size, *i.e.*, when $d(t-1) \leq k-1$. Therefore the block remains uncovered by up to $d = \left\lfloor \frac{k-1}{t-1} \right\rfloor$ other blocks, forming a $\left(\left\lfloor \frac{k-1}{t-1} \right\rfloor\right)$-cover-free family and a topology transparent schedule with

$$\mathrm{D}_{\max} = \left\lfloor \frac{k-1}{t-1} \right\rfloor + 1 \text{ and } N = v^t.$$

Definition 2.8. *[27] A* Steiner system*, $S(t, k, v)$, for $2 \leq t < k < v$ is a v-set V together with a family $\mathbb{B}$ of k-subsets of V (blocks) with the property that every t-subset of V is contained in exactly one block.*

There are $\binom{v}{t}$ possible t-subsets of V and each block contains $\binom{k}{t}$ t-subsets. Because each t-subset occurs in exactly one block, the number of blocks in the $S(t, k, v)$ is

$$\binom{v}{t} \Big/ \binom{k}{t}.$$

Any two blocks intersect in no more than $t-1$ points, forming a $\left(\left\lfloor \frac{k-1}{t-1} \right\rfloor\right)$-cover-free family. Steiner systems have been shown to maximize the number of blocks in a cover-free family [33]. In terms of topology transparent scheduling, this means Steiner systems generate schedules with the shortest frame length for given network and maximum neighbourhood sizes [30].

2.3 The Persistence of Medium Access Control

The *demand* of a node is the fraction of time that it would spend transmitting in the absence of other transmitters. Its *occupancy* is the fraction of time it actually transmits. Its *persistence* is the fraction of time that it is permitted to transmit. Demands result from network traffic delivered to the MAC layer by higher layers in the protocol stack; occupancies are observed results of employing a specific MAC protocol to handle this network traffic. What are persistences? We claim that every MAC protocol, either explicitly or implicitly, dictates each node's persistence. This is true even for contention-based schemes such as IEEE 802.11. (An evaluation of the persistence of IEEE 802.11 is provided in Appendix A.1.) To make this precise, we examine observed behaviour of a MAC protocol. The *instantaneous occupancy* for a single packet transmission is

$$\frac{\text{Transmit Time}}{\text{Idle Time} + \text{MAC Latency} + \text{Transmit Time}},$$

where Transmit Time is the time required to transmit the packet, MAC Latency is the time the packet spends queued for transmission at the MAC layer, and Idle Time is the time spent waiting for a packet to transmit. Then occupancy is a (long-term) average of these instantaneous occupancies. Because a node need not have packets queued at all times, occupancy is a lower bound on persistence. When every node always has packets queued for transmission and employs each transmission opportunity, occupancy is persistence. If permission to transmit is granted only when the transmitter has a packet to transmit, persistence is occupancy. In one way, this makes occupancy and persistence synonymous. Nevertheless, there remains a very important distinction: occupancies address how frequently transmitters *do* access the channel, while persistences address how frequently they *could* access the channel.

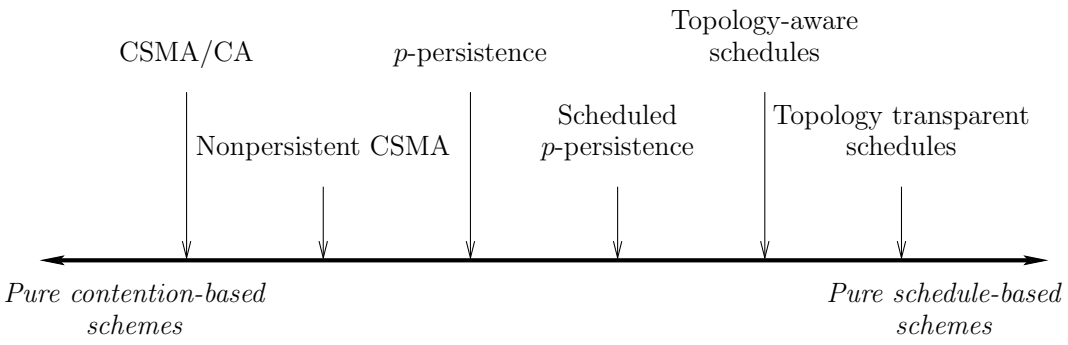


Figure 2.2: A continuum of approaches to medium access control, ranging from purely contention-based to schedule-based.

2.4 A Continuum of Approaches to Medium Access Control

Existing MAC protocols for wireless networks can be loosely organized as a continuum ranging from pure contention-based to schedule-based schemes. Figure 2.2 depicts this organization for a representative set of approaches to medium access control.

On the left of Figure 2.2 is the family of Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) [31] protocols, representing pure contention-based MACs. These MACs do not set a predefined persistence level. A node running CSMA/CA transmits a packet when it has one to send, but only after sensing the channel to make sure it is not in use. If the channel is in use, it waits a random amount of time before trying again. After each failed attempt to access the channel, the node increases the expected time before it tries again, effectively backing off when contention is detected in the network. The primary strength of these protocols is their inherent ability to adapt. Adaptation allows a node to increase its channel utilization when in lightly loaded neighbourhoods, and back off when in more heavily loaded neighbourhoods. We review existing contention-based protocols in Section 2.5.1.

Nonpersistent CSMA [86] does not employ a backoff mechanism. It transmits when it has something to send, but only after sensing the channel to see if it is available.

If it finds the channel already in use, it waits for a random amount of time and tries again. The expected time it waits before trying again remains constant, regardless of channel contention.

In contrast to CSMA/CA and nonpersistent CSMA protocols, p-persistent protocols [86] do not sense the channel before transmitting, nor do they back off when contention is detected in the network. Instead, time is divided into slots long enough to support a single packet transmission. When a node has something to send, it transmits in the current slot with probability p and defers to the next slot with probability $q = 1 - p$. The decision to transmit is made without regard to past transmission attempts.

Scheduled p-persistence is a (k, v)-scheduled MAC [29] in which slots are organized into frames of length v. Each node is permitted k transmission slots from each frame, yielding an effective persistence of $p = k/v$. Each node selects k new transmission slots uniformly at random at the start of every frame. Nodes transmit in their k transmission slots when they have a packet to send, and refrain from transmitting in all other slots. The frame structure of scheduled p-persistence bounds the maximum time between transmission slots, helping to reduce variation in packet delay [29]. A related protocol is scheduled vector-persistence. A vector $\mathbf{p} = (p_1, p_2, \ldots, p_N)$ defines the persistence for each of the N nodes. Node i implements persistence p_i by selecting $k_i = \lfloor p_i n \rfloor$ random transmission slots at the start of every frame.

Although scheduled p-persistence assumes slot and frame synchronization, its random schedules do not prevent collisions in the network. In contrast, topology-aware schedules attempt to generate collision-free slot assignments, usually with the goal of maximizing spatial reuse of slots. Some topology-aware protocols compute their slot

assignments in a distributed manner allowing for periodic updates to accommodate changes in network topology. The collision-free slot assignments generated by these schemes avoid power wasted on collisions and can offer a bound on maximum delay. However, the collision-free schedules are not trivial to generate; the problem of deciding whether a schedule has the shortest possible frame length is NP-hard [34],[60]. Existing topology-aware protocols are reviewed in Section 2.5.2.

Finally, topology-transparent MACs employ fixed schedules that do not change regardless of topology. The schedules are designed to ensure at least one successful transmission to each neighbour per frame. The most primitive topology transparent MAC is Time Division Multiple Access (TDMA) [86] in which every node is assigned exactly one transmission opportunity for every frame. Although TDMA provides a collision-free schedule, it does not take advantage of spatial reuse. Its frame length is greater than or equal to the number of nodes in the network, resulting in long delays and low throughput for large networks. Topology transparent schemes have been proposed that assign each node multiple transmission slots per frame, ensuring each node a successful transmission within the frame to each neighbour provided the maximum neighbourhood size is not too large. These topology transparent MACs are discussed further in Section 2.5.3.

Many protocols employ a hybrid of scheduled- and contention-based mechanisms. Several of these protocols are reviewed in Section 2.5.4. For these hybrid protocols, their relative positioning on the continuum of Figure 2.2 becomes subjective. Nevertheless, the continuum provides a general classification of the protocols in terms of their underlying allocation strategy. Protocols on the left of Figure 2.2 are primarily contention-based and adapt quickly to changes in the network. Protocols on the right

of Figure 2.2 are primarily schedule-based and adapt slowly, if at all, to network changes. We turn now to review existing medium access control protocols.

2.5 Related Work

2.5.1 Contention-based MAC Protocols

ALOHA [9], a simple contention-based MAC, transmits a message whenever there is one to send. If unsuccessful, it waits a random amount of time before retransmitting. Other schemes build on ALOHA by incorporating control messaging or revising the back-off mechanism for retransmissions.

Various control messages have been proposed. Collision avoidance in the form of a Ready to Send (RTS) and Clear to Send (CTS) message handshake is used by MACA [55], MACAW [14], and IEEE 802.11 [45]. Following this scheme, when a node has something to transmit, it first transmits an RTS message announcing its intention to transmit to a particular receiver. The receiver responds with a CTS message to inform the sender that it is clear to send and to inform the receiver's neighbours, including any nodes that are hidden from the sender, of the pending transmission. Upon hearing the CTS, the neighbours know to remain quiet, reducing the likelihood of collisions caused by hidden nodes. MACAW and IEEE 802.11 also implement per packet acknowledgements, enabling the sender to detect failed transmissions and retransmit accordingly. Ready to Receive (RTR) messages are used by the receiver initiated protocols MACA-BI [85], RIMA-SP and RIMA-DP [36]. In these protocols, the receiver coordinates the transmissions of its neighbours by broadcasting RTR messages. Negative acknowledgements are employed in [78].

ALOHA does not change the expected time between packet retransmissions; it essentially operates with a fixed contention window. Binary exponential back-off

(BEB), originally developed for wired communication [44], is employed by MACA and IEEE 802.11. When using BEB, a node's retransmission times are selected uniformly at random from a contention window. The size of the contention window is doubled up to a maximum value following each failed transmission and reset to a minimum value upon a successful transmission.

Numerous variations on BEB have been proposed. In [14], Multiplicative Increase Linear Decrease (MILD) is proposed as a method to resolve the inherent unfairness of BEB. MILD implements a multiplicative increase to the contention window following failed transmissions and a linear decrease following a successful attempt. As a result, nodes back-off quickly, but do not recover immediately after a successful transmission. It takes many successful transmissions to reset the contention window to its smallest value. A node running MILD also updates its contention window upon overhearing a successful transmission; the node assumes the contention window used for the successful transmission. Linear/Multiplicative Increase Linear Decrease (LMILD) [32] is similar to MILD except that it responds to overheard successful transmissions with a linear increase to the contention window size. Because nodes running LMILD do not copy the contention window from successful transmissions, neighbouring nodes are free to adjust their contention windows independently to accommodate local network conditions. The Sensing Back-off Algorithm (SBA) [40] implements a multiplicative increase to the contention window following a failed transmission, a linear decrease upon overhearing a successful transmission to a neighbour, and a multiplicative decrease following each successful transmission.

2.5.2 Topology-Aware Scheduled MAC Protocols

Many of the topology-aware scheduled approaches to medium access control compute collision-free TDMA—one slot per node per frame—schedules that take advantage of

spatial reuse to minimize the frame length. In graph theoretic terms, these algorithms compute a *distance-2 vertex colouring* on the network topology graph; the nodes are the vertices and node adjacencies identify the edges in the graph. A graph's *distance-2 chromatic number* is the smallest number of colours required to achieve a distance-2 colouring [72, 94]. The problem of deciding whether an integer n is the distance-2 chromatic number of a graph is NP-hard [34]. In terms of schedules, the colours identify slots and the number of colours in a colouring determines the number of slots in a frame, affecting both the delay and throughput of the schedule. Because the colouring enables spatial reuse of slots, this approach is sometimes referred to as *spatial reuse TDMA*.

One of the first topology-aware scheduling protocols for multi-hop networks was proposed by Chlamtac and Pinter in [21]. This protocol runs a distributed algorithm to compute a distance-2 colouring with a provable bound on the number of colours. A centralized algorithm for computing distance-2 colourings is given in [72]. This algorithm has two phases, a labelling phase and a colouring phase. The labelling phase determines the order in which vertices are coloured. Three variations on labelling are proposed: (1) vertices labelled in random order (RAND), (2) vertices labelled according to the number of neighbouring vertices with minimum neighbourhood vertices labelled first (MNF), and (3) vertices labelled progressive minimum neighbours first (PMNF), a variant of MNF in which the edges of a vertex are removed once the vertex is labelled. Although PMNF provides an improved bound on the number of colours used in the distance-2 colouring, the computation of the RAND label ordering is more easily distributed.

Distributed-RAND (DRAND) [76] is a distributed implementation of RAND. DRAND runs a series of loosely synchronized rounds. In each round, auctions are held

to select one node from each two-hop neighbourhood to choose a transmission slot. The selected nodes choose a slot that has yet to be chosen by any of their two-hop neighbours during previous rounds. The chosen slots are published to the two-hop neighbours before the next round is started. The result is a collision-free spatial reuse TDMA schedule with expected frame length equal to that of RAND.

TDMA schedules with exactly one transmission slot per node per frame provision every node with equal channel access. These schedules do not use the channel to its fullest potential in sparsely populated neighbourhoods and nodes cannot adjust their channel access to accommodate varying traffic loads. To address these concerns, periodic slot chains are proposed in [50]. The slot chains are not limited to a fixed length frame. A slot chain is defined by its starting transmission slot and period between its subsequent transmission slots. By combining multiple slot chains with different periods, schedules are constructed to target any rational persistence in the range $[0, 1]$ and to support varying neighbourhood sizes and traffic loads.

In [98], a five phase reservation protocol (FPRP) is proposed as a mechanism to compute collision-free schedules. While FPRP can compute distance-2 colourings with each node assigned a single colour, it can also generate colourings where nodes are assigned multiple colours. In terms of schedules, this allows the number of slots assigned to a node to adjust according to traffic load and neighbourhood size. To accommodate changes in traffic load and topology, FPRP executes "reservation" frames periodically, interspersed among "information" frames. Each reservation frame computes the slot schedule for subsequent information frames in which data is transmitted. Depending on how quickly the network is changing, overhead can be reduced by running the reservation frames less frequently. Each reservation frame contains a sequence of reservation cycles. Each reservation cycle implements a five phase protocol (hence

the protocol name) through which nodes reserve collision-free transmission slots. The principles behind FPRP are extended further in the Evolutionary-TDMA MAC [97] which maintains two schedules: a collision-free schedule that reserves exactly one slot per frame and a schedule that can reserve multiple slots for each node to support broadcast, multicast, and unicast traffic with varying quality of service requirements.

Some topology-aware MAC protocols do not implement collision-free schedules. An example is SEEDEX [77] in which schedules are derived from a deterministic hash function. Each node is assigned a unique seed, which it hashes to recover its schedule. Nodes learn the identities and the seeds of their two-hop neighbours, allowing them to predict each other's schedules. The schedules do not identify transmission slots in the traditional sense; they identify "potential" transmission slots. When a node has something to send, it transmits in its next potential transmission slot with probability p selected to maximize the likelihood of a successful transmission occurring in that slot, either by itself or by the other nodes competing for use of the slot. The probability p is derived from the number and identity of nodes within range of the intended receiver who have the potential to transmit. Although the transmission probabilities accommodate neighbourhood size, they do not accommodate traffic load.

2.5.3 Topology Transparent Scheduled MAC Protocols

Topology transparent scheduling was first proposed by Chlamtac and Faragó [16] who use polynomials over finite fields to generate schedules with limited intersection. The polynomial degree and finite field is selected with the goal to minimize frame length and, therefore, maximum delay for a given N and $\mathrm{D}_{\max}$. In [84], the delay guarantee is shown to degrade gracefully for neighbourhood sizes larger than $\mathrm{D}_{\max}$.

The collection of topology transparent schedules is shown equivalent to orthogonal arrays in [83]; the requirements for topology transparency are implicitly satisfied in the structure of an orthogonal array, regardless of its construction. Topology transparent schedules are further generalized as cover-free families. The densest cover-free families, those constructed from Steiner systems, provide the shortest possible frame length for a given D_{max} and N [30].

In [16], [83] and [28], schedules are designed to minimize frame length while maintaining *at least one* successful transmission per frame. Alternatively, topology transparent schedules have been designed to ensure *more than one* successful transmission per frame [53], [73], [74], improving minimum throughput at the expense of maximum delay. A number of other constructions have been proposed [96], [80], [12], [54], each of which employs constant-weight schedules.

If the size of a neighbourhood exceeds D_{max}, the guarantee for successful communication is lost. Protocol threading [17] has been proposed as a method to handle inevitable violations of D_{max}. Multiple schedules—each with a unique D_{max}—are interleaved to achieve different delay guarantees depending on the neighbourhood size. The delay guarantee is lengthened in proportion to the number of interleaved schedules, resulting in a maximum delay significantly longer than that of TDMA. For the scheme proposed in [82], each node handles violations of D_{max} by picking a new constant-weight schedule selected at random, voiding the guarantee initially offered by the schedule.

Schemes mentioned so far assume frame and slot synchronization. Optical orthogonal codes (OOCs) [24] are used to create slot synchronized [23] and asynchronous [22] topology transparent schedules. Variable-weight OOCs [95], initially designed to support multiple quality of service requirements in CDMA over optical

fiber, are applied to topology transparent scheduling in [59]. In this scheme, each node is assigned a single schedule with one of two weights; the choice of weight depends on the node's desired quality of service, assumed to be static. The nodes do not adapt to variations in neighbourhood size or traffic demand. A discussion of practical schedule weights for medium access control is provided in Appendix A.3. The distribution of schedule weights is observed to be as important as their number. A second weight, when both weights are small, improves network throughput very little compared to a constant-weight schedule. Other applications of topology transparent scheduling to CDMA networks are the link activation schemes of [18], [81].

2.5.4 Hybrid MAC Protocols

A number of MAC protocols employ a hybrid approach, relying on both contention and schedule-based mechanisms to implement medium access control. A Dynamically Adaptive Protocol for Transmission (ADAPT), proposed in [19], employs CSMA/CA within the slots of a TDMA schedule. Each node is assigned its own slot in the TDMA schedule, to which it is given priority access. If a slot is not claimed by its owner, other nodes are free to contend for use of the slot. Two RTS/CTS handshakes are performed at the start of each slot: The first allows the slot owner to claim and use the slot. The second handshake enables slots to be reused by other nodes when the slot owner is out of range or has nothing to transmit. ADAPT behaves like CSMA/CA in lightly loaded neighbourhoods and resorts to TDMA in heavily loaded and dense neighbourhoods, successfully combining the advantages of contention- and schedule-based medium access control. The slot structure of ADAPT is extended to support reliable broadcast transmissions in an Adaptive Broadcast (ABROAD) protocol [20]. In [66], ADAPT and ABROAD are integrated into A Generalized Transmission (AGENT) protocol to support both unicast and reliable broadcast within a single MAC.

Collision Avoidance Time Allocation (CATA) [87] is a slotted protocol that employs contention at the start of each slot to negotiate collision-free schedules. In CATA, each slot is divided into five mini-slots. The first four mini-slots are reserved for contention-based control messages that allow nodes to reserve slots for unicast, multicast, and broadcast transmissions. The fifth mini-slot is larger than the first four and is meant for data transmissions. A reservation made in one frame persists for the frames that follow until the reservation is terminated.

Another hybrid MAC designed for static wireless sensor networks is Zebra-MAC (Z-MAC) [75]. Like ADAPT, Z-MAC superimposes CSMA/CA over slots. There are two distinct differences: (1) Z-MAC uses spatial reuse TDMA schedules computed by DRAND instead of traditional TDMA schedules. (2) Nodes running Z-MAC operate in either low contention level (LCL) mode or high contention level (HCL) mode. In LCL mode, a node can contend for use of any slot, although slot owners are given priority through careful selection of the contention windows. In HCL mode, a node is only permitted to contend for use of its own slot, per the spatial reuse schedule. Nodes operate in HCL mode in response to explicit contention notifications (ECNs) sent by neighbouring nodes. Z-MAC represents a medium access control protocol that exhibits the strengths of CSMA/CA when lightly loaded and the stability of a topology-aware schedule when heavily loaded. Due to the complexity of DRAND, the TDMA schedules are only computed once during network initialization, preventing the protocol's use in highly mobile networks.

2.6 The Network Simulator 2

The Network Simulator 2 (`ns-2`) [69], is a discrete event simulator designed specifically for wired and wireless networks. Being both open-source and free for research and educational purposes, it is a popular tool for the evaluation of new or improved network

protocols. We make extensive use of `ns-2` in evaluating the contributions presented in this thesis; here, we give a short introduction and discuss several of its limitations.

While `ns-2` is written in C++, it implements a robust Object oriented Tcl (OTcl) interpreter allowing much of the code base to be written in OTcl. At a high-level, the `ns-2` architecture is organized in two hierararchies that mirror each other: a *compiled* hierarchy and an *interpreted* hierarchy [70]. The compiled hierarchy is described using C++ and the interpreted hierarchy is described in OTcl. Through this architecture, `ns-2` enables the efficient and detailed modelling of protocols in C++ while simultaneously providing a streamlined mechanism to configure and control the network using OTcl written at a higher layer of abstraction.

Implementations of many standard protocols are included with the base distribution of `ns-2`, including the IEEE 802.11 MAC [45], the Address Resolution Protocol (ARP) [3], On Demand Distance Vector (AODV) Routing [8], the User Datagram Protocol (UDP) [1], and the Transmission Control Protocol (TCP) [2]. Various flavours of TCP are supported including TCP-Reno [5] and TCP with selective acknowledgements [7]. Most importantly, custom network functions can be incorporated into `ns-2` using the OTcl interface to be evaluated within the context of a full network stack. The simulator also provides out-of-the-box support for different types of traffic; these include constant bit-rate flows and file transfers mimicking the File Transfer Protocol (FTP) [4]. The simulations discussed here are either loaded with constant bit-rate flows transported over UDP or FTP file transfers transported over TCP.

The `ns-2` simulator supports simple node movements defined in terms of trajectory and speed. However, complex node movements can be simulated by repeatedly updating a node's trajectory and speed. The random waypoint mobility model [52] is a popular model for wireless node mobility. Under this model, nodes are

placed at random positions within the simulation area. The movements of each node are determined independently of the others. Each node (1) picks a destination (within the simulation area), speed, and pause time, (2) remains stationary for the duration of the pause time, and then (3) moves toward the destination at the specified speed. Upon reaching its destination, the node repeats these three steps. The result is a network of nodes that move randomly about the simulation area, pausing occasionally before moving on. The degree of node mobility is influenced by the dimensions of the simulation area and the distributions from which the node speeds and pause times are selected. Given time, the distribution of node speeds and locations converge on a "steady-state" distribution [68]. The longer the nodes are permitted to repeat these three steps, the closer the distributions get to the steady-state. Rather than wait for the network to converge, it is possible to initialize node speeds and pause times so that the initial distributions of node speed and location match that of the steady-state distribution. This is accomplished by the steady-state mobility generator of [49].

A primary drawback of the `ns-2` simulator is its overly simplistic models of the wireless channel. `ns-2` supports three radio propagation models: the free space model, the two-ray ground reflection model, and the shadowing model. Several key assumptions are made by the free space and two-ray ground reflection models:

1. The transmission range of node A is modelled as a circular disk, centered at node A. Other nodes can hear the transmissions of node A if, and only if, they are positioned within the disk.

2. The transmission ranges of all nodes share a common radius. Implicit in this assumption is a symmetric hearing matrix, *i.e.*, node A can hear node B if, and only if, node B can hear node A.

3. The interference range of node A is also modelled as a disk centered at node A. Node A interferes at nodes within its disk and no others.

4. Node transmissions are the only source of interference.

The shadowing model weakens these assumptions slightly by implementing probabilistic communication when the distance between nodes is near the communication range. Under this model, the transmission and interference ranges are no longer ideal circles, symmetric hearing is not guaranteed, and random interference is modelled by probabilistic packet loss. The MAC protocols discussed in Chapter 3, Chapter 4, and Chapter 5 assume a symmetric hearing matrix, so we employ the two-ray ground reflection model.

In spite of its limitations, `ns-2` is an effective tool for exploring and clarifying ideas; as such, it remains the de facto simulator for research in wireless networks.

2.7 Summary

In this section, we have presented background material in preparation for the contributions of the remaining chapters. We have defined our network model and notation, introduced the basic principles behind medium access control in wireless networks, and reviewed existing wireless MAC protocols. The definitions provide the foundation on which the TLA allocation, ATLAS, and VWATT are built and the introduction to `ns-2` gives context to our evaluations of each. We now turn our attention to the first contribution, the TLA allocation.

Chapter 3

TOPOLOGY- AND LOAD-AWARE PERSISTENCES

This chapter addresses the selection of transmitter persistences: At what persistence should a transmitter be permitted to access the channel? To answer this question, we propose the Topology- and Load-Aware (TLA) allocation.

For MAC channel allocation, each transmitter is a demand, and each receiver j is a resource required by demands in D_j, and no others. Transmitters in D_j are precisely the demands contending for access at receiver j. When nodes are identical, receiver capacities can be normalized to a unit value so that $c_i = 1$, $1 \leq i \leq N$. The lexicographic max-min solution, when applied to transmitter persistences, defines the TLA allocation. The allocation ensures that no receiver is overrun, no transmitter is over-provisioned, and every unsatisfied transmitter has been allocated the largest share at a saturated receiver. This model reflects desirable properties for channel access, without limiting the MAC protocol employed. The computation and evaluation of the TLA allocation is the prime focus of this chapter. The contributions of this chapter are as follows:

1. The TLA allocation to transmitter persistences, defined in terms of topology and traffic load, is described as a target allocation goal.

2. A centralized algorithm and a distributed algorithm are developed to compute TLA persistences, and the correctness of each is established.

3. The distributed algorithm is integrated into a scheduled MAC protocol and applied to wireless networks simulated using `ns-2` [69].

4. Expectation and variation in delay, throughput, and drop rate reveal potential merits of the TLA allocation.

This rest of this chapter is organized as follows. The centralized and distributed algorithms for computing TLA persistences are described in Section 3.1 and Section 3.2. The integration of the distributed algorithm is described in Section 3.3 and simulation results are presented in Section 3.4. The distributed algorithm is considered for use in networks with dynamic topologies and changing traffic demands; next steps toward a fully adaptive solution are outlined in Section 3.5. Use of the distributed algorithm to compute fair allocations and enable quality of service is discussed in Section 3.5.

3.1 Centralized Computation of TLA Persistences

Algorithm 3.1 gives a centralized algorithm to compute TLA persistences. The set V contains all nodes in the network, while set V_{active} contains only the nodes whose persistences have yet to be finalized. Initially, $V_{active} = V$ and the capacity vector $\boldsymbol{c} = (1, \ldots, 1)$. The matrix $\boldsymbol{T} = [t_{i,j}]$ is the adjacency matrix of the neighbourhood graph. Then $D_j = \{i : t_{i,j} = 1\}$. At each recursive step a small, non-zero increment to the allocation, ϵ, is granted to each node in V_{active} and the capacity vector is updated to reflect this allocation. Nodes that have either reached their desired persistence (*i.e.*, $s_i \geq w_i$) or have been bottlenecked by a saturated resource are removed from V_{active}. The procedure continues until V_{active} is empty.

To reduce the number of recursive steps, ϵ is selected to be the smaller of $\min\left\{\frac{c_j}{|V_{active} \cap D_j|} : j \text{ not saturated}\right\}$ and $\min\{w_i : i \in V_{active}\}$. The first limits ϵ to allocate remaining capacity at node j among its unsatisfied neighbours, while the second limits ϵ so that no transmitter is over-provisioned. A larger value for ϵ generates an infeasible allocation. Lemma 3.1 and Theorem 3.2 demonstrate the correctness of Algorithm 3.1.

Lemma 3.1. *Algorithm 3.1 terminates in a finite number of recursive steps.*

Algorithm 3.1 A centralized algorithm to compute TLA persistences in a wireless network.

procedure COMPUTETLA $(V, \boldsymbol{c}, V_{active}, \boldsymbol{w}, \boldsymbol{T})$
    Inputs:
        – Set of nodes $V, N = |V|$
        – Capacity vector $\boldsymbol{c} = (c_1, \ldots, c_N)$
        – Active nodes $V_{active} \subseteq V$
        – Desired persistences $\boldsymbol{w} = (w_1, \ldots, w_N)$
        – $N \times N$ adjacency matrix $\boldsymbol{T} = [t_{i,j}]$
    Result:
        – TLA persistences $\boldsymbol{s} = (s_1, \ldots, s_N)$
    $\boldsymbol{s} \leftarrow (0, \ldots, 0)$
    // Select largest possible, non-zero ϵ to be allocated to all demands
    // in V_{active}.
    $\epsilon \leftarrow 1$
    for all $j \in V$ **do**
        if $(t_{i,j} = 1$ for some $i \in V_{active})$ **then**
            $\epsilon \leftarrow \min\left(\epsilon, \frac{c_j}{|\{i : i \in V_{active}, t_{i,j} = 1\}|}\right)$
    for all $i \in V_{active}$ **do**
        $\epsilon \leftarrow \min(\epsilon, w_i)$
    // Increase persistence of all nodes in V_{active}. Remove satisfied
    // demands from V_{active}.
    for all $i \in V_{active}$ **do**
        $s_i \leftarrow \epsilon$
        $w_i \leftarrow w_i - \epsilon$
        if $(w_i = 0)$ **then** $V_{active} \leftarrow V_{active} \setminus i$
        for all $j \in V$ **do**
            if $(t_{i,j} = 1)$ **then** $c_j \leftarrow c_j - \epsilon$
    for all $j \in V$, $i \in V_{active}$ **do**
        if $(c_j = 0$ **and** $t_{i,j} = 1)$ **then** $V_{active} \leftarrow V_{active} \setminus i$
    // Recurse until V_{active} is empty.
    if $(V_{active} \neq \emptyset)$ **then**
        $\boldsymbol{s} \leftarrow \boldsymbol{s} +$ COMPUTETLA $(V, \boldsymbol{c}, V_{active}, \boldsymbol{w}, \boldsymbol{T})$
    return $\boldsymbol{s}$
end procedure

Proof. ϵ is the smaller of $\min\left\{\frac{c_j}{|V_{active} \cap D_j|} : j \text{ not saturated}\right\}$ and $\min\{w_i : i \in V_{active}\}$. If $\epsilon = \min\left\{\frac{c_j}{|V_{active} \cap D_j|} : j \text{ not saturated}\right\}$, choose i so that $\epsilon = c_i/|V_i|$ and let $V_i = \{j \in V_{active} \cap D_i\}$. Then $\epsilon = c_i/|V_i|$ is allocated to each of the $|V_i|$ nodes in V_i, consuming capacity $(c_i/|V_i|) \cdot |V_i| = c_i$. As a result, node i is saturated, and all nodes in V_i are removed from V_{active}. If $\epsilon = \min\{w_i : i \in V_{active}\}$, choose i so that $\epsilon = w_i$; then ϵ is allocated to all nodes in V_{active} including node i. w_i is reduced to zero and node i is removed from V_{active}. Because V_{active} is finite, and at least one node is removed from V_{active} in each step, $V_{active} = \emptyset$ within a finite number of steps. □

Theorem 3.2. *Algorithm 3.1 generates the TLA allocation.*

Proof. It is sufficient to show that Algorithm 3.1 generates transmitter persistences that are lexicographically max-min. By Lemma 3.1, Algorithm 3.1 terminates with V_{active} empty. We show that allocation $\boldsymbol{s}$ is lexicographically max-min at the time each node is removed from V_{active}.

Consider when node i is removed from V_{active}. Node i is removed because either it has reached its desired demand, or there exists a saturated node within its range. In the first case, the demand of node i is satisfied and so satisfies the constraints of Definition 2.2 on page 14. For the second case, we must show that s_i is maximal among nodes contending for the channel at the saturated node. Because channel access is allocated to nodes in V_{active} equally at each step, node i is allocated more capacity than any node removed from V_{active} at an earlier step. Furthermore, any node i' still contending for channel access at the bottleneck is removed

concurrently with i from V_{active}, and hence has $s_{i'} = s_i$. By Definition 2.2, $\boldsymbol{s}$ is lexicographically max-min and is the TLA allocation. □

The correctness of Algorithm 3.1 does not require that the adjacency matrix $\boldsymbol{T}$ be symmetric, so different transmission ranges are, in principle, treated correctly.

3.2 Distributed Computation of TLA Persistences

In order to compute the persistences $\boldsymbol{s}$ in a decentralized manner, each receiver accumulates the required information locally to decide when its available capacity is allocated, either by becoming saturated or by having no remaining demands; a transmitter does not increase its persistence if it is satisfied or if it has a saturated receiver within range. Decentralization is accomplished by treating each receiver as an *auctioneer*, and each transmitter as a *bidder*. Auctioneer j holds an auction for channel access at node j, making offers to all bidders in D_j. Bidder i claims channel access offered by auctioneers in R_i. The adjacency matrix of Algorithm 3.1 defines the collection of bidders in attendance of each auction and, conversely, the collection of auctions attended by each bidder. The auctioneer at node j requires knowledge of, and communication with, a localized set of bidders in D_j. Similarly, the bidder at node i requires knowledge of, and communication with, auctioneers in R_i. Each bidder derives the TLA persistence for its host node.

The auctioneers start out with an initial offer to which neighbouring bidders respond with claims. If a bidder claims its desired persistence, it terminates. The auctioneers respond to claims by either closing their auctions or increasing their offering. Auctioneers and bidders are loosely synchronized in that a bidder does not respond with a claim until it has heard from all its auctioneers. Likewise, an auctioneer does not send out subsequent offers until it has heard from all its bidders. This process

is repeated until all bidders have claimed their desired persistence or have been limited at an auction whose capacity has been completely allocated. Upon termination, the final claim made by each bidder is the TLA persistence for the associated node. Algorithm 3.2 and Algorithm 3.3 give the auctioneer and bidder algorithms which run independently on each network node. Each *auctionMsg*[r] contains the originating auctioneer's *offer* and *closed* status for round r; each *bidderMsg*[r] contains the originating bidder's *claim* and *finished* status for round r.

We assume throughout that a node with a message to send does not delay indefinitely in sending it, and that there is no indefinite delay until successful receipt. That is, we assume reliable message delivery within finite time. Lemma 3.3 and Theorem 3.4 establish eventual termination and correctness.

Lemma 3.3. *All auctioneers and bidders in Algorithm 3.2 and Algorithm 3.3 terminate within a finite amount of time.*

Proof. First, we show that all auctioneers and bidders continue to send messages until they terminate. Second, we show that all auctioneers and bidders eventually send their final message and terminate.

Consider an auctioneer or bidder that has not terminated, but is unable to send its next message. If it is an auctioneer, it is waiting on one or more (necessarily active) bidders to submit their bids. If it is a bidder, it is waiting for one or more (necessarily open) auctioneers to announce the next offers. Form a directed graph G with a vertex for each auctioneer and a vertex for each bidder, and add a directed edge from node x to node y if x is waiting on y. To ensure that each node can send its next message eventually, it suffices to ensure that G has no directed cycle. Suppose to the contrary that G contains a directed cycle on nodes $(u^{(0)}, \ldots, u^{(L-1)})$.

Algorithm 3.2 Auctioneer for Node j.

1: **initialization**
2: $round \leftarrow 1$
3: $B \leftarrow \{i : i \in V, t_{i,j} = 1\}$
4: $B_{active} \leftarrow B$
5: $B_{recvd} \leftarrow \emptyset$
6: $auctionMsg[round].closed \leftarrow$ `False`
7: $auctionMsg[round].offer \leftarrow (1/|B_{active}|)$
8: **send** $\langle auctionMsg[round] \rangle$ to bidders in B_{active}
9: **end initialization**
10: **upon** receiving $\langle bidderMsg[round] \rangle$from bidder i
11: $claims[i] \leftarrow bidderMsg[round].claim$
12: $B_{recvd} \leftarrow B_{recvd} \cup i$
13: *// If this bidder is finished, remove its index from* B_{active}.
14: **if** $(bidderMsg[round].finished =$ `True`$)$ **then**
15: $B_{active} \leftarrow B_{active} \setminus i$
16: *// If a message has been received from every bidder in* B_{active},
17: *// respond with another offer.*
18: **if** $(B_{recvd} \supseteq B_{active})$ **then**
19: $B_{recvd} \leftarrow \emptyset$
20: $round \leftarrow round + 1$
21: **if** $(B_{active} = \emptyset)$ **then**
22: **terminate**
23: *// Compute the next offer.*
24: $X \leftarrow \left(\sum_{i \in B, i \notin B_{active}} claims[i] \right)$
25: $auctionMsg[round].offer \leftarrow \frac{1-X}{|B_{active}|}$
26: $claimed \leftarrow \left(\sum_{i \in B} claims[i] \right)$
27: **if** $(claimed = 1)$ **then**
28: $auctionMsg[round].closed \leftarrow$ `True`
29: **send** $\langle auctionMsg[round] \rangle$ to bidders in B_{active}
30: **terminate**
31: $auctionMsg[round].closed \leftarrow$ `False`
32: **send** $\langle auctionMsg[round] \rangle$ to all bidders in B_{active}
33: **end upon**

Algorithm 3.3 Bidder for Node i.

```
1: initialization
2:     round ← 1
3:     A ← {j : j ∈ V, t_{i,j} = 1}
4:     A_recvd ← ∅
5:     bottlenecked ← False
6:     maxClaim ← w_i
7: end initialization
8: upon receiving ⟨auctionMsg[round]⟩from auctioneer j
9:     offers[j] ← auctionMsg[round].offer
10:    A_recvd ← A_recvd ∪ j
11:    // If the auction is closed, it is a bottleneck.
12:    if (auctionMsg[round].closed = True) then
13:        bottlenecked ← True
14:    // If a message has been received from all auctioneers in A, it is time
15:    // to respond with a claim.
16:    if (A_recvd = A) then
17:        A_recvd ← ∅
18:        claim ← min ({offer[j] : j ∈ A} ∪ maxClaim)
19:        bidderMsg[round].claim ← claim
20:        if (bottlenecked = True or claim = maxClaim) then
21:            bidderMsg[round].finished ← True
22:            send ⟨bidderMsg[round]⟩ to all auctions in A
23:            terminate
24:        bidderMsg[round].finished ⟵ False
25:        send ⟨bidderMsg[round]⟩ to all auctions in A
26:        round ← round + 1
27: end upon
```

Because bidders and auctioneers alternate, L is even and without loss of generality $u^{(0)}$ is an auctioneer waiting on $bidderMsg[k]$ from $u^{(1)}$. For $0 \leq i < \frac{L}{2}$, auctioneer $u^{(2i)}$ is then waiting for $bidderMsg[k-i]$ from $u^{(2i+1)}$, while bidder $u^{(2i+1)}$ is waiting for $auctionMsg[k-(i+1)]$ from $u^{(2i+2)}$, arithmetic modulo L. Then $u^{(L-1)}$ is waiting for $auctionMsg[k-\frac{L}{2}]$ from $u^{(0)}$, but $u^{(0)}$ has already transmitted $auctionMsg[k-\frac{L}{2}]$, because it is waiting for $bidderMsg[k]$ from $u^{(1)}$. Therefore there is no directed cycle in G.

It remains to show that the number of messages is finite. Partition all messages into subsets, where subset M_k contains every message of the form *auctionMsg*[k] or *bidderMsg*[k]. Then M_k contains one message for each active bidder and each open auction, and hence is finite. Let auctioneer j have the smallest offer among all auctioneers with messages in M_k. This offer is equal to the previous offer made by auctioneer j plus the remaining unclaimed channel resources divided by $|B_{active}|$. Either there exists a bidder in B_{active} who claims its *maxClaim*, or all bidders in B_{active} claim the full amount offered by auctioneer j. In the first case, a bidder that claims its *maxClaim* terminates immediately. In the second case, the auction's capacity is fully allocated causing the termination of all bidders in B_{active}. In either case, $|M_{k+1}| < |M_k|$, and the total number of messages sent is finite. □

The argument behind the proof of Lemma 3.3 also sets an upper bound on the number of rounds required for algorithm termination. One or more bidders finish during each round, limiting the number of rounds to be less than or equal to N. In practice, bidders tend to finish concurrently and the number of rounds is much smaller than N; see Section 3.4.

Theorem 3.4. *Algorithm 3.2 and Algorithm 3.3 compute the TLA allocation for all nodes in the network.*

Proof. It is sufficient to show that Algorithm 3.2 and Algorithm 3.3 compute lexicographically max-min transmitter persistences. Bidder i makes a final claim for one of two reasons: Either i claims its *maxClaim* value, or one of the auctions, j, in which i participates closes. For auction j to close, it must be saturated and all of its bidders in B_{active}, including bidder

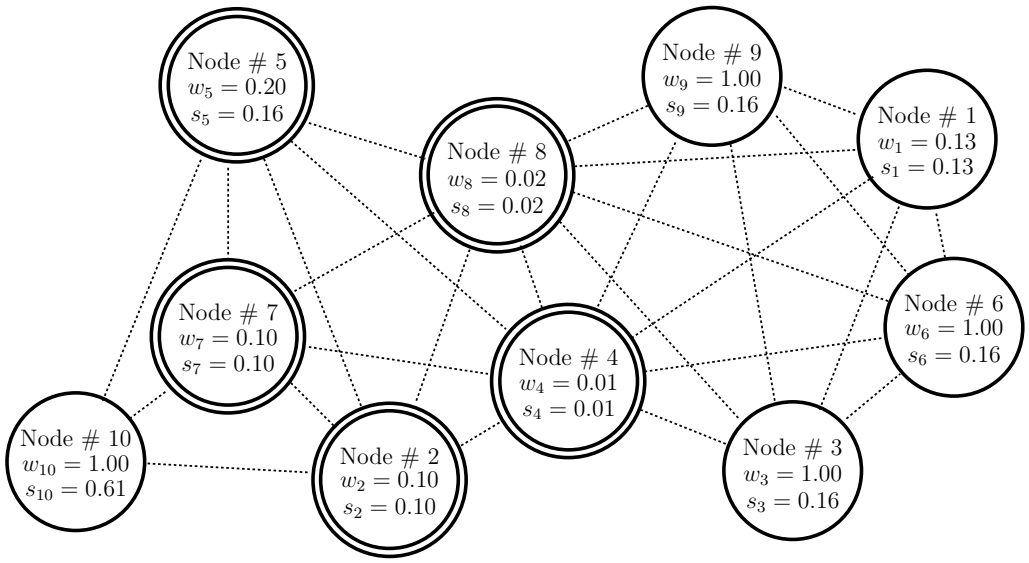


Figure 3.1: Example network and TLA allocation. Bottleneck nodes are identified by double-lined circles.

i, claimed the full amount offered by auctioneer j. Because no bidder can claim more than is offered, the final claim of node i is maximal among all claims made in auction j. By Definition 2.2, the allocation containing all final bidder claims is lexicographically max-min and is, therefore, the TLA allocation. □

Figure 3.1 shows an example network topology of 10 nodes, each labelled with desired persistence w_i and TLA persistence s_i. It can be verified that $\boldsymbol{s}$ is feasible, and that the resources of bottleneck nodes 2, 4, 5, 7, and 8 are saturated. Furthermore, while the demands of nodes 1, 2, 4, 7, and 8 are satisfied, those of nodes 3, 5, 6, 9, and 10 are not.

To cope with an asymmetric adjacency matrix, response messages generated by an auctioneer can be forwarded on to any bidders that lie outside the transmission range of the auctioneer. This raises questions about the effective implementation of the algorithm, so we turn to these issues next.

3.3 Efficient Implementation of the Distributed Algorithm

Several implementation details must be addressed before the distributed algorithm can operate in a real wireless network. Next we describe an efficient and reliable implementation.

3.3.1 Unreliable Communication

As written, the algorithm assumes lossless communication. One convenient feature of the distributed algorithm is the built-in acknowledgement system. Effectively, an *auctionMsg* acknowledges receipt of a *bidderMsg* and vice versa precluding the necessity of an additional acknowledgement scheme. Reliable interprocess communication can be achieved by retransmitting *bidderMsg* and *auctionMsg* messages repeatedly until the appropriate acknowledgements are received.

3.3.2 Message Size and Content

Auctioneer messages contain a current offer. Bidder messages contain a current claim. The number of bits required to encode them depends on the desired precision. A 10-bit encoding, supporting 1024 unique persistences, may be sufficient for most applications. Auctioneer and bidder messages must indicate whether the auction is closed, and whether the bidder has finished. One bit for each suffices. Because message retransmissions can occur, round information is required. The least significant bit of the round count is sufficient to detect the change in round.

3.3.3 Message Delivery

A primary concern is the mechanism by which auctioneer and bidder messages are delivered. Auctioneer and bidder communication can be piggybacked on existing traffic, or implemented with additional control packets. With a precision of 10 bits

for claims/offers, only 12 additional bits of information are required, making the sizes of the auctioneer and bidder messages smaller than typical control packets. If piggybacked on existing traffic by embedding auctioneer and bidder information in the MAC header, the additional 24 bits translates to an overhead of 0.3% for 1000 byte packets.

3.4 Simulation Results

In this section, we evaluate the convergence time, accuracy, and performance of the *Scheduled TLA-Persistence* MAC, a schedule-based MAC protocol in which persistences are defined by the distributed algorithm of Section 3.2.

First, we describe the simulation set-up. All simulations are performed using the `ns-2` [69] network simulator. Nodes are configured with omni-directional wireless antennas with physical parameters chosen to match those of the 914 MHz Lucent WaveLAN DSS radios. The data rate is 11 megabits/second. UDP provides network layer services and traffic is created by the `ns-2` Constant Bit Rate (CBR) generators. All topologies consist of $N = 50$ static nodes randomly placed in a 1500 meter by 300 meter area with both the transmission and carrier sense ranges for the radios set to 250 meters. The payload size for all generated packets is 900 bytes.

Each simulated network is loaded with one of four traffic loads: *a few small demands*, *many small demands*, *a few large demands*, or *many large demands*. Traffic loads with a few demands insert traffic at 10 nodes randomly selected from the 50 nodes in the network. Traffic loads with many demands insert traffic at all 50 nodes. Small demands generate traffic at a randomly selected rate of 75 ± 50 packets/second (an effective persistence of 0.06 ± 0.04) while large demands generate traffic at a randomly selected rate of 800 ± 50 packets/second (an effective persistence of 0.64 ± 0.04).

Both control and knowledge of per-node demands are essential to the evaluation of the distributed algorithm. Our objective is an improved understanding of the distributed algorithm and associated TLA allocation. To this end, traffic is constrained to be single hop to avoid complexity introduced by upper network layers and to provide precise control of the traffic demands at all nodes. By changing the number of loaded nodes, varying the magnitude of the demands, and by randomizing demand placement, a wide variety of scenarios are simulated.

Two MAC protocols are simulated: IEEE 802.11, due to its ubiquitous use, and Scheduled TLA-Persistence. In the latter, time is divided into frames, which are divided into v slots. At the start of every frame, each node independently selects a subset of the v slots for transmission. Although slots are selected at random, the number of slots is derived from the current claim made by the node's bidder. Let p_{claim} be the current claim at node i; then node i selects $\lfloor p_{\text{claim}} \cdot v \rfloor + 1$ slots with probability π_i and $\lfloor p_{\text{claim}} \cdot v \rfloor$ slots with probability $1 - \pi_i$ where $\pi_i = (p_{\text{claim}} \cdot v - \lfloor p_{\text{claim}} \cdot v \rfloor)$. Over time, the effective persistence of node i approaches p_{claim}. Each node updates its schedule immediately—mid-frame if necessary—upon any change to its bidder's claim. Traffic demands are converted to equivalent persistences to define $\boldsymbol{w} = (w_1, \ldots, w_M)$. Auctioneer offers and bidder claims are embedded into the MAC header and piggybacked over existing network traffic. Scheduled TLA-Persistence implements acknowledgements similar to those of IEEE 802.11. The slot length is set to 0.0008 seconds, making room for 900 bytes of payload, 72 bytes of MAC header, turn around time for the receiver, and transmission of the return acknowledgement. The retry count is set to ten. Unless specified otherwise, the frame length is 100 slots.

Packets created by the CBR generators do not account for all traffic in the network. For instance, the Address Resolution Protocol (ARP) [3] requires commu-

nication between source and destination nodes during address resolution. Clearly, a node with a persistence of zero cannot send packets to neighbouring nodes. But, this inability suppresses responses to ARP requests, thereby preventing the node from receiving packets as well. A similar problem arises with the distributed algorithm of Section 3.2 where a persistence of zero prevents auctioneer and bidder communication. This does not hinder bidders who desire a persistence of zero. For auctioneers, a persistence of zero is fatal; they must respond to the claims of neighbouring bidders. To resolve this, Scheduled TLA-Persistence employs a network-wide minimum persistence. If a node encounters a transmission slot, but does not have a packet to transmit, it creates a dummy packet (empty payload, valid auctioneer/bidder information) and transmits it. These packets are considered MAC overhead and are *not* included in the delay, throughput, or drop rate reported. (The choice of minimum persistence is discussed in Section 3.4.2.)

For simulations of IEEE 802.11, the maximum packet retry count is set to seven for RTS, CTS, and ACKs and reduced to four for data packets per [45]. The minimum and maximum contention window sizes are set to 32 and 1024 slots, respectively, with the slot length set to 20 microseconds.

3.4.1 Convergence Time

Figure 3.2 plots the expected convergence time and the number of rounds for simulations of 400 random topologies (100 of each traffic load) running Scheduled TLA-Persistence with a minimum persistence of 0.002. The expected convergence times for scenarios with many demands (1.29 seconds for many small demands and 1.37 seconds for many large demands) are significantly smaller than those for scenarios with just a few demands (2.07 seconds for a few small demands and 8.95 seconds for a few large demands).

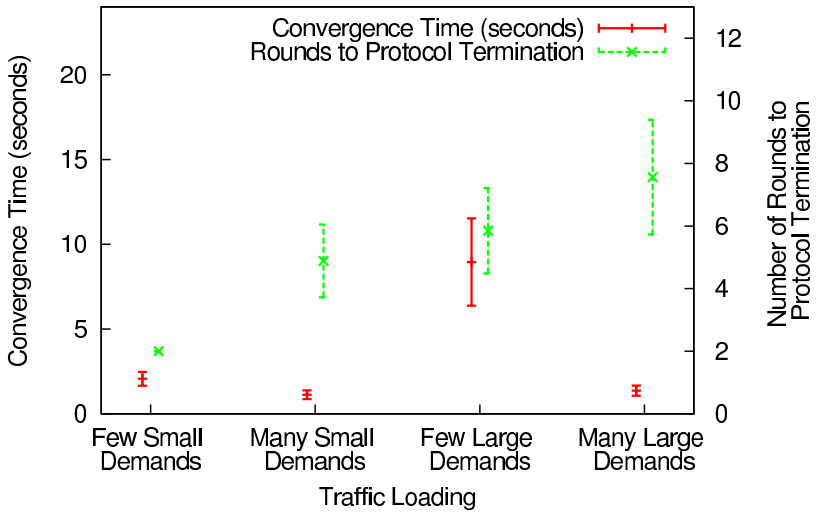


Figure 3.2: Convergence in time and number of rounds for Scheduled TLA-Persistence with minimum persistence of 0.002.

The convergence time for networks with a few demands, small or large, is explained by the small minimum persistence employed by the unloaded nodes. A node operating with a persistence of 0.002 is expected to transmit once every 0.4 seconds (assuming a slot length of 0.0008 seconds). Considering the infrequent communication between nodes, the convergence times are understandable. Scenarios with a few large demands take longer to converge because nodes operating at the minimum persistence must compete with neighbours transmitting at much higher persistences. In contrast, non-loaded nodes in networks with only a few small demands operate in lightly loaded neighbourhoods with very little contention. The nodes may not transmit often; when they do transmit, their communications are more likely to succeed. For scenarios with many demands, all nodes operate with persistences much greater than the minimum persistence, enabling the protocol to converge quickly.

Figure 3.2 shows the number of rounds for protocol termination to be smallest for networks with a few small demands, larger for many small demands, larger still for a few large demands, and largest for many large demands. The number of rounds

is influenced by the number of saturated resources (*i.e.*, bottlenecked auctions) that must be discovered. For networks loaded with a few small demands, it is probable that all resources remain unsaturated, allowing most bidders to become satisfied by their auctioneers' initial offers. If a bidder is not satisfied in the first round, it is almost certainly satisfied by the increased offers of the second round. Of the 100 simulated networks with a few small demands, all 100 terminated in exactly 2 rounds. Networks with many small demands are more heavily loaded and, in dense areas, may saturate resources. Networks with a few large demands are even more likely to saturate resources—any two can saturate a resource. The heavy load of networks with many large demands results in the greatest number of saturated resources, and consequently, the distributed algorithm requires the greatest number of rounds to terminate under this traffic load.

There is an apparent disconnect between expected convergence time and the number of rounds required for protocol termination. The time required for convergence is influenced more by small persistences than it is by the number of rounds.

Figure 3.3 shows observed persistences and persistence errors for nodes in a network loaded with a few large demands. Persistence error for node i is $(s_i - p_i)$ where s_i and p_i represent the TLA allocation and the measured persistence of node i, respectively. Most of the persistence error represents a deficit relative to the TLA allocation occurring in the first two seconds. The TLA persistences, (0.16, 0.16, 0.16, 0.16, 0.16, 0.16, 0.24, 0.24, 0.50, 0.67), are evident in Figure 3.3. Plateaus represent periods of time when a bidder is waiting to hear from auctioneers; abrupt changes identify updated claims by bidders. Although the last node converges at 6.86 seconds, most nodes have converged on the TLA allocation long before then. This time lag suggests the need for a metric that accounts for the magnitude of persistence error.

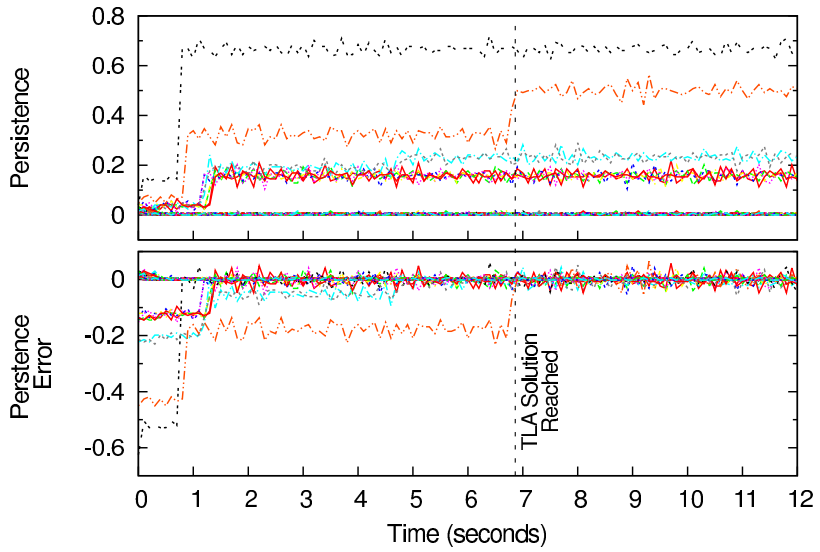


Figure 3.3: Per-node persistence and per-node persistence error for nodes running Scheduled TLA-Persistence with a minimum persistence of 0.002.

A node's relative persistence error, defined as the ratio of persistence error to TLA persistence, measures the fraction of the TLA allocation realized by the node. By computing the geometric average, we characterize a node's expected error during execution of the distributed algorithm.

Figure 3.4 shows expected relative persistence error (for the simulations of Figure 3.2) in three parts: error due to persistences greater than the TLA allocation, error due to persistences smaller than the TLA allocation, and total error due to persistences above and below the TLA allocation. Persistence errors for nodes operating at the minimum persistence are not included in the geometric average. The results show an expected 38% error in networks with a few small demands and between 23% and 25% error for networks loaded with the other three traffic loads. The higher percentage error for networks with few small demands is a consequence of the small number of rounds required to reach the TLA allocation. With an expectation of two rounds (see Figure 3.2), these nodes are more likely to converge on their final persistences during the last round close to protocol termination, driving up the expected relative

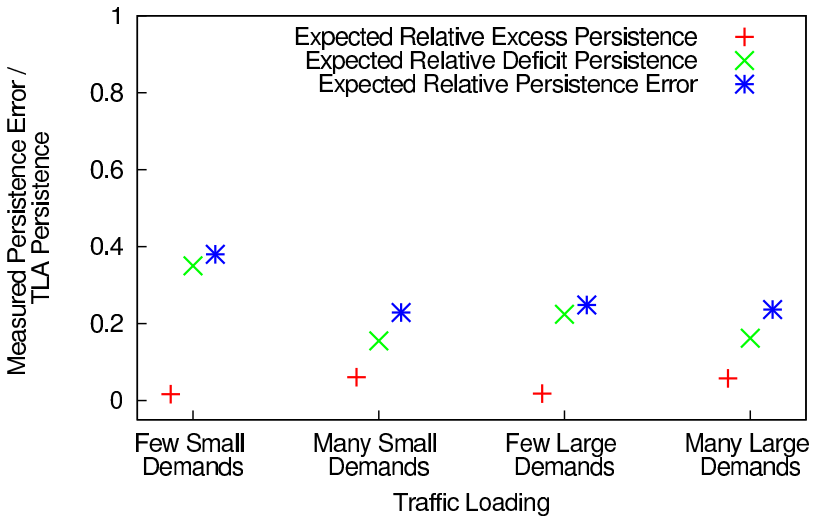


Figure 3.4: Expected persistence error relative to the TLA allocation.

persistence error. The other traffic loads tend to require a greater number of rounds and, in consequence, a smaller proportion of the nodes finalize their persistence during the last round. As a result, nodes have a tendency to reach their final persistence earlier relative to network-wide convergence, reducing the expected relative persistence error.

3.4.2 Convergence Time with Larger Minimum Persistences

Intuitively, a larger minimum persistence drives down the convergence time by allowing auctioneers running at non-loaded nodes to respond more quickly to the claims made by neighbouring bidders. This improvement comes at a cost. The larger minimum persistence must be reserved by nodes with no packets to send at the expense of nodes with packets to send. The result is an overall degradation in system-wide throughput.

Figure 3.5 shows the relationship between minimum persistence, convergence time, and throughput. The statistics are taken from a set of 10 topologies loaded with a few large demands, each topology simulated 30 times with a different minimum persistence ranging from 0.001 to 0.030 in increments of 0.001. Convergence times are

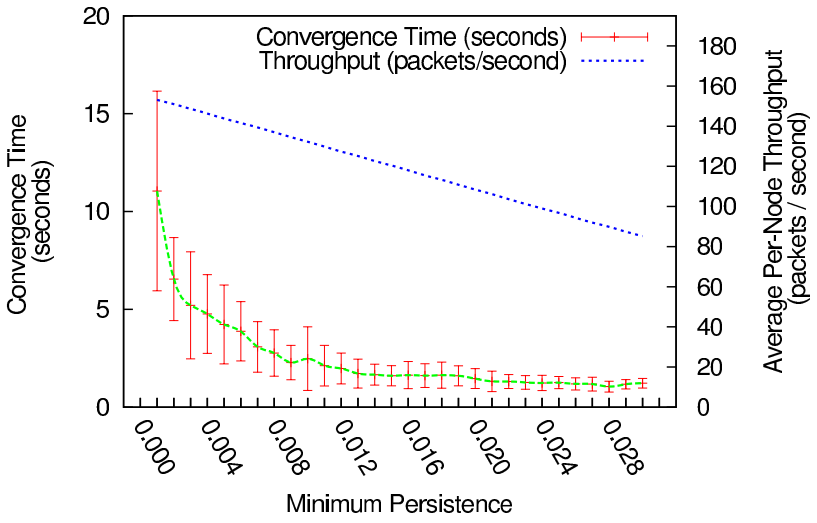


Figure 3.5: Influence of minimum persistence on convergence and throughput for topologies loaded with a few large demands.

measured from time zero to the time the last bidder arrives at the TLA allocation. Throughput measurements are taken for 75 seconds after convergence. The average per-node throughput is computed for nodes loaded with large demands; nodes operating at the minimum persistence are excluded from the calculation.

The simultaneous drop-off in convergence time and gradual contraction in throughput is unique to networks with few large demands. Networks loaded with many demands are not affected by the change to minimum persistence; all nodes in these networks operate above the minimum persistence. For networks with few small demands, a larger minimum persistence reduces the convergence time, but leaves throughput unchanged because allocation to minimum persistences is not made at the expense of other nodes.

The extent to which throughput suffers from allocations made to non-loaded nodes is also a function of the network's expected neighbourhood size. Larger neighbourhoods tend to have more non-loaded nodes contending for access to the channel,

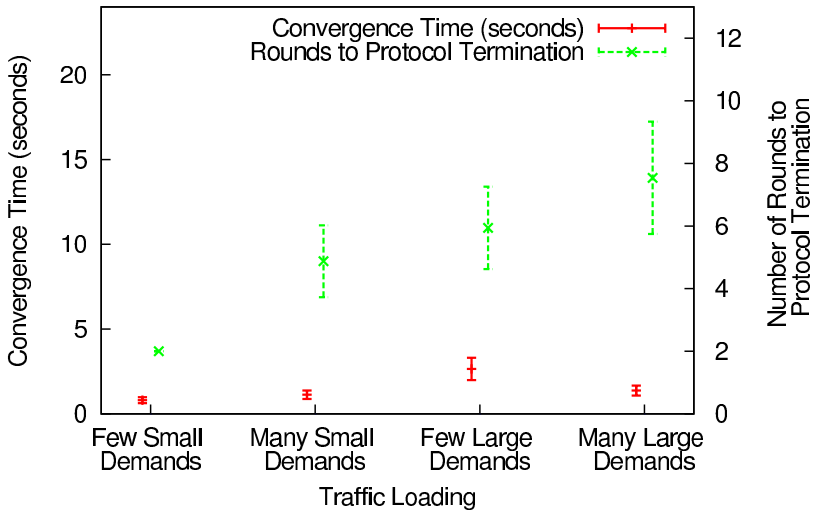


Figure 3.6: Convergence in time and in number of rounds for Scheduled TLA-Persistence with a larger minimum persistence of 0.01.

thereby reducing the allocation available to the remaining nodes. We simulate fairly dense topologies, accentuating this behaviour.

From the results in Figure 3.5, we conclude that a minimum persistence of 0.01 is expected to reduce the average convergence time to 2.48 seconds while keeping expected throughput at 132 packets/second (a drop of 21 packets/second from its peak). To verify this, the 400 topologies used in the simulations of Figure 3.2 are simulated again, this time with a minimum persistence of 0.01. All other MAC parameters are held constant. Results are shown in Figure 3.6. While the expected number of rounds does not change, the convergence time is reduced from 2.07 seconds to 0.80 seconds for networks with few small demands and from 8.95 seconds to 2.65 seconds for networks with few large demands. As expected, convergence times for networks with many demands did not change.

3.4.3 Convergence Time with Larger Negotiation Persistences

The communication needs of the distributed algorithm constitute a set of demands that are not represented in the desired persistences passed down to the MAC layer. While an increased minimum persistence meets these demands indirectly, the demands are temporary, resulting in a long-term waste of the channel. Here, we examine the use of *negotiation persistences* which are determined according to neighbourhood size and are used only during execution of the distributed algorithm. The negotiation persistence of node i is $1/X$ where X is the size of the largest neighbourhood in which i participates. Each node starts with its negotiation persistence and then transitions to its TLA persistence when (1) it has determined its own TLA persistence and (2) all nodes within its one-hop neighbourhood have determined their TLA persistences. Upon termination of the distributed algorithm, the TLA allocation must maintain a network-wide minimum persistence in support of network functions such as ARP. Use of negotiation persistences permits a smaller minimum persistence without degrading the distributed algorithm's time to convergence.

Figure 3.7 shows the expected convergence times for the distributed algorithm when configured to use negotiation persistences and run on the 400 topologies and traffic loads of Figure 3.2. The minimum persistence is 0.002. The expected number of rounds remains unchanged. Expected convergence times are consistently small (0.60 seconds for a few small demands, 1.05 seconds for many small demands, 1.07 seconds for a few large demands, and 1.33 seconds for many large demands).

The traces in Figure 3.8 show persistence, persistence error, and neighbourhood over-allocation for Scheduled TLA-Persistence when configured to use negotiation persistences. The topologies and traffic loads of Figure 3.3 are reused. The minimum persistence is 0.002. The negotiation persistences are clearly visible during the first

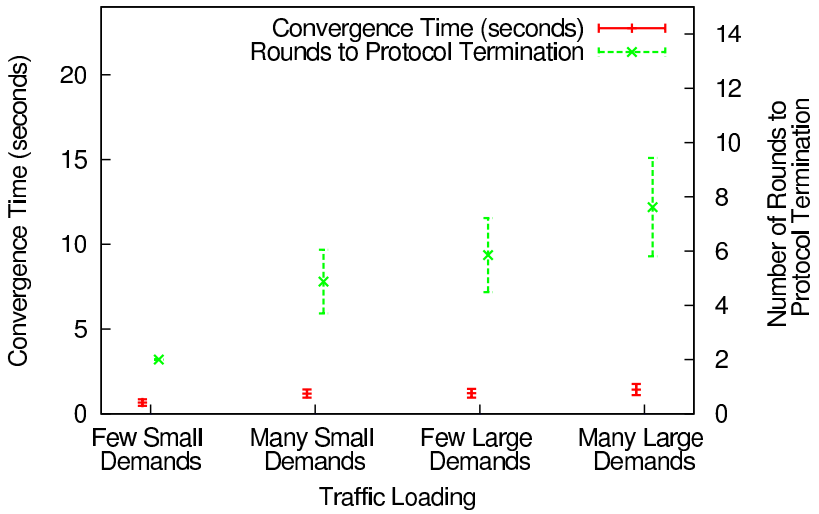


Figure 3.7: Convergence in time and in number of rounds for Scheduled TLA-Persistence configured to use negotiation persistences.

second of simulation in the persistence and persistence error traces. While some of the negotiation persistences are smaller than their corresponding TLA persistences, they are much larger than the minimum persistence and show up as excess persistence in the persistence error trace. The excess in persistence resolves itself by the time the TLA solution is reached at 1.39 seconds.

The transition from negotiation persistences to TLA persistences is only loosely synchronized; neighbouring nodes may not transition to their TLA persistences at the exact same time. During the transition, over-allocation of the channel is possible as is evident in the over-allocation trace of Figure 3.8 during the interval starting at approximately 0.8 seconds through 1.39 seconds. The bidder claims are designed to prevent over-allocation; therefore, over-allocation only occurs when negotiation persistences are used.

Expected relative persistence error for the simulations of Figure 3.7 is shown in Figure 3.9. Negotiation persistences drive up relative persistence error for networks with a few large demands. While these persistences are large compared to the minimum

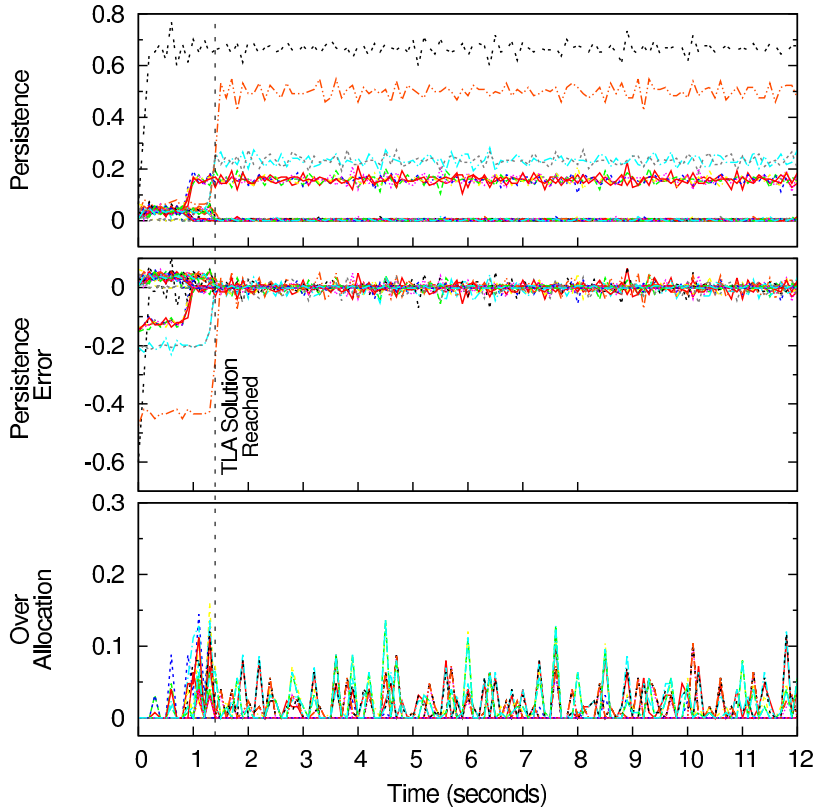


Figure 3.8: Per-node persistence, persistence error, and over-allocation for Scheduled TLA-Persistence configured to use negotiation persistences.

persistence, they are small compared to the large allocations that are common in networks with only a few large demands. The amplified difference between the negotiation persistences and the TLA persistences under this traffic load exacerbates the measured error. TLA persistences for the other three traffic loads are smaller and therefore more similar to the negotiation persistences; the result is a smaller relative persistence error.

3.4.4 Accuracy of Distributed Algorithm

To validate the accuracy of the distributed algorithm, calculated persistences are compared against values generated by the centralized algorithm. Absolute error is $|p_{\text{dist}} - p_{\text{cntrl}}|$ where p_{cntrl} and p_{dist} are the persistences derived from the centralized and

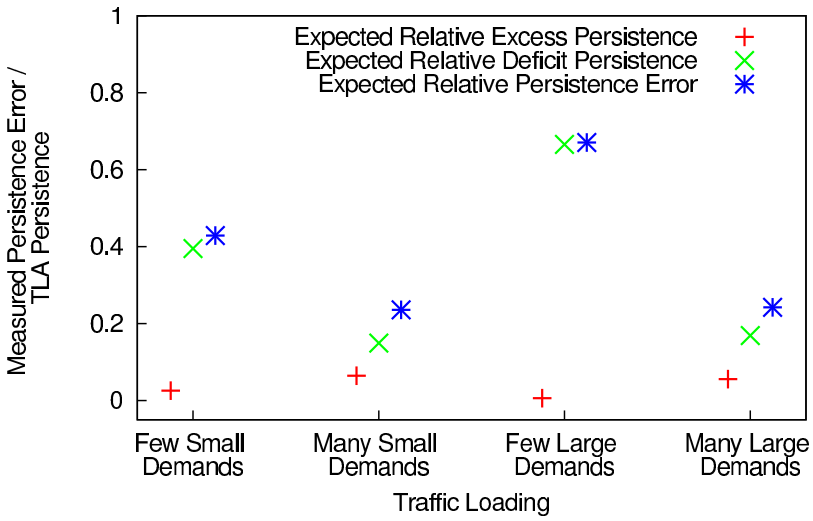


Figure 3.9: Expected persistence error relative to the TLA allocation for Scheduled TLA-Persistence configured to use negotiation persistences.

distributed algorithms, respectively. Table 3.1 reports observed errors for precision levels ranging from 5 to 30 bits. The data in the table are taken from simulations of 400 random topologies—100 of each traffic load—running Scheduled TLA-Persistence with a minimum persistence of 0.01.

The accuracy of the distributed algorithm improves as bits are added to the persistence representation. The real-valued auctioneer offers are mapped into the persistence representation rounding their offers up. The expected rounding error for a single mapping from a real-valued persistence into a β-bit discrete persistence representation is one half of $1/2^{\beta}$. The expected rounding error is shown in column 3 of Table 3.1 to be compared against the average errors in column 4. The average absolute error tracks the theoretical expected rounding error, staying within an order of magnitude.

The large maximum errors shown in column 6 can be explained through a simple, yet extreme, example. Consider an auction with b bidders. One bidder has a demand of 1; the remaining bidders have demands less than $1/b$. During the first

round, the auctioneer offers an allocation of $1/b$, rounded up to the nearest persistence level in the representation. The $b-1$ bidders with small demands become satisfied immediately and their first and final claims assume the rounding error introduced by the auctioneer, claiming a maximum of $1/2^{\beta}$ more than their TLA allocation. To prevent over-allocation of the channel, the auctioneer reduces subsequent offers, so the final offer made to the one remaining bidder is $(b-1)/2^{\beta}$ smaller than its TLA allocation. With 50 nodes, the theoretical maximum for absolute error is bounded by $49/2^{\beta}$. These numbers are included in column 5 to compare to the maximum observed errors of column 6.

The final two columns of Table 3.1 show the observed convergence times and expected number of rounds to protocol termination. For representations with 10 or more bits of precision, the convergence time and rounds to protocol termination remain relatively unaffected. For representations with 9 and fewer bits, a marked decrease is observed for both measurements. Offers that are close to, but not exactly, equal are mapped to the same level in the persistence representation, simplifying the computation and reducing the number of rounds. Fewer rounds means shorter convergence time.

At 10 bits of precision, the average observed rounding error is 0.0008 with a maximum observed error of 0.0274. For the purposes of channel allocation these errors seem acceptable. All simulations, except those of Table 3.1, use a 10-bit representation for persistences.

3.4.5 Frame Length and Persistence Stability

Figure 3.10 demonstrates the relationship between frame length and variation in persistence, corroborating the analytical results reported in [29]. Persistence traces

Table 3.1: Accuracy of the distributed computation.

Persistence Representation (number of bits)	**Number of Persistence Levels**	**Expected Absolute Error**		**Maximum Absolute Error**		**Expected Convergence Time (seconds)**	**Expected Rounds to Termination**
		Theoretical	**Observed**	**Theoretical**	**Observed**		
5	2^5	1.56×10^{-2}	2.27×10^{-2}	1.53×10^{0}	3.25×10^{-1}	0.71	2.89
6	2^6	7.81×10^{-3}	8.54×10^{-3}	7.66×10^{-1}	2.78×10^{-1}	0.98	3.56
7	2^7	3.91×10^{-3}	6.05×10^{-3}	3.83×10^{-1}	1.29×10^{-1}	1.07	4.18
8	2^8	1.95×10^{-3}	2.19×10^{-3}	1.91×10^{-1}	6.53×10^{-2}	1.32	4.63
9	2^9	9.77×10^{-4}	1.65×10^{-3}	9.57×10^{-2}	3.03×10^{-2}	1.37	4.87
10	2^{10}	4.88×10^{-4}	7.99×10^{-4}	4.79×10^{-2}	2.74×10^{-2}	1.43	5.01
11	2^{11}	2.44×10^{-4}	3.19×10^{-4}	2.39×10^{-2}	1.32×10^{-2}	1.50	5.01
12	2^{12}	1.22×10^{-4}	1.14×10^{-4}	1.20×10^{-2}	6.17×10^{-3}	1.49	5.05
13	2^{13}	6.10×10^{-5}	5.89×10^{-5}	5.98×10^{-3}	2.70×10^{-3}	1.50	5.03
14	2^{14}	3.05×10^{-5}	2.41×10^{-5}	2.99×10^{-3}	1.24×10^{-3}	1.48	5.04
15	2^{15}	1.53×10^{-5}	1.48×10^{-5}	8.56×10^{-4}	8.56×10^{-4}	1.50	5.04
16	2^{16}	7.63×10^{-6}	1.02×10^{-5}	7.48×10^{-4}	4.14×10^{-4}	1.48	5.01
17	2^{17}	3.81×10^{-6}	3.81×10^{-6}	3.74×10^{-4}	1.93×10^{-4}	1.48	5.04
18	2^{18}	1.91×10^{-6}	2.78×10^{-6}	1.87×10^{-4}	9.18×10^{-5}	1.48	5.05
19	2^{19}	9.54×10^{-7}	7.80×10^{-7}	9.35×10^{-5}	4.42×10^{-5}	1.47	5.05
20	2^{20}	4.77×10^{-7}	4.71×10^{-7}	4.67×10^{-5}	2.68×10^{-5}	1.48	5.05
21	2^{21}	2.38×10^{-7}	3.29×10^{-7}	2.34×10^{-5}	1.29×10^{-5}	1.47	5.06
30	2^{30}	4.66×10^{-10}	6.00×10^{-10}	4.56×10^{-8}	1.89×10^{-8}	1.48	5.03

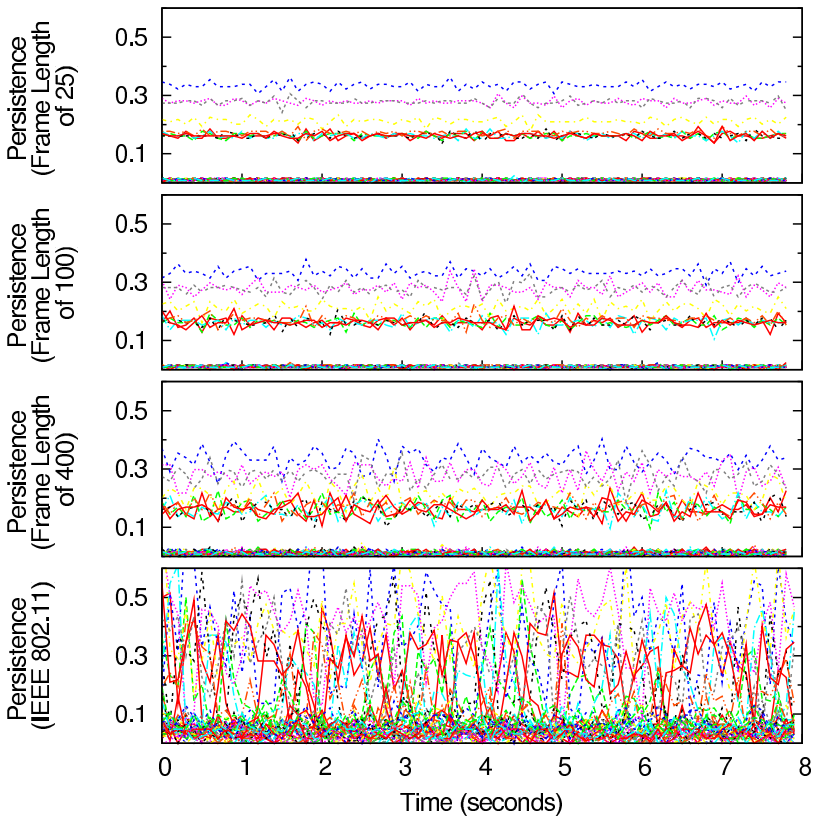


Figure 3.10: Persistences for Scheduled TLA-Persistence and 802.11.

are shown from four simulations of the same topology and traffic load. The top three traces show persistence for the Scheduled TLA-Persistence MAC with frame lengths of 25, 100, and 400 slots. The bottom trace shows persistences for the IEEE 802.11 MAC. Persistences are estimated over 0.1 second intervals. A noticeable increase in persistence variation is seen when running with the longer frame lengths. At a frame length of 25, the observed persistences vary only slightly; at a frame length of 400, variation increases allowing the measured persistences to run together. Regardless of frame length, the observed persistences of the Scheduled TLA-Persistence MAC are far less variable than those of IEEE 802.11.

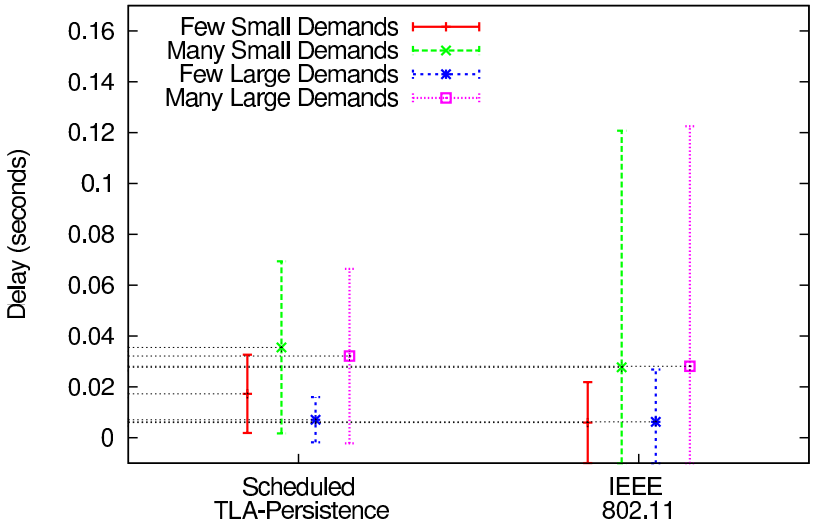
3.4.6 Comparison with IEEE 802.11

When compared to IEEE 802.11, Scheduled TLA-Persistence performs surprisingly well, especially considering the added overhead of a network wide minimum persistence of 0.01. Figure 3.11 shows delay, throughput, and drop rate for Scheduled TLA-Persistence compared to IEEE 802.11. Measurements are taken from simulations of 120 different networks, each with a randomly generated topology and traffic load—30 of each traffic load. Each network is simulated once using Scheduled TLA-Persistence and then using IEEE 802.11. Delay, throughput, and drop rate measurements are taken over 20 seconds following a 5 second network warm-up period. The warm-up period begins at time zero for simulations of IEEE 802.11 and at convergence to the TLA allocation for Scheduled TLA-Persistence.

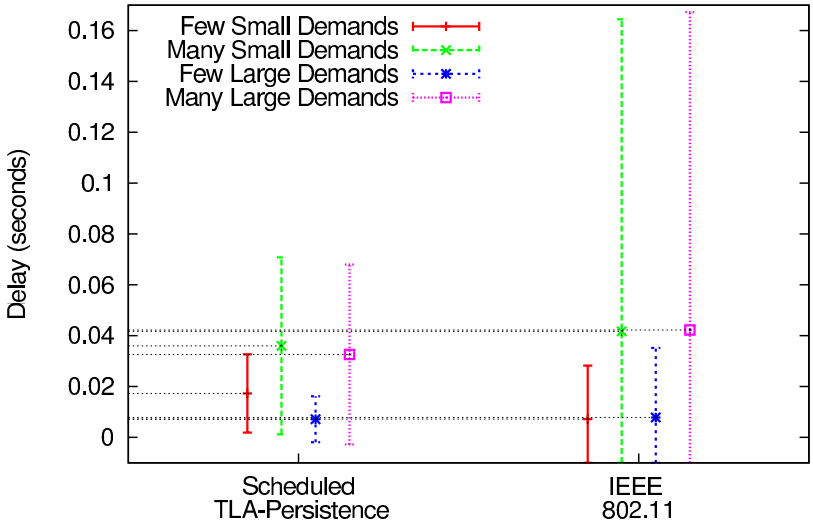
Comparing Delay

Two forms of delay are reported in Figure 3.11. The first measures delay for successful packet transmissions only. Measuring delay in this manner under-estimates delay in networks with a significant packet drop rate. The second form of delay accounts for dropped packets by including delay for successful and unsuccessful packet transmissions. Delay for an unsuccessful packet transmission is arbitrarily defined to be the time the packet spends queued for transmission at the MAC layer before being dropped.

We start with a discussion of delay calculated over successful packet transmissions. For networks with only a few small demands, Scheduled TLA-Persistence has an expected delay of 0.0173 seconds for successful packets, nearly 3 times that of IEEE 802.11. Lightly loaded networks lend themselves to the greedy approach of IEEE 802.11. With only a few small demands spread out over the network, collisions are unlikely and most packets are transmitted immediately upon receipt from the network

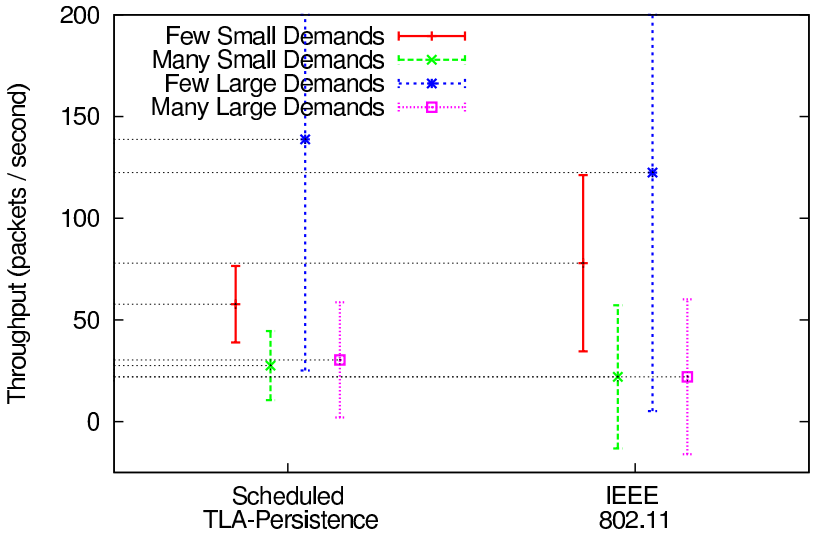


(a) Average Delay (Successful Packets).

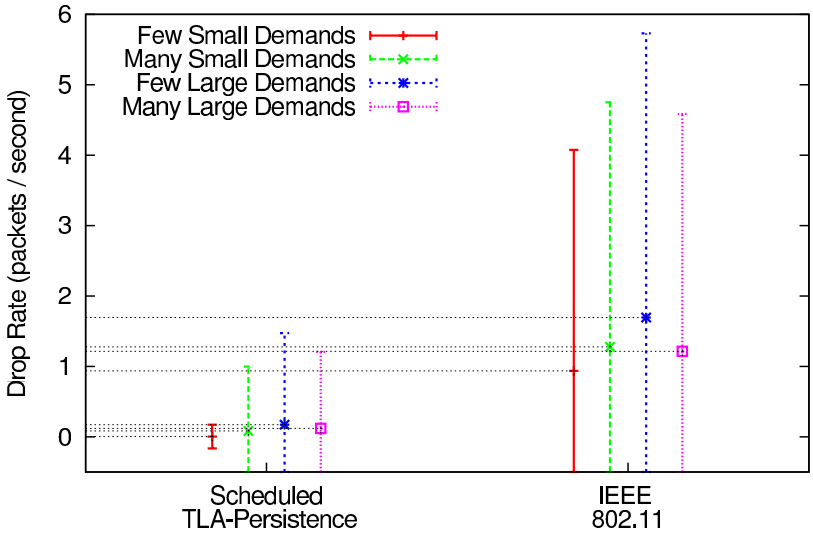


(b) Average Delay (All Packets).

Figure 3.11: Comparing Scheduled TLA-Persistence (frame length of 100 and minimum persistence of 0.01) with IEEE 802.11.



(c) Average Per-Node Throughput.



(d) Average Per-Node Drop Rate.

Figure 3.11: Continued from page 60.

layer. In contrast, Scheduled TLA-Persistence waits until the next transmission slot to transmit a packet. Assuming a persistence of 0.06 and a slot length of 0.0008 seconds, the expected time between transmission slots is 0.013 seconds, accounting for nearly all of the packet delay measured.

Both MAC protocols exhibit higher delay when loaded with many small demands. The added congestion increases the likelihood of collisions, driving up the number of retransmissions necessary for successful packet delivery. Retransmissions mean added delay. Scheduled TLA-Persistence has an expected delay of 0.0356 seconds for successful packets—29% greater than IEEE 802.11. While the average MAC layer delay for successful transmissions for Scheduled TLA-Persistence is longer than that of IEEE 802.11, it has a 64% smaller variation in delay, which is striking.

For networks loaded with few large demands, Scheduled TLA-Persistence has an expected delay of 0.0071 seconds, 15% larger than the 0.0062 seconds measured for IEEE 802.11. The larger demands result in larger TLA persistences and a marked drop in delay compared to networks with few small demands. The standard deviation in the delay measurements for the Scheduled TLA-Persistence MAC is 0.0089 seconds, roughly 43% that of IEEE 802.11.

Finally, when loaded with many large demands, Scheduled TLA-Persistence has an expected delay of 0.0321 seconds, 15% larger than the delay of IEEE 802.11; standard deviation in delay for Scheduled TLA-Persistence is 0.0343 seconds, 63% smaller than it is for IEEE 802.11.

The two forms of delay when measured over simulations of Scheduled TLA-Persistence are almost identical, suggesting a high success probability for MAC layer communication. In contrast, inclusion of unsuccessful packets in the delay calculation for IEEE 802.11 reveals a marked increase in both delay expectation and variation

in delay for all four traffic loads. For three of the network loads (all but a few small demands), the expected delay for all packets is larger for IEEE 802.11 than for Scheduled TLA-Persistence, reflecting a tendency for IEEE 802.11 to drop packets.

Comparing Throughput

Scheduled TLA-Persistence shows a distinct advantage over IEEE 802.11 with higher throughput rates reported for all but one of the four traffic loads. As was seen with delay, IEEE 802.11 performs very well on lightly loaded networks with few small demands; its expected per-node throughput is 77.88 packets/second, a 34% improvement over that of Scheduled TLA-Persistence. The smaller throughput achieved by Scheduled TLA-Persistence comes from scheduled channel access, where each node maintains its target persistence without regard to the number of retransmissions necessary to deliver a packet. When the magnitude of a node's traffic demand matches the persistence of the MAC layer, a packet retransmission by Scheduled TLA-Persistence consumes a transmission slot "intended" for another packet. Because the MAC protocol does not increase its persistence to make up for the lost transmission opportunity, the packet retransmission represents a loss in overall throughput. The inherent flexibility of IEEE 802.11 allows it to dynamically increase its persistence to accommodate packet retransmissions, minimizing loss of throughput due to collisions in lightly loaded networks.

Scheduled TLA-Persistence fares better in terms of expected per-node throughput for the other three traffic loads with an improvement of 25% over IEEE 802.11 for networks with many small demands, 13% for a few large demands, and 38% for many large demands. When compared to IEEE 802.11, the standard deviation in throughput is reduced to 43% when loaded with a few small demands, 48% for many small demands, 97% for a few large demands; and 74% for many large demands. The

large variation in throughput for Scheduled TLA-Persistence in networks with a few large demands is an artifact of the traffic load. It is possible for some neighbourhoods to contain a single loaded node while others contain several loaded nodes; neighbourhoods with a single loaded node assign the greater portion of channel capacity to that one node; neighbourhoods with several loaded nodes must divide the channel capacity equally among contending nodes. The result is a large variation in persistence within the TLA allocation. IEEE 802.11 has no notion of TLA persistence; its variation in throughput comes from dynamically changing per-node persistences (see Figure 3.10).

Comparing Drop Rate

Perhaps the most dramatic improvement in Scheduled TLA-Persistence is the drastically reduced packet drop rate. Compared to IEEE 802.11, Scheduled TLA-Persistence rarely drops packets. Networks with a few large demands result in the highest expected drop rate of 0.171 packets/second, roughly 10% that of IEEE 802.11. It also cuts variation in drop rate by a minimum of 67% and as much as 94% for networks with few small demands where IEEE 802.11 favours some nodes at the expense of others, resulting in an exaggerated drop rate. To put this in context, consider a drop rate of 4.5 packets/second (*i.e.*, expected drop rate plus one standard deviation). For a node loaded with 75 packets/second, an expected 6% of packets queued for transmission are lost.

Variation in Delay and Throughput

The delays reported in Figure 3.11 are organized by MAC protocol and traffic load, masking the behaviour of individual nodes by combining delay measurements for packets sent by nodes from 30 different networks. Figure 3.12 plots delays for the same set of simulations, but with packet delays organized by transmitting node. The result

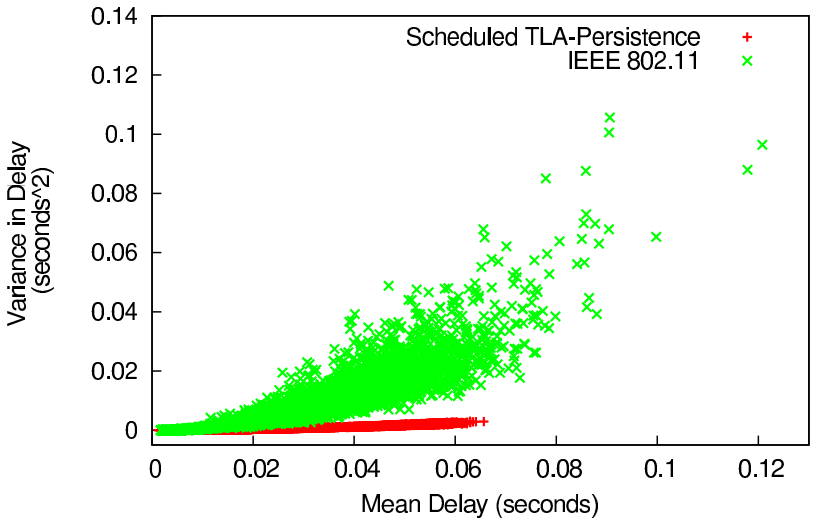


Figure 3.12: Delays for Scheduled TLA-Persistence and IEEE 802.11.

is a scatter plot where each data point communicates expectation (x-coordinate) and variation (y-coordinate) in delay for exactly one node. Scheduled TLA-Persistence demonstrates a remarkable ability to control variation in delay for packets sent from a common node. The largest reported variation in delay is 0.0031 seconds^2, nearly two orders of magnitude smaller than the maximum 0.106 seconds^2 reported for IEEE 802.11.

Figure 3.13 shows a similar scatter plot for throughput by transmitting node. Each data point identifies expectation (x-coordinate) and variation (y-coordinate) for throughput at a single node. Throughput is estimated over 0.1 second intervals. As seen with delay, Scheduled TLA-Persistence proves adept in the control of variation in throughput achieved by each node. The maximum variation in throughput for Scheduled TLA-Persistence is 2200 packets^2/second^2, less than 6% of the 29,000 packets^2/second^2 measured for IEEE 802.11.

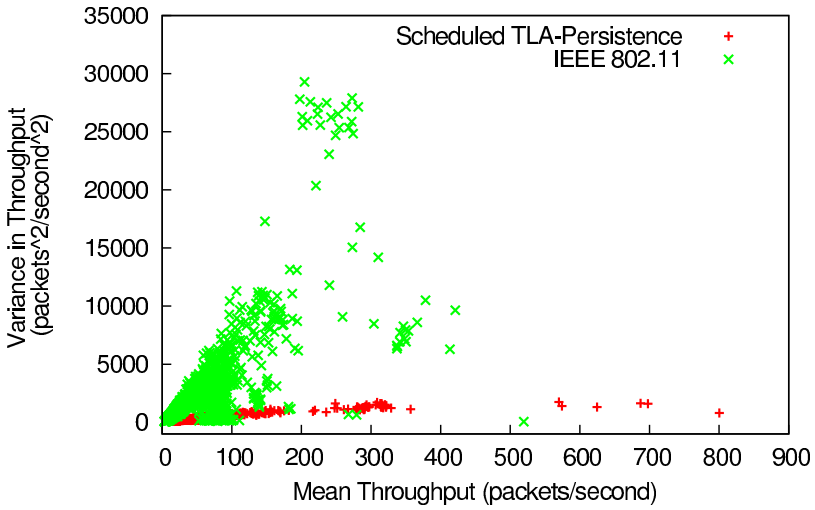


Figure 3.13: Throughputs for Scheduled TLA-Persistence and 802.11.

3.5 Discussion

The simulation results in Section 3.4 show the distributed algorithm to be well suited for use in homogeneous networks with fixed topologies. The decentralized nature and minimal overhead make it feasible to compute TLA persistences for any network topology or traffic load. Here we discuss consequences and remaining concerns.

3.5.1 Obstacles to Adaptation

There are several obstacles to the distributed algorithm's deployment in real world wireless networks. One is its requirement for nodes to have a priori knowledge of their neighbours. In practice, a method must be provided for neighbour discovery. A second obstacle is the distributed algorithm's reliance on round synchronization. The distributed algorithm provides its own synchronization, but it assumes that all auctioneers and bidders are started simultaneously. In an implementation, either the algorithm's synchronization requirements must be weakened, or a method provided to synchronize. A third obstacle is the distributed algorithm's assumption of a static topology. Once initialized, each auctioneer assumes a fixed set of bidders and each

bidder a fixed set of auctioneers. Changes occurring after algorithm initialization are ignored, resulting in a channel allocation that may not match the TLA allocation defined for the new topology. In practice, the algorithm must accommodate topology changes. A fourth obstacle is the distributed algorithm's assumption of static desired persistences, determined for all time prior to algorithm execution. In an actual wireless network, the desired persistences change throughout the life of the network; the algorithm must accommodate these changes.

These obstacles suggest potential next steps for the design of an adaptive algorithm capable of operating in networks with dynamic topologies and traffic load.

1. The distributed algorithm should be modified to perform neighbour discovery for itself, detecting both new and lost neighbours.

2. The synchronization requirement imposed by the distributed algorithm should be weakened. A fully asynchronous approach would obviate the need for synchronization.

3. The distributed algorithm should adjust to changes in both topology and traffic load, automatically converging toward the latest TLA allocation. The impact of any network change should be limited to nodes whose persistences need updating to match the TLA allocation. Ideally, other nodes should remain unaffected.

4. The convergence time for the distributed algorithm should be further reduced. Faster convergence improves the algorithm's responsiveness to network changes, making it relevant to networks with high mobility or bursty traffic.

Considering the obstacles outlined, the distributed algorithm should not be viewed as a full solution to the challenge of distributed computation of the TLA

allocation in mobile wireless networks; rather, it is an important and useful step toward such a solution. The already fast convergence time, intuitive structure, computational simplicity, and minimal communication overhead make the distributed algorithm a promising place to start in the design of a fully adaptive solution. In Chapter 4, the distributed algorithm is extended to resolve all four obstacles mentioned here.

3.5.2 The TLA Allocation and Fairness

The TLA allocation is lexicographically max-min to transmitters of the network. Such an allocation has been extensively explored as a fairness metric for single-hop and end-to-end flows; its use as a MAC allocation strategy represents a new approach to wireless channel access.

Perhaps most closely related is *Single-hop flow-fairness* which concerns fair allocation of the channel to flows defined at the MAC layer traversing a single hop. Modelling single-hop flows as edges in an undirected multigraph enables one to determine a flow contention graph, whose cliques represent regions of possible contention [67, 43, 51]. These do not take into account the asymmetry of flows. Max-min fair allocation to directed flows is treated in [88, 91, 92].

3.5.3 Achieving Single-hop Flow-Fairness

The difference between single-hop flow-fairness and the TLA allocation results from a fundamental distinction: Are transmitters or single-hop flows to be treated equally? In focussing on the MAC problem, the TLA allocation ensures the equal treatment of transmitters. But, the distributed algorithm is not limited to the implementation of the TLA allocation. It is a distributed method capable of solving a general lexicographic max-min optimization problem. Consider the application to single-hop flow-fairness, defined as follows. The resources are the receivers at each node. The demands

are the single-hop flows. Auctioneers manage the allocation offered by receivers; bidders manage the allocation claimed for single-hop flows. The resulting allocation is lexicographically max-min to single-hop flows.

3.5.4 Single-Hop Flow-Fairness as a MAC Allocation Scheme

On one hand, once traffic is offered and admission control, flow control, and routing decisions are made, it may be desirable to provide fair access for each single-hop flow. On the other hand, each of these higher-layer decisions is impacted by the access of each transmitter to the channel; in making these decisions, it is desirable to know what persistence each transmitter is permitted, in addition to knowing what traffic is routed from or through it. This circularity, with flows depending on persistences and persistences depending on flows, creates a challenging cross-layer optimization problem not easily solved by treating any layer in isolation.

Imposing persistences based on single-hop flow-fairness may fail to lead to fairness for applications, because some may set up many end-to-end flows while others use one; by routing through a congested node, it may adversely impact the total channel utilization; and when flows are bursty, they require frequent changes in the persistences. Persistences based on transmitter-fairness can inform higher layers of the opportunities to transmit, thereby supporting routing, flow control, and admission decisions. Indeed, setting all demand magnitudes to 1, transmitter-fairness becomes a function of topology alone, and serves as a measure of network capacity.

Our objective is the effective determination of persistences that permit each transmitter an amount of channel access that does not disadvantage any of its neighbours, which may diverge substantially from single-hop flow-fairness.

3.5.5 Fairness and Quality of Service

In practice, it may be desirable to allow nodes to have different entitlements to channel access, so that a node carrying more flows has more right to channel access. This is accommodated by a *weighted TLA allocation*: each transmitter is assigned a *weight*, $(\gamma_1, \gamma_2, \ldots, \gamma_M)$, and the allocation vector is chosen so that $(\gamma_1 s_1, \gamma_2 s_2, \ldots, \gamma_M s_M)$ is lexicographically max-min. If we denote by n_i the number of single hop flows initiated at transmitter i and set $\gamma_i = 1/n_i$, the allocation is single-hop flow-fair.

The weighted TLA allocation is not limited to weighting by the number or total demand of flows initiated at a transmitter. Indeed, the upper layers of the network stack are free to select the weights according to topological properties such as neighbourhood size and node betweenness [35], or they can be driven in response to differentiated traffic in support of quality of service. Accommodating extensions to the weighted TLA allocation in the auction algorithms is straightforward, and may provide a mechanism for accommodating both standard fairness requirements and differentiated traffic.

The dramatically reduced variation in packet delay and throughput for Scheduled TLA-Persistence motivates its use in both delay sensitive and resource constrained networks. In delay sensitive networks, packets with a large delay may be unusable by the application; example applications include voice, video, and other real-time constrained communications. In resource constrained networks, nodes may have insufficient buffer space to handle spikes in throughput resulting in packet loss due to buffer overruns. In either case, low variation in delay and throughput increases the percentage of usable packets.

The consistency offered by Scheduled TLA-Persistence also enables prediction of future behaviour with greater accuracy. Even if throughput and delay vary from

node to node, consistent behaviour of individual nodes allows the network to better characterize its path delays and localized capacity. This information can be passed up the network stack and used by routing, admission control, and resource reservation protocols. In order to be useful, this information must be accurate. The high variation in delay and throughput experienced at nodes running IEEE 802.11 make it impossible to accurately predict delay or throughput. Scheduled TLA-Persistence, on the other hand, maintains predictable delay and throughput characteristics, enabling the use of MAC layer performance measurements by higher layer functions.

3.5.6 MAC Protocol Design

We have argued that TLA persistences set target occupancies for transmitters. Naturally the question is how to use these persistences, once computed. We have used them in Scheduled TLA-Persistence. Certainly any MAC protocol that employs explicit persistences can employ the TLA allocation directly. There remain protocols that do neither, for example contention-based schemes with a back-off mechanism. Even for these, TLA persistences determine how long to wait on average, and may provide a useful tool in tuning the back-off mechanism.

3.6 Summary

In this chapter we have proposed the TLA allocation as a target channel allocation strategy for multi-hop wireless networks. The allocation defines target persistences that can inform each node's answer to the question: When can I transmit next? To be used in an ad hoc network, the TLA computation must be computed in a decentralized and adaptive manner. Although the distributed algorithm presented in this chapter computes the TLA allocation, it requires loose synchronization and

cannot accommodate changes to the topology or traffic load. Chapter 4 presents a fully adaptive and asynchronous computation of the TLA allocation.

Chapter 4

ADAPTIVE TOPOLOGY- AND LOAD-AWARE SCHEDULING

The focus of this chapter is on the distributed and adaptive computation of the TLA allocation. We improve on the distributed algorithm discussed in Chapter 3 to create an algorithm that does not require synchronization, performs its own neighbour discovery, and adapts continuously to changes in topology and traffic load. The algorithm is integrated into an Adaptive Topology- and Load-Aware Scheduled (ATLAS) MAC protocol and evaluated through simulation. Simulation results show ATLAS to adapt quickly to changes in the network supporting highly mobile nodes and multi-hop TCP flows.

The remainder of this chapter is organized as follows: Section 4.1 describes a distributed algorithm to compute the general lexicographic max-min allocation, emphasizing that the algorithm may be applicable beyond the computation of TLA persistences. Its correctness is proved. Section 4.2 expresses channel allocation in terms of a resource allocation problem and discusses implementation choices to be made when integrating the distributed algorithm into ATLAS. After describing the simulation set-up in Section 4.3, Section 4.4 studies how ATLAS adapts to controlled changes in topology and load, and to dynamic network conditions. In Section 4.5, we discuss potential applications of the distributed algorithm with an emphasis on how ALTAS supports the design of higher-layer services that inform, and are informed by, the underlying communication network.

4.1 Distributed Resource Allocation

In this section we describe a distributed algorithm to compute the lexicographic max-min allocation. In it, each resource is represented by an auctioneer and each

demand by a bidder. Auctioneer j runs one auction to distribute capacity at resource j; bidder i claims resources for demand i.

Bidder i knows w_i and maintains set R_i. Offers are stored in *offers*[]; *offers*[j] holds the offer last received from auctioneer j. Bidder i constrains its *claim* to be no larger than w_i or the smallest offer from auctioneers in R_i,

$$\mathit{claim} = \min\left(\{\mathit{offers}[j] : j \in R_i\}, w_i\right). \tag{4.1}$$

Auctioneer j knows c_j and maintains set D_j. Bidder claims are stored in *claims*[]; *claims*[i] holds the claim last received from bidder i. Auctioneer j identifies set $D_j^* \subseteq D_j$ containing bidders with claims strictly smaller than its offer,

$$D_j^* = \{b : b \in D_j, \mathit{claims}[b] < \mathit{offer}\}. \tag{4.2}$$

Bidders in D_j^* are constrained elsewhere and cannot increase their claims in response to a larger offer from auctioneer j. Bidders in $D_j \setminus D_j^*$ are constrained by auction j. They may increase their claims in response to a larger offer. Resources left unclaimed by bidders in D_j^*,

$$\mathcal{A}_j = c_j - \left(\textstyle\sum_{i \in D_j^*} \mathit{claims}[i]\right), \tag{4.3}$$

are offered in equal portions to bidders in $D_j \setminus D_j^*$. If claims of all bidders in D_j are smaller than the offer (*i.e.*, $D_j = D_j^*$), there are no bidders to share resources in $\mathcal{A}_j$. The auctioneer sets its offer to $\mathcal{A}_j$ plus the largest claim, ensuring that any adjacent claim can be increased to consume resources in $\mathcal{A}_j$:

$$\mathit{offer} = \begin{cases} \mathcal{A}_j / |D_j \setminus D_j^*|, & \text{if } D_j \neq D_j^*, \\ \mathcal{A}_j + \max\left(\mathit{claims}[i] : i \in D_j\right), & \text{otherwise.} \end{cases} \tag{4.4}$$

Algorithm 4.1 and Algorithm 4.2 describe actions taken by bidders and auctioneers in response to externally triggered events. Collectively, auctioneers and bidders

Algorithm 4.1 Bidder for Demand i.

```
1: upon initialization
2:     R_i ← ∅
3:     w_i ← 0
4:     UpdateClaim()
5: end upon
6: upon receiving a new demand magnitude w_i
7:     UpdateClaim()
8: end upon
9: upon receiving offer from auctioneer j
10:    offers[j] ← offer
11:    UpdateClaim()
12: end upon
13: upon bidder i joining auction j
14:    R_i ← R_i ∪ j
15:    UpdateClaim()
16: end upon
17: upon bidder i leaving auction j
18:    R_i ← R_i \ j
19:    UpdateClaim()
20: end upon
21: procedure UpdateClaim()
22:    claim ← min({offers[j] : j ∈ R_i} ∪ w_i)
23:    send claim to all auctions in R_i
24: end procedure
```

know the inputs to the allocation problem. Each bidder i maintains w_i and R_i; each auctioneer j maintains c_j and D_j. As we will see, bidder claims converge on the lexicographic max-min solution to the allocation problem; the claim of bidder i is s_i.

The correctness of Algorithm 4.1 and Algorithm 4.2 is established in two steps: Lemma 4.1 establishes forward progress on the number of auctioneers to have converged on their final offer. Theorem 4.2 employs Lemma 4.1 to show eventual convergence to the lexicographic max-min allocation. Let $claim_i$ denote the claim of bidder i and $offer_j$ the offer of auctioneer j. Assume that the resource allocation remains constant for the period of analysis, that bidder i knows R_i and w_i, and that

Algorithm 4.2 Auctioneer for Resource j.

```
1: upon initialization
2:     D_j ← ∅
3:     c_j ← 0
4:     UpdateOffer()
5: end upon
6: upon receiving a new capacity of c_j
7:     UpdateOffer()
8: end upon
9: upon receiving claim from bidder i
10:    claims[i] ← claim
11:    UpdateOffer()
12: end upon
13: upon bidder i joining auction j
14:    D_j ← D_j ∪ i
15:    UpdateOffer()
16: end upon
17: upon bidder i leaving auction j
18:    D_j ← D_j \ i
19:    UpdateOffer()
20: end upon
21: procedure UpdateOffer()
22:    D*_j ← ∅
23:    A_j ← c_j
24:    done ← False
25:    while ( done = False ) do
26:        if ( D*_j = D_j ) then
27:            done ← True
28:            offer ← A_j + max({claims[i] : i ∈ D_j})
29:        else
30:            done ← True
31:            offer ← A_j/|D_j \ D*_j|
32:            for all b ∈ {D_j \ D*_j} do
33:                if ( claims[b] < offer ) then
34:                    D*_j ← D*_j ∪ b
35:                    A_j ← A_j − claims[b]
36:                    done ← False
37:    send offer to all bidders in D_j
38: end procedure
```

auctioneer j knows D_j and c_j. Further assume communication between adjacent auctioneers and bidders is not delayed indefinitely. A claim or offer is *stable* if it has converged on its final value. Denote by A_{stable} the set of auctioneers whose offers (1) are stable and (2) remain the smallest among all offers.

Lemma 4.1. *Suppose A_{stable} contains k auctioneers, $0 \leq k < N$. Then, within finite time, at least one auctioneer converges on the next smallest offer o_{min}. Offers equal to o_{min} are stable and remain smaller than all other offers not in A_{stable}.*

Proof. Wait sufficient time for every bidder i to send a new claim to auctioneers in R_i and for every auctioneer j to send a new offer to bidders in D_j. Let $o_{\min}$ be the smallest offer of an auctioneer not in A_{stable}. Assume to the contrary that offer_x for some auctioneer $x \notin A_{\text{stable}}$ is the *first* to become smaller than $o_{\min}$. By Equation 4.2 and Equation 4.4, a decrease to offer_x can only occur *after* a bidder y at auction x with $\mathit{claim}_y < \mathit{offer}_x$ increases its claim. By Equation 4.1, claim_y can increase only *after* its limiting constraint starts out smaller than offer_x and increases. Constraints in the system smaller than offer_x are maximum claims, offers from A_{stable}, and offers equal to $o_{\min}$. Maximum claims and offers from A_{stable} do not change, leaving some auctioneer x' with $\mathit{offer}_{x'} = o_{\min}$ as the only potential limiting constraint for claim_y. By Equation 4.2 and Equation 4.4, $\mathit{offer}_{x'}$ can increase only *after* one of its bidders y' reduces its claim to be smaller than $\mathit{offer}_{x'}$. By Equation 4.1, $\mathit{claim}_{y'}$ can get smaller only *after* one of its auctioneers, say x'', reduces its offer to be $\mathit{offer}_{x''} < o_{\min} = \mathit{offer}_{x'}$ contradicting the assumption that offer_x is the *first* to get smaller than $o_{\min}$. Therefore, offers equal to $o_{\min}$ remain smaller than offers not from A_{stable}.

By Equation 4.2 and Equation 4.4, any auctioneer j offering $o_{\min}$ can change only after a bidder i at auction j with $\mathit{claim}_i \leq o_{\min}$ changes. By Equation 4.1, claim_i only changes if its limiting constraint changes. Potential limiting constraints include w_i, offers from A_{stable}, and offers equal to $o_{\min}$. These constraints are stable; therefore, offers equal to $o_{\min}$ are stable. □

Theorem 4.2. *Bidders and auctioneers of Algorithm 4.1 and Algorithm 4.2 compute the lexicographic max-min allocation.*

Proof. We apply Lemma 4.1 to show by induction that every auctioneer eventually computes a stable offer.

Base Case: Consider an allocation problem with arbitrary w_i, c_j, R_i, and D_j for $1 \leq i \leq M$, $1 \leq j \leq N$. Let $|A_{\text{stable}}| = 0$. By Lemma 4.1, at least one auctioneer eventually converges on a smallest offer $o_{\min}$. Offers equal to $o_{\min}$ are stable and remain smallest among all offers. Add auctioneers offering $o_{\min}$ to A_{stable} so that $|A_{\text{stable}}| \geq 1$.

Inductive Step: Let $|A_{\text{stable}}| = k$, $1 \leq k < N$. Then, by Lemma 4.1 a non-empty set of auctioneers A^+, with $A^+ \cap A_{\text{stable}} = \emptyset$, eventually converge on the next smallest offer. Offers from A^+ remain smaller than offers not from A^+ or A_{stable} and are stable. Add auctioneers in A^+ to A_{stable} so that $|A_{\text{stable}}| \geq k + 1$.

By the principle of mathematical induction, all auctioneers are eventually added to A_{stable}. Wait for auctioneers to send their offers to adjacent bidders. Bidder claims are now stable. By Equation 4.1, bidder i is either satisfied with its claim ($\mathit{claim}_i = w_i$) or its claim is maximal at an

auction in R_i. By Definition 2.2 on page 14, the claims are lexicographic max-min. □

4.2 Distributed Computation of TLA Persistences in ATLAS

The problem of channel allocation in wireless networks can be expressed in terms of the resource allocation problem in Section 2.2.1 on page 13. Recall that transmitters correspond to demands in D and receivers to resources in R. Receiver j is in R_i if it is within transmission range of transmitter i. D_j contains the transmitters for which receiver j is within transmission range. For traffic load, w_i at bidder i is set to the percentage of slots required to support the demand at transmitter i. Receiver capacities are set to one targeting 100% channel allocation. The lexicographic max-min solution $\boldsymbol{s} = (s_1, \ldots, s_N)$ for a given topology and traffic load is the *TLA allocation*.

To apply the distributed algorithm to channel allocation, we integrate it into ATLAS, a simple random-scheduled MAC protocol. In ATLAS, each node runs a bidder (Algorithm 4.1) and an auctioneer (Algorithm 4.2) continuously. Nodes notify their auctioneers and bidders of topology changes, *i.e.*, new and lost neighbours, maintaining sets R_i and D_j. Likewise, each node updates its bidder's demand magnitude to accurately reflect its traffic load. Offers and claims are embedded within the MAC header of all transmissions to be piggybacked on existing network traffic. A node's offer and claim are received by all single-hop neighbours reaching the bidders and auctioneers that need to know the offer and claim. In time, the bidder claims converge on the TLA allocation $\boldsymbol{s}$.

The TLA allocation could be interpreted directly as a set of persistences in a p-persistent MAC. However, we achieve lower variation in delay by introducing the notion of a frame [29]. Specifically, ATLAS divides time into slots which are organized

into frames of v slots. Node i operates at persistence $p_i = s_i$. At the start of every frame and upon any change to p_i, node i computes $k_i = \lfloor p_i v \rfloor + 1$ with probability π_i and $k_i = \lfloor p_i v \rfloor$ with probability $1 - \pi_i$ where $\pi_i = p_i v - \lfloor p_i v \rfloor$. Node i constructs a transmission schedule of k_i slots selected uniformly at random from the v slots in the frame. Over many frames, $E[k_i]/v$ approaches p_i where $E[k_i]$ is the expectation for k_i. Each node's selection of persistence p_i is informed by the state of its bidder.

There are many implementation choices to be made in applying the distributed algorithm to channel allocation. We identify three to be evaluated further in simulation.

4.2.1 Lazy or Eager Persistences

At the heart of ATLAS is its use of the TLA allocation computed by the distributed algorithm. Here, we identify a lazy approach and an eager approach to deriving transmitter persistences from the state of the algorithm.

A lazy approach sets persistence p_i equal to the *claim* of bidder i. Once converged, p_i matches the TLA allocation interpreted as a persistence. There is a potential disadvantage with being lazy. For many applications, nodes cannot predict future demand for the channel; they can only estimate demand based on past events, *i.e.*, packet arrival rate or queue depth. As a consequence, w_i lags the true magnitude of the demand at node i. If w_i is the limiting constraint for the *claim* of bidder i, the lag affects both the *claim* and persistence p_i. The end result is a sluggish response to increases in demand with potential for degraded performance as is seen in the TCP throughput results of Section 4.4.6.

Alternatively, an eager approach sets transmitter persistences according to the offers of adjacent auctioneers rather than the claim of its bidder, breaking direct dependence on estimation of w_i. Each bidder computes $p_i = \min(\mathit{offers}[j] : j \in R_i)$.

Under stable conditions a node's channel *occupancy*, the fraction of time it spends transmitting, matches its TLA allocation; its occupancy is limited by the availability of packets to transmit which is no larger than w_i, even when $p_i > w_i$. The motivation for selecting p_i according to auctioneer offers becomes apparent when demands are dynamic. Consider the case where the demand at node i starts out at zero and increases abruptly. If p_i is set equal to the *claim* at bidder i, p_i remains equal to zero until w_i catches up with the demand increase. If, on the other hand, p_i is set equal to the smallest offer, it is already non-zero—even before the demand increase is detected—resulting in faster response to demand increases. With improved responsiveness comes the risk of short term channel over-allocation. A sudden increase to a node's demand can result in an occupancy greater than what is reported to adjacent auctions through the bidder's *claim*. Such over-allocation is not resolved until the demand estimate w_i catches up with the true demand and the bidder updates its *claim* to reflect the increased channel occupancy.

4.2.2 Physical Layer or MAC Layer Receivers

A central objective of the TLA allocation is to ensure no receiver is overrun. In a wireless network, receivers can be defined in terms of physical layer or MAC layer communication. At the physical layer, every node with an active neighbour is a receiver. At the MAC layer, packets are filtered by destination address; a node is only a receiver if one of its neighbours has MAC packets destined to it.

If the TLA allocation is defined in terms of MAC layer receivers, non-receiving nodes can disable their auctions. Disabled auctions no longer constrain bidder claims, allowing for a potentially higher channel allocation, but also slow detection of MAC receivers. A node identifies itself as a receiver upon receiving a packet, but its ability

to do so is contingent on not too much contention. It only takes the reception of a single MAC packet to re-enable the auction.

4.2.3 Weighted or Non-Weighted Bidders

So far, we have described a MAC protocol where transmitters are represented by equally weighted bidders. For applications requiring multiple demands per transmitter, we propose the *weighted* TLA allocation which allows demands comprised of one or more *demand fragments*; the number of fragments accumulated into a demand is the demand's *weight*. Let γ_i denote the weight for demand i. Demand fragments in demand i have magnitude w_i/γ_i. The weighted TLA allocation defines the lexicographically max-min vector $u = (u_1, \ldots, u_N)$ where u_i is the allocation to *each* demand fragment in demand i for a total allocation of $u_i\gamma_i$ to demand i.

The distributed algorithm can be extended to compute the weighted TLA allocation. Each bidder transmits its weight γ_i alongside its *claim*. Auctioneers record the weights of adjacent bidders in a *weights*[] array; *weights*[i] holds the weight for bidder i. Bidder i sets its claim to be no larger than the offer of any adjacent auctioneer and no larger than w_i/γ_i. Equation 4.1 becomes

$$claim = \min\left(\{offers[j] : j \in R_i\}, w_i/\gamma_i\right).$$

The auctioneer's calculation of $\mathcal{A}_j$ is modified to subtract the weighted claims for all bidders $i \in D_j^*$ from c_j. Equation 4.3 becomes

$$\mathcal{A}_j = c_j - \left(\sum_{i \in D_j^*} claims[i] \cdot weights[i]\right).$$

Table 4.1: ATLAS configurations selected for simulation.

Configuration Name	**Eager (0) or Lazy (1) Persistences**	**MAC (0) or Physical (1) Receivers**	**Unweighted (0) or Weighted (1) Bidders**
Nominal	0	0	0
Lazy Persistences	1	0	0
Physical Receivers	0	1	0
Weighted Bidders	0	0	1

Finally, the available channel is offered in equal portions to demand fragments at bidders in $D_j \setminus D_j^*$. Equation 4.4 becomes

$$\mathit{offer} = \begin{cases} \mathcal{A}_j / \left(\sum_{i \in D_j \setminus D_j^*} \mathit{weights}[i] \right), & \text{if } D_j \neq D_j^*, \\ \mathcal{A}_j + \max \left(\mathit{claims}[i] : i \in D_j \right), & \text{otherwise.} \end{cases}$$

4.3 Simulation Set-up

We now describe details of the simulation used to produce the experimental results for ATLAS presented in Section 4.4. The configurations are chosen to evaluate the three implementation options outlined in Section 4.2: eager or lazy persistences, MAC or physical layer receivers, and unweighted or weighted bidders. Table 4.1 lists the four ATLAS configurations simulated. The *Nominal* configuration employs eager persistences, defines receivers in terms of MAC layer communication, and operates with unweighted bidders. The other three configurations differ from the nominal case by a single choice and are named accordingly: *Lazy Persistences*, *Physical Receivers*, and *Weighted Bidders*.

4.3.1 Minimum Persistence $p_{\min}$

A node can maintain a persistence of zero without impacting the communication requirements of its bidder. For auctioneers, a persistence of zero is problematic. If a receiver becomes overwhelmed by neighbouring transmitters, a non-zero persistence is needed to quiet the neighbours. To accomplish this, the node enforces a minimum persistence $p_{\min}$, creating dummy packets if necessary, whenever the sum of claims from adjacent bidders exceeds the auction capacity.

4.3.2 Overriding the TLA Allocation with p_{default}

There are two situations where a node constrains its persistence to be no larger than p_{default}. The first is when the node has no neighbours. The TLA allocation permits an isolated node to consume 100% of the channel. However, the primary task of an isolated node is to discover new neighbours, a task made impossible if it is always transmitting. By constraining the persistence to be no larger than p_{default}, the isolated node has a chance to discover new neighbours.

The second time a node employs p_{default} is after the discovery of a new neighbour. It is possible for two or more nodes operating with large persistences to join a neighbourhood at about the same time. If the persistences are large enough, the channel can be overwhelmed, preventing the nodes from discovering each other. By limiting the persistence for a short time following the discovery of a neighbour, the number of transmissions on the channel is reduced, improving odds of successful neighbour discovery.

4.3.3 Adaptation to Topology Changes and t_{lostNbr}

Changes in network topology are detected externally to the distributed algorithm. In ATLAS, neighbour discovery is performed independently by each node. If a node hears from a new neighbour, then the node notifies its bidder of the new auction and its auctioneer of the new bidder. Conversely, if a node has not heard from a neighbour in more than t_{lostNbr} seconds, it presumes the node is no longer a neighbour and informs its auctioneer and bidder accordingly. A symmetric hearing matrix is implicitly assumed.

4.3.4 Scenario Details

Unless otherwise noted, all four configurations are run with $p_{\text{default}} = 0.05$, $t_{\text{lostNbr}} = 0.5$ seconds, and $p_{\min} = 0.01$. The selection of p_{default} and t_{lostNbr} are justified later using Figure 4.3, Figure 4.9a, and Figure 4.9b. The selection of $p_{\min}$ is based on the evaluation found in Section 3.4. Frames contain $v = 100$ slots of length 800 microseconds (1100 bytes per slot).

Simulations are run using the `ns-2` simulator [69]. Each wireless node is equipped with a single unicast transceiver and omni-directional antenna whose physical properties match those of the 914 MHz Lucent WaveLAN DSS radios. The data rate for all simulations is 11 megabits/second. The transmission and carrier sense ranges are 250 meters.

Each simulation runs a network scenario composed of a randomly generated topology and a randomly generated traffic load. Unless specified otherwise, topologies contain 50 randomly placed nodes constrained to a 300 meter by 1500 meter area. With the exception of the multi-hop TCP flows in Section 4.4.6, traffic consists of single-hop constant rate traffic. Four traffic loads are simulated:

- 20% of nodes loaded with small demands (75 ± 50 packets/second),
- 80% of nodes loaded with small demands,
- 20% of nodes loaded with large demands (500 ± 50 packets/second), and
- 80% of nodes loaded with large demands.

For networks containing 50 nodes, these are called 10 Small Demands, 40 Small Demands, 10 Large Demands, and 40 Large Demands, respectively. Nodes to be loaded with traffic are selected at random and the demand magnitudes are selected uniformly at random from the specified range. The packet destination is dynamically selected from the set of neighbouring nodes as the packet is passed down to the MAC layer. For the Weighted Bidders configuration, each demand is assigned a random integer weight between one and five. Combined with the random placement of nodes and the addition of mobility, these four traffic loads enable simulation of a wide variety of network conditions.

Traffic is generated by `ns-2` Constant Bit Rate (CBR) generators; UDP provides transport layer services; packets are 900 bytes in length, leaving room in each slot for header bytes and a MAC layer acknowledgement. Unacknowledged MAC packets are retransmitted up to ten times before they are dropped by the sender.

4.3.5 Offer, Claim, and Weight Encoding

Offers and claims are encoded using eight bits to support a total of 256 unique persistences uniformly distributed between 0 and 1; the error in the representation does not exceed 0.004. Sixteen unique weights (requiring a four-bit representation) may be sufficient for many applications. Adding an offer, claim, and weight to the data packet and acknowledgement of each transmission brings the total communication

overhead to five bytes per packet. For the slot size and data rate simulated in Section 4.4, the communication overhead is 0.46%.

4.3.6 Relative Error

A metric of interest is the average relative error for a node's persistence with respect to the TLA allocation. Error is reported in two parts: relative excess persistence error and relative deficit persistence error. Errors are measured per node over 0.08 seconds consecutive intervals in time (equal to the length of one MAC frame). Let $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_n)$ be a vector of n persistence measurements; $\boldsymbol{\rho}$ can contain measurements for one or more nodes. Each measurement ρ_i is associated with a node and time interval during which the TLA allocation defines a target persistence τ_i. The excess and deficit errors (ϵ_i^+ and ϵ_i^-, respectively) for persistence measurement ρ_i are

$$\epsilon_i^+ = \rho_i - \tau_i \text{ and } \epsilon_i^- = 0$$

if $\rho_i > \tau_i$, and

$$\epsilon_i^+ = 0 \text{ and } \epsilon_i^- = \tau_i - \rho_i$$

otherwise. The relative errors are

$$\eta_i^+ = \epsilon_i^+ / \tau_i$$

and

$$\eta_i^- = \epsilon_i^- / \tau_i.$$

Our intent is to compute the average relative excess error and average relative deficit error for a given sample set of persistence measurements. The relative errors are ratios, requiring use of the geometric rather than arithmetic mean. But, the errors η_i^+ and η_i^- are often zero, preventing direct use of their mean.[1] Instead, we convert errors into

[1] The geometric mean of any data set containing zero is zero.

accuracies eliminating zeros from the data set for a more meaningful geometric average. The average relative accuracies are converted back to relative errors. Specifically, the expectation for excess relative persistence error is

$$E[\eta_i^+] = \left(\prod_{1 \le i \le n} \eta_i + 1 \right)^{1/n} - 1$$

and the expectation for relative deficit persistence error is

$$E[\eta_i^-] = 1 - \left(\prod_{1 \le i \le n} 1 - \eta_i \right).^{1/n}$$

4.4 Evaluation of ATLAS

The results in this section answer four questions concerning the ATLAS MAC protocol: How quickly does the protocol converge on the TLA allocation? Does it scale to larger networks? Can it keep up with changes in a mobile network? And, can it adapt to the needs of multi-hop traffic flows? Results of IEEE 802.11 are included as a common, well known point of reference to assist interpretation. In ATLAS, the upper network layers inform the MAC layer, providing traffic demands to the distributed algorithm. However, it is also possible for ATLAS to inform the decisions made at the upper layers; see Section 4.5.

4.4.1 Convergence after Network Initialization

Figure 4.1 reports average convergence times for all four configurations of ATLAS. Error bars denote the arithmetic standard deviation from the mean for each sample set. Convergence is measured from network initialization (time = 0) to the time the distributed algorithm converges on the TLA allocation. Times are collected from simulations of 1000 network scenarios simulated four times each, once per configuration. Each scenario consists of a randomly generated topology and traffic load; there are 250 scenarios for each traffic load.

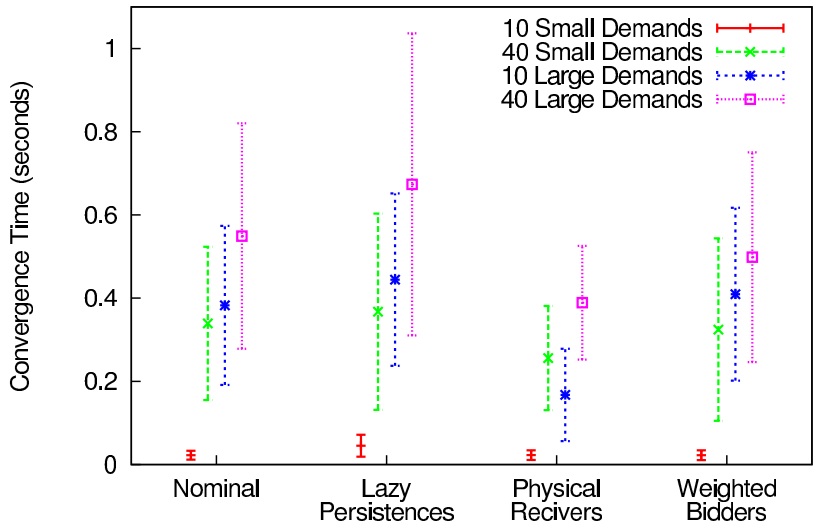
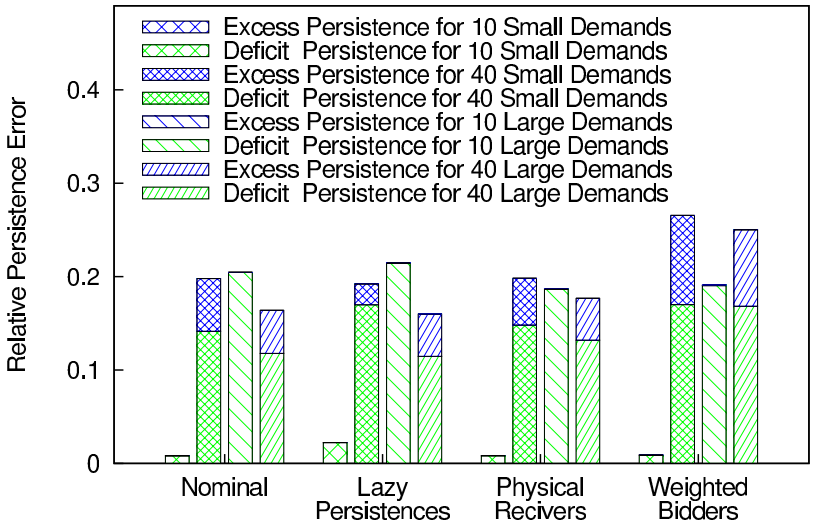


Figure 4.1: Convergence time following network initialization.

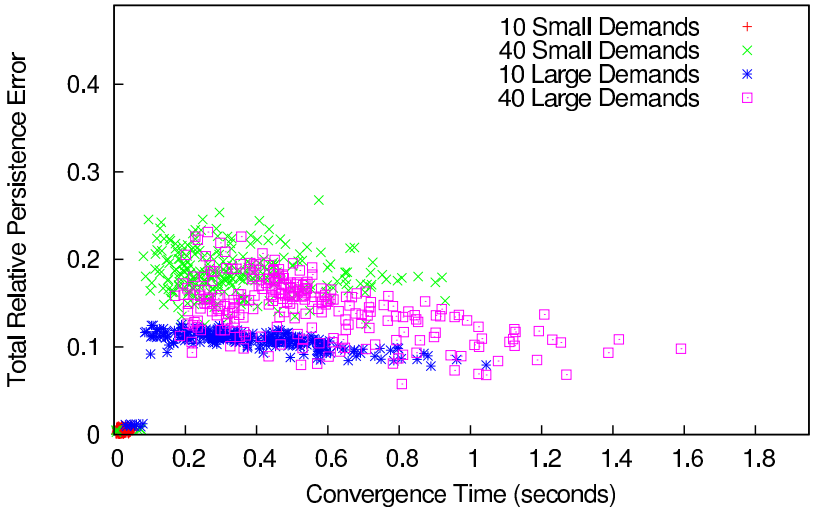
The Physical Receivers configuration converges fastest, in less than 0.4 seconds on average for networks with 40 large demands and faster for other traffic loads. It is the only configuration to enable all auctions unconditionally. The extra step of detecting MAC receivers slows convergence. The Lazy Persistences configuration is the slowest with an average convergence time of 0.67 seconds for networks with 40 large demands. The strict limit on persistences enforced by this configuration compared to the other configuration slows convergence.

Figure 4.2a shows average excess and deficit relative persistence errors for all four configurations. The averages are computed over measurements taken at nodes with a non-zero TLA allocation and only during convergence. Nodes are observed to operate within approximately 20% of their TLA allocation regardless of configuration. Deficit errors are larger than excess errors reflecting a tendency to converge on the TLA allocation from below.

Figure 4.2b plots total relative persistence error—deficit plus excess persistence error—against convergence time for the Nominal configuration. Each data point



(a) Relative excess and deficit persistence errors.



(b) Convergence time vs. error for the Nominal configuration.

Figure 4.2: Relative persistence error for ATLAS.

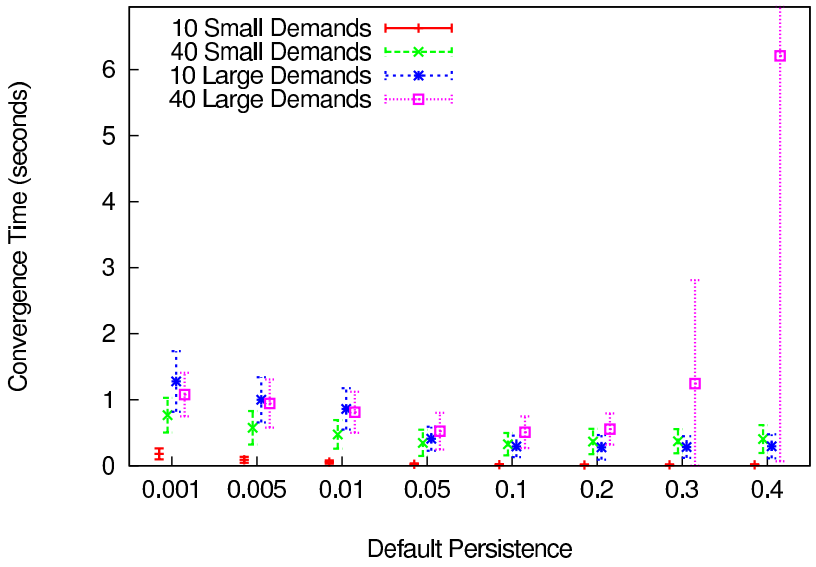


Figure 4.3: Convergence times when run with varying default persistences.

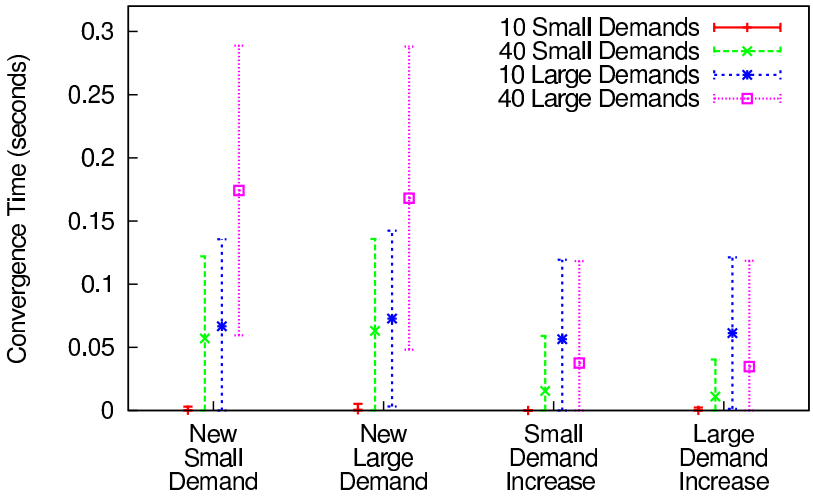
reflects the convergence time (x-coordinate) and total relative persistence error (y-coordinate) for one simulation. The data shows relative persistence error to be fairly consistent from network to network with a maximum observed error of 27%.

Figure 4.3 reports convergence time for the Nominal configuration with different default persistences. Convergence is measured for simulations of 1000 randomly generated network scenarios, 250 of each traffic load. The scenarios are simulated eight times each, once per default persistence: 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, and 0.4. Small default persistences ($p_{\text{default}} \leq 0.01$) limit a node's ability to communicate during neighbour discovery, slowing convergence. Large default persistences ($p_{\text{default}} \geq 0.3$) permit nodes to transmit with large persistences before they discover their neighbours. In networks with 40 of 50 nodes loaded with large demands, the large persistences can overwhelm the channel preventing neighbour discovery and delaying convergence. The distributed algorithm is robust to the selection of p_{default} with a suitable range of [0.05–0.2]. For the remaining simulations, p_{default} is set to 0.05.

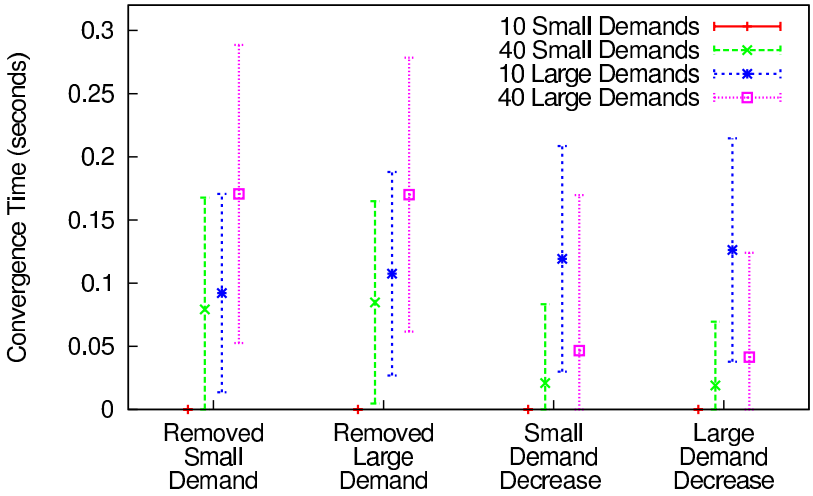
4.4.2 Convergence after a Change in Demand

Figure 4.4 reports convergence times and relative persistence errors for the Nominal configuration following a change to a single demand magnitude. Four types of demand increase are simulated: a new small demand, a new large demand, a small demand increase, and a large demand increase. New demands start with magnitude zero and change to 75 ± 50 packets/second for small demands and to 500 ± 50 packets/second for large demands. Both small and large demand increases start at 75 ± 50 packets/second and change to 150 ± 50 packets/second for small increases and to 500 ± 50 packets/second for large increases. Similarly, four types of demand decrease are simulated. Removed small demands and removed large demands start at 75 ± 50 packets/second and at 500 ± 50 packets/second respectively; both change to zero. Small demand decreases start at 150 ± 50 packets/second and large demand decreases start at 500 ± 50 packets/second; both change to 75 ± 50 packets/second. The eight demand change types are simulated under the four traffic loads. The network is allowed to converge on the initial TLA allocation prior to the demand change. Convergence times and error measurements are taken from simulations of 8000 randomly generated network scenarios, 250 for each of the 32 demand change and traffic load combinations.

Figure 4.4a and Figure 4.4c report convergence times measured from the time of the demand change to the time of convergence on the new TLA allocation. The largest convergence times of approximately 0.175 seconds are found in networks loaded with 40 demands involving new or removed demands. The average convergence time for the other scenarios is 0.125 seconds or smaller. Figure 4.4b and Figure 4.4d show relative persistence errors measured during convergence at nodes whose TLA allocation are

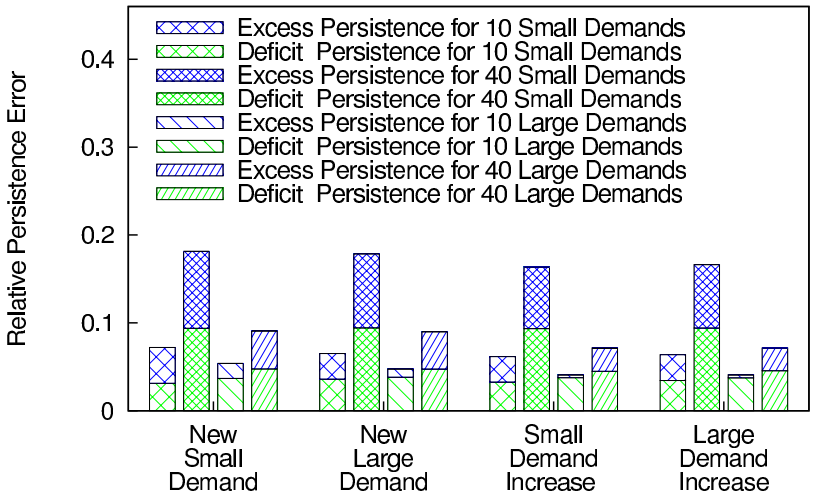


(a) Convergence time after a demand increase.

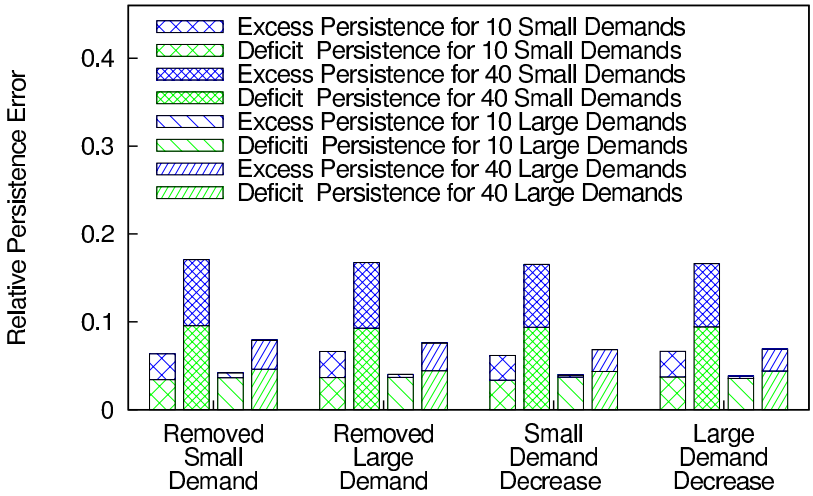


(b) Convergence time after a demand decrease.

Figure 4.4: Convergence time and relative persistence error during convergence following a single demand change.



(c) Relative persistence error after a demand increase.



(d) Relative persistence error after a demand decrease.

Figure 4.4: Continued from page 93.

affected by the demand change. Persistences are observed to be within 10% of the TLA allocation.

4.4.3 Convergence after a Change in Topology

Figure 4.5 reports convergence time and relative persistence error following two types of topology change: the creation of a link and the removal of a link between a pair of nodes. Simulations are run on 2000 randomly generated network scenarios, 250 for each topology change type and traffic load combination. Networks that lose a link are simulated once per neighbour timeout t_{lostNbr} of 0.5, 2.0 and 5.0 seconds.

Network topologies are generated as follows. A first node is placed at a random location in the simulation area. A random point (x, y) is selected inside the simulation area and on the perimeter of the node's transmission range. For topologies gaining a link, a second node is placed near point (x, y) but just *outside* the transmission range of the first node with a trajectory *toward* the first node. For topologies losing a link, the second node is placed near point (x, y) but just *inside* the transmission range of the first node with a trajectory *away* from the first node. The remaining 48 nodes are placed at random locations in the simulation area. The distance travelled by the second node is tightly constrained to avoid unintentional topology changes. In the event that a scenario is generated with more than one topology change, data collected from the scenario is not reported.

The expectation for convergence time following the addition of a new link is 0.025 seconds. The time required to converge after the removal of a link is dominated by the value selected for the neighbour timeout. For $t_{\text{lostNbr}} = 0.5$ seconds, convergence is reached in less than 0.13 seconds on average. During convergence, nodes affected by

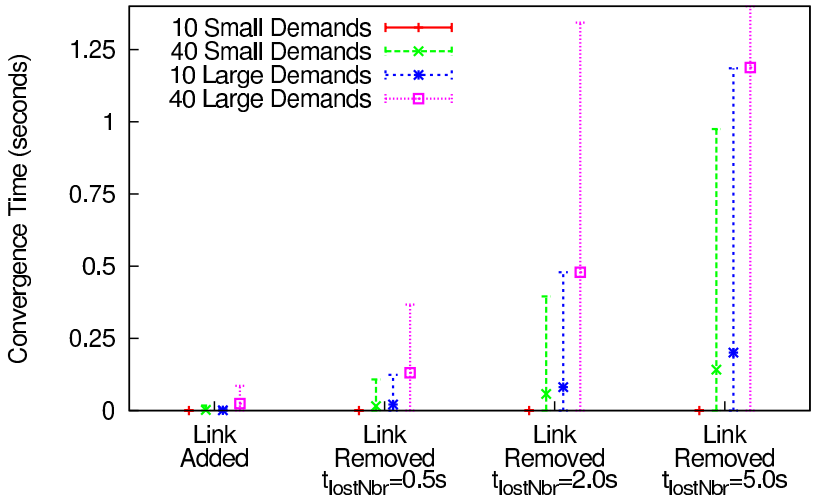
the topology change are observed to operate within 4% of their TLA allocation on average.

These numbers are striking. The small convergence times stem from a counterintuitive feature of the TLA allocation: the majority of topology changes do not affect the TLA allocation. A new link only has an effect if the link connects a bidder with an auction that lacks the capacity to support the bidder's claim. Even in heavily loaded networks, many auctions have spare capacity to support a new bidder. For these scenarios, convergence is instantaneous.

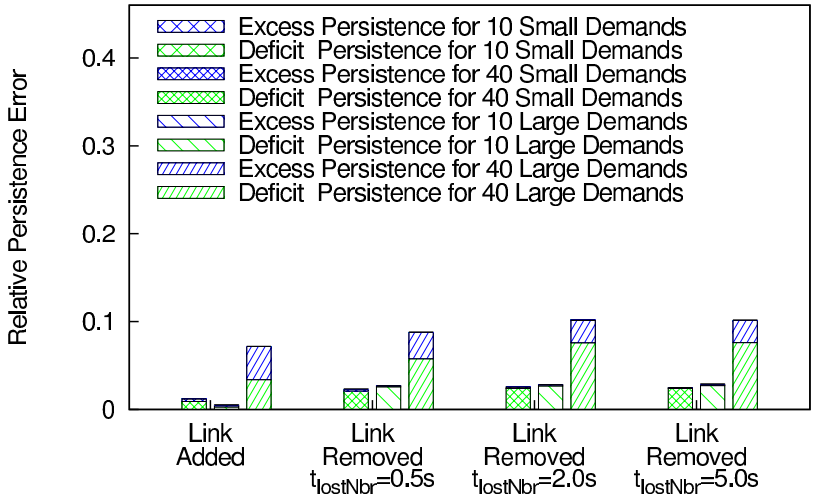
4.4.4 Scalability to Large Networks

We now turn to results demonstrating ATLAS's ability to support large networks. We simulate 10 network sizes with the x-dimension ranging from 600 meters (2.4 hops) to 6000 meters (24 hops) in 600 meter increments; the y-dimension is held constant at 300 meters. The number of nodes is selected to keep the average neighbourhood density constant across all network sizes. Table 4.2 lists the dimensions and number of nodes for each network size. Figure 4.6 reports convergence times for four thousand randomly generated network scenarios, 100 of each traffic load and network size combination. ATLAS's ability to scale to large networks is striking. In networks spanning 24 hops, convergence is reached in an average of 0.89 seconds, a mere 40% increase compared to networks spanning 4.8 hops.

The impressive convergence times, particularly those of networks spanning 12 or more hops, suggest that convergence happens locally, allowing distant neighbourhoods to converge in parallel. The local behaviour is captured in Figure 4.7 which reports the expectation for distance between a network change and a node whose bidder changes its claim during convergence to the new TLA allocation. Distances are reported in



(a) Convergence time after a topology change.



(b) Relative persistence error after a topology change.

Figure 4.5: Convergence time and relative persistence error following a single topology change.

Table 4.2: Topology dimensions and number of nodes for the simulations of Figure 4.6. The y-dimension is fixed at 300 meters.

Number of Nodes	**x-dimension (meters)**	**x-dimension (multiples of tx range)**
20	600	2.4
40	1200	4.8
60	1800	7.2
80	2400	9.6
100	3000	12
120	3600	14.4
140	4200	16.8
160	4800	19.2
180	5400	21.6
200	6000	24

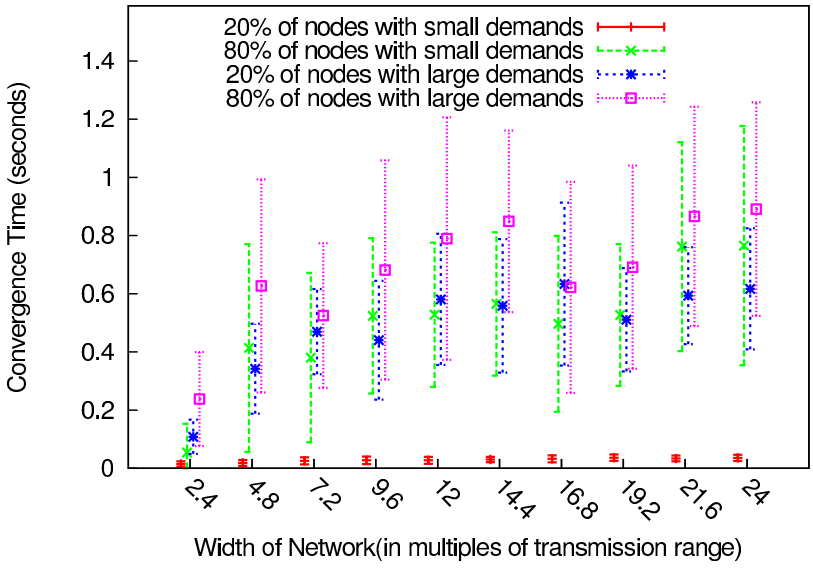


Figure 4.6: Convergence times as the width of the network grows.

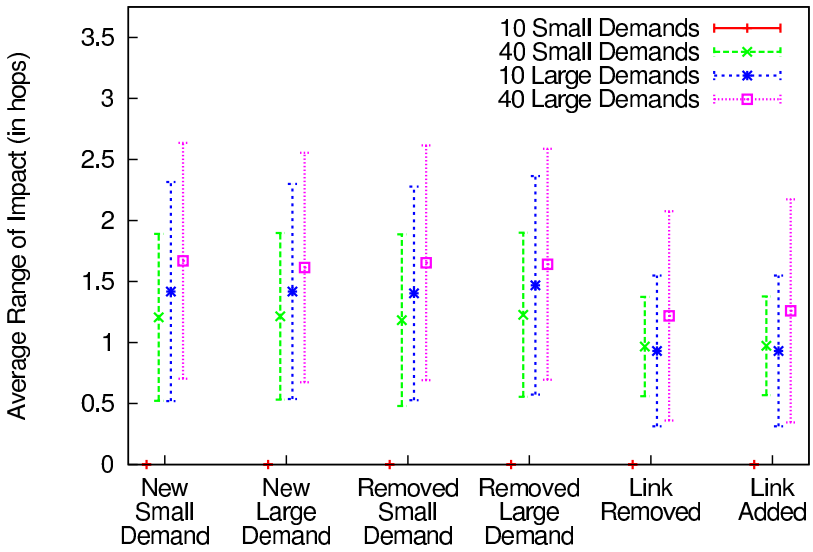


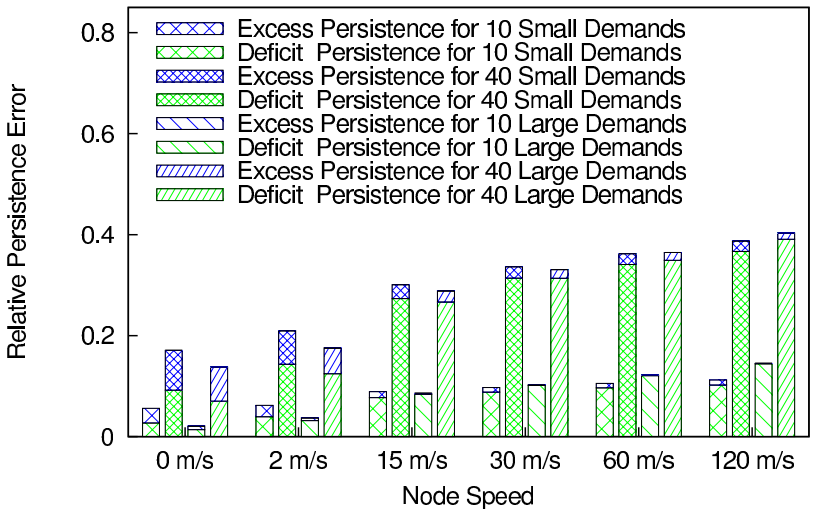
Figure 4.7: Average range of impact (in hops) for a demand or topology change.

hops. A node that changes its demand or gains/loses a neighbour has distance zero. Neighbours of this node have distance one, and so on. Range of impact is reported for six types of change: a new small demand, a new large demand, a removed small demand, a removed large demand, a removed link, and an added link. Each type of change is simulated in 1000 randomly generated network scenarios, 250 of each traffic load. The range of impact is less than 1.75 hops on average.

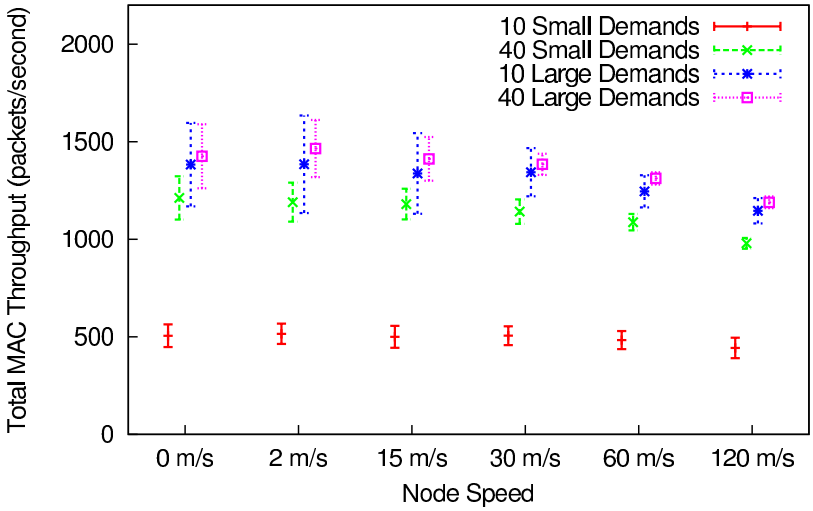
4.4.5 Performance with Node Mobility

Section 4.4.3 addresses the robustness of ATLAS to single topology changes (*i.e.*, new and broken links). We now turn to evaluate its performance in networks with continuous mobility which may not have the opportunity to fully converge on the TLA allocation. It does not make sense to report convergence times; instead, we focus on relative persistence error, total MAC throughput, and packet delay.

Figure 4.8a reports persistence error for node speeds ranging from 0 meters/second to 120 meters/second with 200 random scenarios simulated for each node



(a) Relative persistence error.



(b) Total MAC throughput.

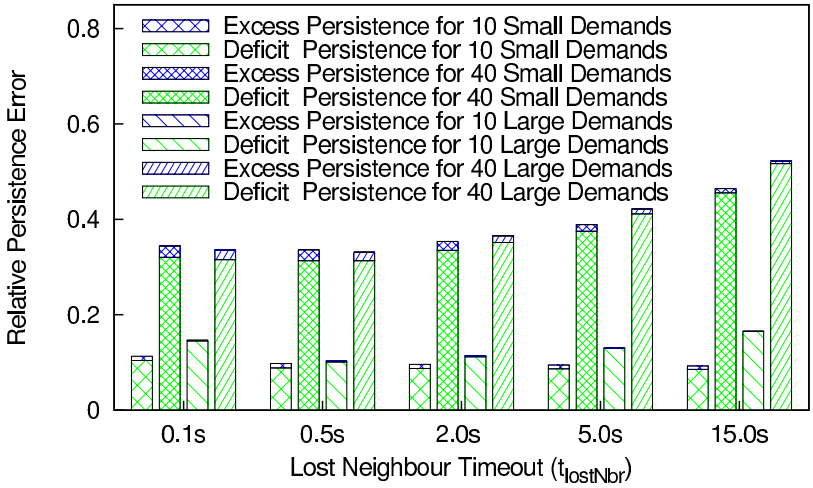
Figure 4.8: Relative persistence error and total MAC throughput for varying levels of node mobility.

speed, 50 of each traffic load. Node movements are generated using the steady-state mobility model generator of [49] with a pause time of zero. Simulations are run for 20 seconds. As node speeds increase, so do deficit persistence errors. The larger deficit errors are an artifact of lost neighbour detection which is delayed by up to $t_{\text{lostNbr}} = 0.5$ seconds for these simulations. As a result, nodes tend to think their neighbourhoods are more crowded than they are, a tendency that gets worse as node speeds increase. In terms of the distributed algorithm, auctioneers and bidders unnecessarily constrain their offers and claims to accommodate lost neighbours.

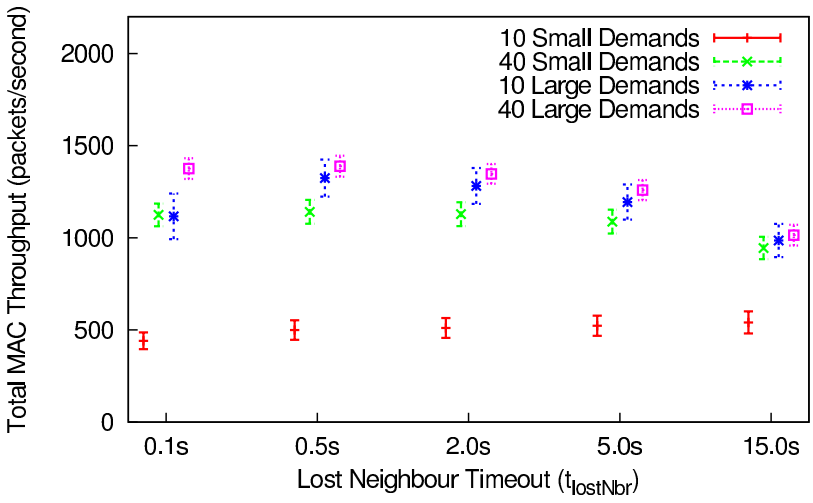
The deficit persistences translate to degraded throughput. Figure 4.8b reports total MAC throughput for the simulations of Figure 4.8a. Even with node speeds of 120 meters/second where a node travels its transmission range in just over two seconds, throughput degrades modestly, decreasing by less than 20% when compared to networks without mobility.

Figure 4.9 shows that a large t_{lostNbr} exacerbates deficit persistence error and further degrades throughput. Data is collected from 200 random scenarios, 50 of each traffic load. Each scenario is simulated five times with neighbour timeouts ranging from 0.1 seconds to 15.0 seconds. Node speeds are fixed at 30 meters/second. Degraded performance is observed for large timeouts, $t_{\text{lostNbr}} \geq 0.5$ seconds, but also for small timeouts, $t_{\text{lostNbr}} = 0.1$ seconds. In networks loaded with 10 large demands, $t_{\text{lostNbr}} = 0.1$ seconds causes nodes to falsely identify lost neighbours that must be rediscovered at the cost of limiting persistences to p_{default}. The remaining simulations are run with $t_{\text{lostNbr}} = 0.5$ seconds.

Figure 4.10 reports packet delay for ATLAS and IEEE 802.11 for the 200 network scenarios of Figure 4.8 with node speeds equal to 30 meters/second. IEEE 802.11 is configured with a maximum packet retry count of seven for RTS, CTS, and



(a) Relative persistence error.



(b) Total MAC throughput.

Figure 4.9: Relative persistence error and total MAC throughput for varying neighbour timeouts.

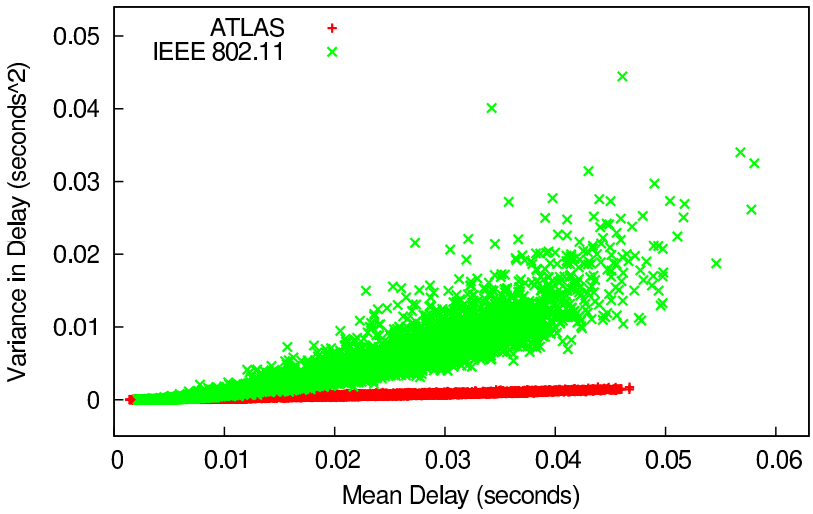


Figure 4.10: Delays for ATLAS and IEEE 802.11.

ACKs and four for data packets [45]. The slot length is set to 20 microseconds; the minimum and maximum contention window sizes are 32 and 1024 slots, respectively. Each point in the scatter plot reports the average packet delay (x-coordinate) and variation in packet delay (y-coordinate) for a single node. The largest reported average delay is 0.047 seconds for ATLAS and 0.058 seconds for IEEE 802.11. The largest reported variation in delay for ATLAS is 0.0016 seconds^2, just 3.6% of the 0.0444 seconds^2 reported for IEEE 802.11.

4.4.6 Multi-hop TCP Flows

To this point, we have used MAC layer traffic to simulate a diverse set of network scenarios. We now turn to evaluate the performance of ATLAS using multi-hop TCP flows. To accommodate the dynamic nature of these flows, each node estimates its own demand by monitoring queue behaviour. Demand is estimated as the sum of two parts: w_{enqueue} and w_{level}. The percentage of channel required to keep up with the current enqueue packet rate is

$$w_{\text{enqueue}} = (\text{packet enqueue rate}) \cdot (\text{slot length})$$

where the packet enqueue rate is estimated as the rate (in packets per second) packets have been added to the queue over the previous 0.1 second of enqueue history. The percentage of channel required to transmit all packets in the queue within 0.2 seconds (*i.e.*, 25 slots) is

$$w_{\text{level}} = [(\ \#\ \text{packets in queue}\ )/0.02\ \text{seconds}\ ] \cdot (\ \text{slot length}\ ).$$

To avoid cross-layer interactions between the MAC and routing protocols, Dijkstra's shortest path algorithm [86] computes the next hop address for all packet transmissions. The algorithm is given accurate knowledge of the global topology. This approach to routing is not intended as a practical solution; it is done to avoid routing issues and to allow our focus to remain on MAC layer behaviour.

We simulate TCP traffic on five MAC protocols: the four configurations of ATLAS and IEEE 802.11. The configurations of ATLAS implement $p_{\text{default}} = 0.05$, $t_{\text{lostNbr}} = 0.5$ seconds, and $p_{\min} = 0.01$. IEEE 802.11 parameters match those described in Section 4.4.5. Each node dynamically sets its bidder weight to one or the number of outgoing TCP flows it services, whichever is larger.

Infinitely sized file transfers are emulated to create flows with throughput limited only by the performance of the network. Transfers start at time zero and run for 20 seconds. Nodes are statically placed at random locations in a 300 meter by 1200 meter simulation area. The source and destination nodes for each FTP file transfer are selected at random. The majority of paths have five or fewer hops.

Each FTP transfer is transported over TCP Reno configured for selective acknowledgements, the extensions of RFC 1323 [6], and 900 byte TCP segments. The return ACKs are not combined with each other or with other data packets. Consequently, the transmission of a single 40-byte TCP ACK consumes an entire

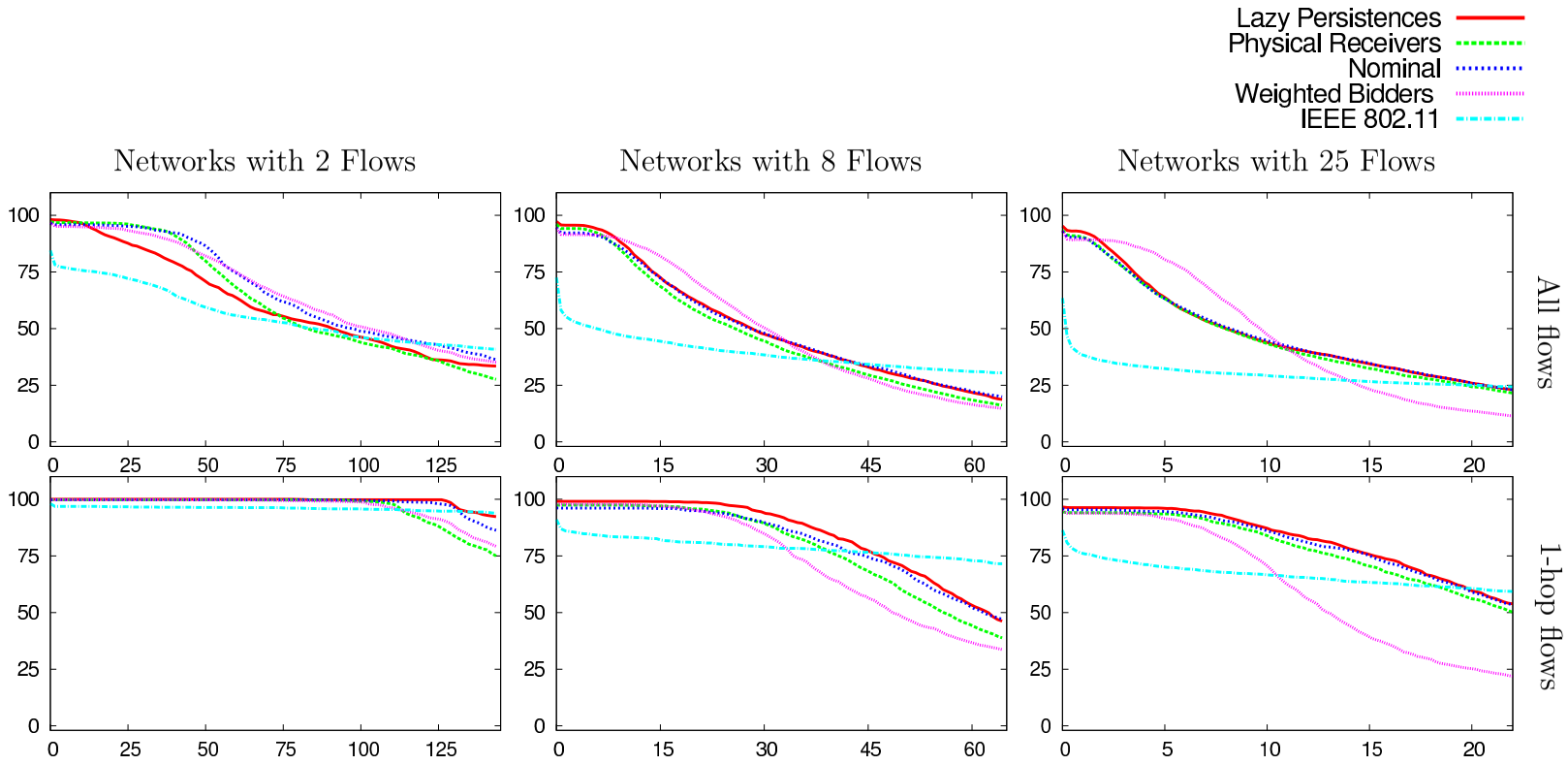


Figure 4.11: Percent of TCP flows achieving a minimum throughput. The y-axis shows the percentage of TCP flows achieving minimum required throughput. The x-axis shows minimum required throughput for TCP flows in packets/second. The figure is continued on page 106.

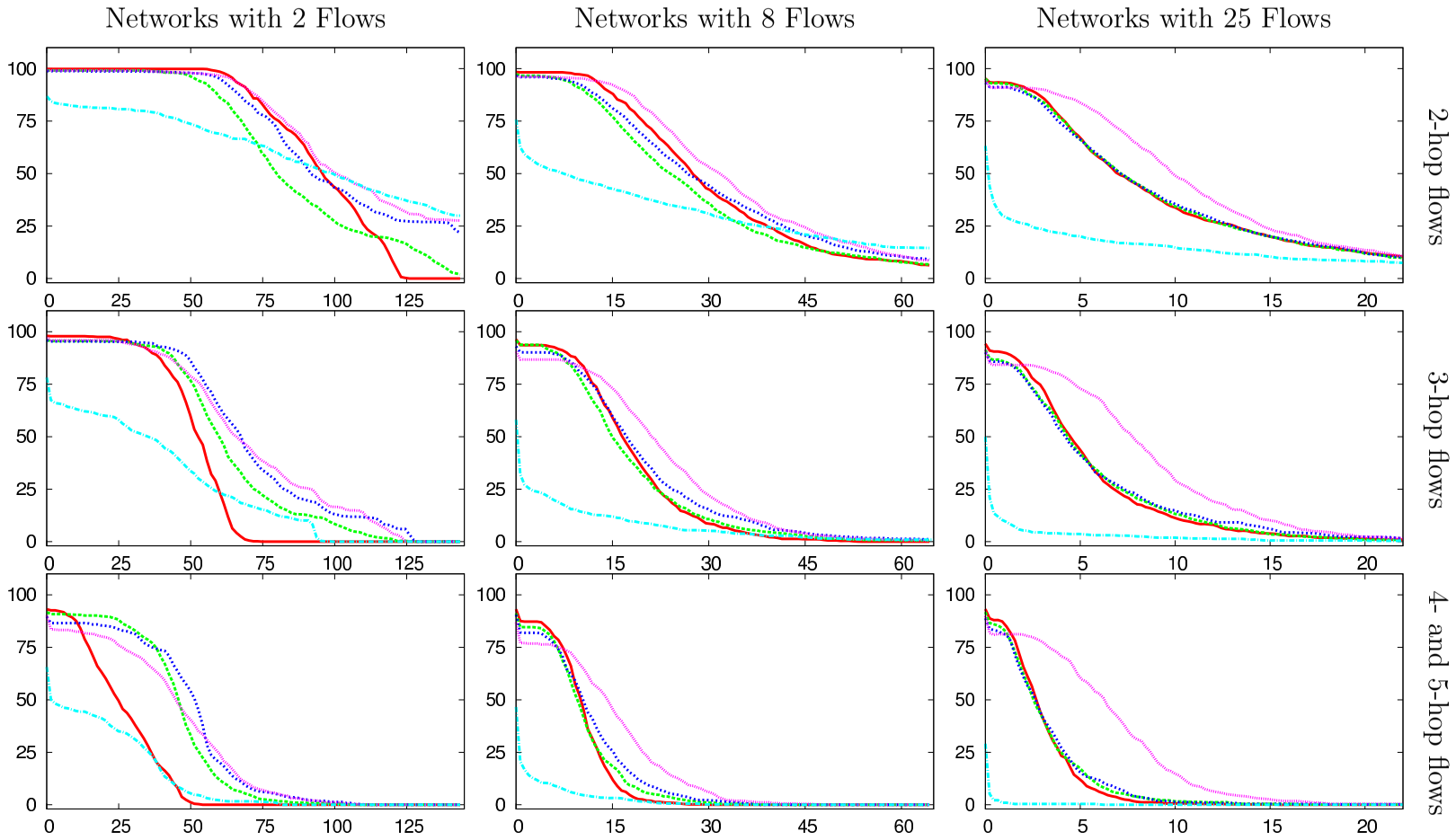


Figure 4.11: Percent of TCP flows achieving a minimum throughput. The figure is continued from page 105.

transmission slot in ATLAS. The maximum congestion window size is four packets to avoid the large delays caused by excessive queueing nodes along multi-hop paths [56, 57].

Network scenarios are simulated for three traffic loads: two, eight, and 25 TCP flows. For each traffic load, the number of replicates simulated is chosen to yield 3000 TCP flows. Consequently, fifteen hundred scenarios are simulated with two TCP flows, 375 with eight TCP flows, and 120 with 25 TCP flows. Each scenario is simulated five times: once with IEEE 802.11 and once per ATLAS configuration.

The 15 sub-plots in Figure 4.11 report on a subset of flows showing the percentage of flows (y-axis) achieving a minimum throughput (x-axis). The plots contain curves for all five MAC protocols. Plots in the left, center, and right columns report on flows from simulations of two, eight, and 25 flows, respectively. The plots in the top row report on flows with five and fewer hops. Plots in the second, third, and fourth rows report on 1-hop, 2-hop, and 3-hop flows, respectively. Plots in the fifth row report on 4- and 5-hop flows.

The distinguishing characteristics of the three unweighted ATLAS configurations are seen in the throughput curves for networks with two flows. These networks are loaded lightly enough for the disabled auctions (at non-receiver nodes) to make a difference in the allocation, improving throughput for 2- and 4-hop flows. Networks with two flows also demonstrate how the longer initial packet delays of the Lazy Persistences configuration increase round trip time for 3-, 4-, and 5-hop flows, preventing TCP from maximizing its contention window and achieving its best throughput.

The Weighted Bidders configuration performs well for multi-hop flows in networks with eight and 25 flows by allocating more to multi-hop flows at the expense of single-hop flows. Because one-hop flows tend to achieve higher throughput, the

configuration maintains a tighter variation in flow throughputs as shown by the steeper slope of the Weighted Bidders curve in the top right plot of Figure 4.11.

Regardless of configuration, ATLAS surpasses IEEE 802.11 in support of concurrent multi-hop flows. The interaction between the IEEE 802.11 back-off algorithm and TCP's congestion control is well known [37]. In testbed experiments, a single TCP flow with no competition has difficulty reaching a destination four hops away [57]. Our simulations corroborate these findings with nearly 50% of the 4- and 5-hop flows reporting a throughput of zero. For networks with 25 demands, greater than 75% of 2-hop flows are non-functional; 3-, 4-, and 5-hop flows are almost completely shut out. The throughput of ATLAS is achieved despite allocating entire slots to 40 byte TCP ACKs. An intelligent packing of TCP ACKs may improve performance further.

4.5 Discussion

In this section we discuss strengths of ATLAS, suggest potential applications of the distributed algorithm, and outline obstacles to the implementation of ATLAS in a real wireless network.

4.5.1 Improved Reliable Transport

TCP is known to perform poorly in wireless networks where signal fading and channel contention are frequent causes of packet loss [58]. Standard TCP assumes loss is due to congestion and responds by reducing its contention window size to limit the number of packets in the network. TCP's congestion control algorithm is further confused by cross-layer interactions with binary exponential back-off (BEB) employed by IEEE 802.11 [37]. BEB tends toward short term unfairness at the MAC layer allowing a single node to capture the channel at the expense of its neighbours [14, 41] causing

high variation in packet delay and making it difficult for TCP to estimate round-trip delay. Multiple extraneous packet timeouts can cause TCP to back-off exponentially retransmitting packets as infrequently as once per second [89].

Many modifications have been proposed to improve performance of TCP over wireless networks [58]; common approaches are detection of packet loss (differentiating it from congestion) and improved estimation of round trip time. An alternative is to minimize packet loss and control variation in packet delay at the MAC layer. ATLAS demonstrates a remarkable ability to control variation in delay (Figure 4.10) enabling TCP to reliably support 3-, 4-, and 5-hop flows over heavily loaded networks (Figure 4.11).

4.5.2 Robust to Selection of Configurable Parameters

ATLAS has three configurable parameters: p_{default}, t_{lostNbr}, and $p_{\min}$. Based on our simulations, [0.01–0.2] is an acceptable range for p_{default} (Figure 4.3) and [0.1–2] seconds is an acceptable range for t_{lostNbr} (Figure 4.9a, Figure 4.9b). Selection of $p_{\min}$ is explored in Chapter 3 for a MAC that enforces a minimum persistence at all nodes and at all times; $p_{\min} \geq 0.1$ is found to provide sufficient communication between auctioneers and bidders. As minimum persistence is increased, MAC throughput declines moderately for dense networks lightly loaded with a few large demands. The throughput decline is less pronounced for ATLAS because $p_{\min}$ is employed temporarily and only at nodes whose auction has become over-allocated.

4.5.3 Dynamic Selection of Auction Capacity

ATLAS targets 100% channel allocation by setting auction capacities to one. Although simulation results show this to be an adequate choice, it is not immediately clear whether performance can be improved by under-allocating or over-allocating the

channel. Indeed, optimal auction capacities (however optimal is defined) are dependent on network topology and quality of the communication channel. We leave a thorough analysis of auction capacity selection to future work, pointing out here that the distributed algorithm adapts continuously to support dynamic selection of the auction capacities.

4.5.4 Potential Applications for the Distributed Algorithm

The weighted TLA allocation opens doors for several potential uses. In the simulations of Section 4.4.6, a bidder's weight is set according to the number of flows serviced by its host node. It may be desirable to set weights according to queue levels, demand magnitudes, neighbourhood sizes, node betweenness [35], distance from a point of interest (*i.e.*, an access point or a common sink), position in a multicast/broadcast tree, or path hop count. The key observation is that the distributed algorithm maintains flexibility by allowing nodes to define bidder weights arbitrarily to suit the needs of the network.

While continuous computation of persistences is our primary motivation, the distributed algorithm is not limited to this. Indeed, the algorithm provides a distributed method for computing the lexicographic max-min solution to any resource allocation problem and is, therefore, applicable to many areas of computer science and engineering. We overview several potential applications in the area of wireless networks.

As one example, consider the Physical Receivers configuration with node demands set to one. The resulting allocation is independent of actions taken by the upper network layers and, therefore, can inform decisions made by those layers. The resulting allocation also serves as a measure of potential network congestion—small allocations are assigned in dense neighbourhoods containing many potentially active

neighbours. The routing protocol can use the allocation to adjust link costs, enabling the discovery of faster, although possibly longer, routes around congestion.

A network can run multiple instances of the distributed algorithm. For example, an instance of the Physical Receivers configuration with all demands set to one can run concurrently with an instance with demands approximating traffic load. The allocation computed by the first instance can inform the upper network layers while the allocation computed by the second instance can inform persistences.

4.5.5 Obstacles to Implementation

A few obstacles remain to the successful deployment of ATLAS in a real-world network. One obstacle is the protocol's assumption that transmission and interference ranges are equal. In real wireless networks, a node's interference range often exceeds it transmission range. In practice, the distributed algorithm's built-in support for dynamic auction capacities can be used to minimize the impact of interference. Each node knows the persistences of its neighbours (from bidder claims and weights) and can compute the expectation for collisions on the channel. Any significant deviation above this expectation can be attributed to interference caused by out-of-range transmitters and can trigger the auction to lower its capacity, making room for the extra transmissions.

Another obstacle is the implicit assumption of a symmetric hearing matrix. This assumption implies that the auction at node A can hear the claims of the bidder at node B if and only if the bidder at node B can hear the offers of the auction at node A. In a real wireless networks, buildings, walls, or localized interference can lead to an asymmetric hearing matrix which can prevent the algorithm from converging on the TLA allocation. In practice, a node's claim and offer can be forwarded on

to neighbouring nodes, effectively providing the information to the node's two-hop neighbourhood. Alternatively, asymmetric links can be treated as interference and auction capacities can be adjusted accordingly.

4.6 Summary

In this chapter we have presented and evaluated ATLAS, a scheduled MAC that implements persistences according to the TLA allocation. ATLAS addresses all four obstacles (listed on page 67) to the deployment of the distributed algorithm of Chapter 3:

1. ATLAS performs its own neighbour discovery.
2. The distributed auction in ATLAS runs asynchronously.
3. ATLAS adapts continuously to changes in the network.
4. ATLAS converges quickly on the TLA allocation, accommodating dynamic topologies and traffic loads.

These improvements make ATLAS a candidate protocol for medium access control in mobile ad hoc networks. Although ATLAS is shown to achieve impressively low variance in delay, it employs random schedules and cannot bound maximum delay. In Chapter 5 we turn to the design of deterministic schedules that can be integrated with ATLAS to achieve a provable bound on delay.

Chapter 5

VARIABLE-WEIGHT AND ADAPTIVE TOPOLOGY SCHEDULING

In this chapter, we introduce the combinatorial requirements for variable-weight topology transparent scheduling and then construct a schedule satisfying those requirements from the blocks of a transversal design. The variable-weight schedules are integrated with ATLAS to create a Variable-Weight and Adaptive Topology Transparent (VWATT) MAC protocol. VWATT enables nodes to select their schedule weights dynamically to accommodate local topology and traffic load without forfeiting the delay guarantee offered by a topology transparent scheme.

The chapter is organized as follows. Section 5.1 develops the concept of variable-weight topology transparent scheduling. A schedule construction is given in Section 5.2. Criterion for selection of schedule weights are given in Section 5.3 and a mechanism for computing weights satisfying the criterion is described in Section 5.4. Simulation results are presented in Section 5.5 and limitations of our construction are discussed in Section 5.6.

5.1 Defining Topology Transparency with Variable-Weight Schedules

Recall that a schedule is a set of transmission slots selected from the slots in a transmission frame. A set of variable-weight schedules contains schedules of different weights. The practical use of variable-weight schedules in a mobile ad hoc network requires that each node be able to select its schedule dynamically, choosing a weight appropriate for the current topology and traffic load. This implies that each node be equipped with multiple schedules, each of a different weight. To maintain homogeneity in the network, the number of schedules and the schedule weights assigned to each node must remain constant across the network. The precise structure of such a schedule

follows. Let $w_1 < \ldots < w_m$ denote m distinct schedule weights and let there be N schedules of each weight for a total of mN schedules. Partition the schedules into N sets, $\mathbb{F}_1, \ldots, \mathbb{F}_N$, with each set containing exactly one schedule of each weight. The set $\mathbb{F}_i = \{F_i^{(1)}, \ldots, F_i^{(m)}\}$ contains the schedules to be used by node i where $F_i^{(\ell)}$ for $1 \leq \ell \leq m$ denotes the schedule with weight $w_\ell = \mathrm{wt}\left(F_i^{(\ell)}\right)$ assigned to node i. $F_i^{(1)}$ is the *base* schedule of node i.

Definition 2.3 on page 14 bounds schedule intersection whenever the number of schedules is not too large. Here, we define topology transparency within the context of variable-weight schedules, bounding schedule intersection when the combined weight, rather than the number, of schedules does not exceed a specified maximum.

Definition 5.1. $\mathbb{F} = \{\mathbb{F}_1, \ldots, \mathbb{F}_N\}$ *is* topology transparent *with parameters* N *and* $W_{\max}$ *if for any* ν *schedule sets* $\mathbb{F}^{(\nu)} = \{\mathbb{F}_{i_1}, \ldots, \mathbb{F}_{i_\nu}\}$*, any schedule set* $\mathbb{F}_{i_0} \not\subset \mathbb{F}^{(\nu)}$*, and weights* $w_{\ell_0}, w_{\ell_1}, \ldots, w_{\ell_\nu} \in \{w_1, \ldots, w_m\}$*,*

$$F_{i_0}^{(\ell_0)} \not\subseteq \bigcup_{j=1}^{\nu} F_{i_j}^{(\ell_j)}$$

whenever

$$\sum_{j=1}^{\nu} \mathrm{wt}\left(F_{i_j}^{(\ell_j)}\right) \leq W_{\max}.$$

Under this definition, a topology transparent schedule ensures that any set of ν nodes $\{i_1, \ldots, i_\nu\}$, provided ν is not too large, *can* select schedule weights so that the union of the ν schedules does not contain the schedule of any node not in $\{i_1, \ldots, i_\nu\}$. Schedules that maintain this property exist whenever $W_{\max}$ is greater than or equal to ν times the minimum schedule weight. It follows that the largest neighbourhood size supported by the schedule is

$$\mathrm{D}_{\max} = \lfloor W_{\max}/w_1 \rfloor + 1.$$

However, when neighbourhood sizes are sufficiently small, nodes can select schedules with larger weights without violating $W_{\max}$.

5.2 Schedules from Transversal Designs

Existing constructions of topology transparent schedules include those derived from orthogonal arrays of strength t, or equivalently, from transversal designs of strength t [16, 53, 83]. We employ the language of transversal designs because it maps readily to cover-free families. In this section we construct a variable-weight topology transparent schedule from a $\mathrm{TD}(t+1, v, v)$, relying on the additional blocks provided by the higher strength and the nested structure.

5.2.1 Constructing a $\mathrm{TD}(t+1, v, v)$

A $\mathrm{TD}(t, v, v)$ can be derived from the set of polynomials of degree less than t with coefficients from $\mathrm{GF}(v)$ [42]. The point set contains the v^2 ordered pairs

$$\boldsymbol{X} = \{(\alpha, \beta) : \alpha, \beta \in \mathrm{GF}(v)\}.$$

The partition $\mathbb{G}$ with groups $\mathbb{G}_0, \ldots, \mathbb{G}_{v-1}$ is formed by grouping points according to the first entry of each pair,

$$\mathbb{G}_\alpha = \{(\alpha, \beta) : \beta \in \mathrm{GF}(v)\}.$$

The blocks of $\mathbb{B}$ are generated from the v^t polynomials of degree less than t over $\mathrm{GF}(v)$. The block generated by polynomial

$$f_{<a_{t-1},\ldots,a_0>}(x) = a_{t-1}x^{t-1} + \ldots + a_0$$

for $a_0, \ldots, a_{t-1} \in \mathrm{GF}(v)$ is

$$B_{<a_{t-1},\ldots,a_0>} = \left\{ \left(b, f_{<a_{t-1},\ldots,a_0>}(b)\right) : b \in \mathrm{GF}(v) \right\}$$

with arithmetic performed over GF(v).

These are not the only polynomials to generate a TD(t, v, v). Any v^t distinct polynomials of degree less than $t+1$ generate a TD(t, v, v) provided they share a common leading coefficient. The polynomials of degree less than t are a special case of this general requirement. There are v possible leading coefficients, each generating a TD(t, v, v) that is isomorphic [42] to the others. Combined, the blocks of the v TD(t, v, v)s represent all the polynomials of degree less than $t+1$, forming a TD($t+1, v, v$).

Table 5.1 shows a TD(3, 3, 3). Each row in the table gives a block in the design: the first and second column name the block and give its generating polynomial; the final three columns contain the points in the block. The blocks are grouped according to the TD(2, 3, 3) to which they belong. The partition $\mathbb{G}$ on $\boldsymbol{X}$ can be seen in columns three, four, and five; each column contains the points (with repetition) of a group in the partition on $\boldsymbol{X}$.

5.2.2 Variable-Weight Schedules from a TD($t+1, v, v$)

Partition the v^{t+1} blocks of a TD($t+1, v, v$) into v^t sets enumerated by the ordered t-tuples over GF(v). The block set numbered $(a_{t-1}, \ldots, a_0)$, for $a_i \in \text{GF}(v)$, is

$$\mathbb{B}_{<a_{t-1},\ldots,a_0>} = \{B_{<b,a_{t-1},\ldots,a_0>} : b \in \text{GF}(v)\}.$$

$\mathbb{B}_{<a_{t-1},\ldots,a_0>}$ contains the blocks generated by the v polynomials with lower-order coefficients $a_{t-1}, \ldots, a_0$. Each of the v^t block sets forms a schedule for a single node, supporting networks of size $N = v^t$. The mapping from block to node is not important provided each node is assigned schedules from a unique block set. Let the schedules

	Block Name	Generating Polynomial	Group $\mathbb{G}_0$	$\mathbb{G}_1$	$\mathbb{G}_2$
TD$(2,3,3)$ from polynomials with $a_2 = 0$	$B_{<0,0,0>}$	0	{(0,0),	(1,0),	(2,0)}
	$B_{<0,0,1>}$	1	{(0,1),	(1,1),	(2,1)}
	$B_{<0,0,2>}$	2	{(0,2),	(1,2),	(2,2)}
	$B_{<0,1,0>}$	x	{(0,0),	(1,1),	(2,2)}
	$B_{<0,1,1>}$	$x+1$	{(0,1),	(1,2),	(2,0)}
	$B_{<0,1,2>}$	$x+2$	{(0,2),	(1,0),	(2,1)}
	$B_{<0,2,0>}$	$2x$	{(0,0),	(1,2),	(2,1)}
	$B_{<0,2,1>}$	$2x+1$	{(0,1),	(1,0),	(2,2)}
	$B_{<0,2,2>}$	$2x+2$	{(0,2),	(1,1),	(2,0)}
TD$(2,3,3)$ from polynomials with $a_2 = 1$	$B_{<1,0,0>}$	x^2	{(0,0),	(1,1),	(2,1)}
	$B_{<1,0,1>}$	x^2+1	{(0,1),	(1,2),	(2,2)}
	$B_{<1,0,2>}$	x^2+2	{(0,2),	(1,0),	(2,0)}
	$B_{<1,1,0>}$	x^2+x	{(0,0),	(1,2),	(2,0)}
	$B_{<1,1,1>}$	x^2+x+1	{(0,1),	(1,0),	(2,1)}
	$B_{<1,1,2>}$	x^2+x+2	{(0,2),	(1,1),	(2,2)}
	$B_{<1,2,0>}$	x^2+2x	{(0,0),	(1,0),	(2,2)}
	$B_{<1,2,1>}$	x^2+2x+1	{(0,1),	(1,1),	(2,0)}
	$B_{<1,2,2>}$	x^2+2x+2	{(0,2),	(1,2),	(2,1)}
TD$(2,3,3)$ from polynomials with $a_2 = 2$	$B_{<2,0,0>}$	$2x^2$	{(0,0),	(1,2),	(2,2)}
	$B_{<2,0,1>}$	$2x^2+1$	{(0,1),	(1,0),	(2,0)}
	$B_{<2,0,2>}$	$2x^2+2$	{(0,2),	(1,1),	(2,1)}
	$B_{<2,1,0>}$	$2x^2+x$	{(0,0),	(1,0),	(2,1)}
	$B_{<2,1,1>}$	$2x^2+x+1$	{(0,1),	(1,1),	(2,2)}
	$B_{<2,1,2>}$	$2x^2+x+2$	{(0,2),	(1,2),	(2,0)}
	$B_{<2,2,0>}$	$2x^2+2x$	{(0,0),	(1,1),	(2,0)}
	$B_{<2,2,1>}$	$2x^2+2x+1$	{(0,1),	(1,2),	(2,1)}
	$B_{<2,2,2>}$	$2x^2+2x+2$	{(0,2),	(1,0),	(2,2)}

Table 5.1: An example TD$(3,3,3)$.

in $\mathbb{F}_i$ be derived from blocks in $\mathbb{B}_{<a_{t-1},\ldots,a_0>}$. Then,

$$F_i^{(\ell)} = \bigcup_{b=0}^{\ell-1} B_{<b,a_{t-1},\ldots,a_0>}$$

for $1 \leq \ell \leq v$. The base schedules $F_i^{(1)}$ for $1 \leq i \leq N$ are formed from the blocks in the TD(t, v, v) generated from polynomials with degree less than t. We refer to this TD(t, v, v) as the *base* transversal design of the schedule. The blocks of $\mathbb{B}_{<a_{t-1},\ldots,a_0>}$ share a single common point of $(0, a_0)$. As a result, $F_i^{(\ell)}$ has weight

$$\mathrm{wt}\left(F_i^{(\ell)}\right) = w_\ell = v + (v-1)(\ell-1) \tag{5.1}$$

with the block from the base transversal design contributing v slots and the $\ell - 1$ other blocks contributing $v - 1$ slots each.

5.2.3 $W_{\max}$ and $\mathrm{D}_{\max}$ for the Variable-Weight Schedule

In order to maintain the collision-free transmission guarantee offered by a topology transparent schedule, any $\nu + 1$ nodes operating with schedule sets $\mathbb{F}_{i_0}, \ldots, \mathbb{F}_{i_\nu}$ must be able to select schedule weights $w_{\ell_0}, w_{\ell_1}, \ldots, w_{\ell_\nu}$ so that

$$F_{i_0}^{(\ell_0)} \not\subseteq \bigcup_{j=1}^{\nu} F_{i_j}^{(\ell_j)}. \tag{5.2}$$

For the variable-weight schedules constructed here, the base schedule of any node is contained within all other schedules assigned to that node. Therefore, Equation 5.2 holds whenever

$$F_{i_0}^{(1)} \not\subseteq \bigcup_{j=1}^{\nu} F_{i_j}^{(\ell_j)}. \tag{5.3}$$

Let

$$\mathcal{I}\left(F_i^{(\ell)}\right) = \max\left(|F_i^{(\ell)} \cap F_{i'}^{(1)}| : i' \neq i\right)$$

be the maximum number of intersections between schedule $F_i^{(\ell)}$ and the base schedule of any other node $F_{i'}^{(1)}$. $F_i^{(\ell)}$ contains a single block that is in the same TD(t, v, v) as

the block defining $F_{i'}^{(1)}$; these blocks are from the base transversal design and they intersect in fewer than t points. The remaining $\ell - 1$ blocks in $F_i^{(\ell)}$ are in the same TD$(t+1, v, v)$ as the block of $F_{i'}^{(1)}$; they intersect in fewer than $t+1$ points. In terms of design strength t and schedule weight ℓ, the maximum intersection is

$$\mathcal{I}\left(F_i^{(\ell)}\right) = (t-1) + t(\ell - 1). \tag{5.4}$$

Schedule $F_{i_0}^{(1)}$ has weight $w_1 = v$ and can accommodate up to $v-1$ intersections while maintaining at least one unique slot. No schedule is permitted to contain the base schedule of another node, so the maximum schedule weight is w_m for

$$m = \left\lfloor \frac{v-t}{t} \right\rfloor + 1.$$

Therefore, the limited schedule intersection identified in Equation 5.3 holds whenever

$$\sum_{j=1}^{\nu} \mathcal{I}\left(F_{i_j}^{(\ell_j)}\right) \leq w_1 - 1 = v - 1. \tag{5.5}$$

The $v-1$ allowable intersections of schedules $F_{i_1}^{(\ell_1)}, \ldots, F_{i_\nu}^{(\ell_\nu)}$ with $F_{i_0}^{(1)}$ can be partitioned into two groups: intersections associated with the ν blocks from the base transversal design—each of these blocks contribute at most $t-1$ intersections—and intersections coming from blocks outside of the base transversal design—each of these blocks contribute at most t intersections. Let h measure the use of higher weight schedules by counting the blocks outside of the base transversal design that are in $F_{i_1}^{(\ell_1)}, \ldots, F_{i_\nu}^{(\ell_\nu)}$. (Recall that base schedules are defined directly by blocks in the base transversal design and that every higher weight schedule is built from one block in the base transversal design and one or more blocks from outside of the base transversal design.) Then Equation 5.5 can be rewritten as

$$\nu(t-1) + ht \leq v - 1. \tag{5.6}$$

There is a trade off between the neighbourhood size (ν) and the use of higher weight schedules (h). When $h = 0$, ν can be its largest: $\nu \leq \frac{v-1}{t-1}$. Therefore, the maximum neighbourhood size supported by the schedule is

$$\mathrm{D}_{\max} = \frac{v-1}{t-1} + 1.$$

Solving Equation 5.6 for h gives a bound on the use of higher weight schedules as a function of ν.

$$h \leq \frac{(v-1) - \nu(t-1)}{t} \tag{5.7}$$

Using Equation 5.7, we can compute the maximum combined weight of the ν schedules that ensures the intersection property of Equation 5.3 holds. The ν blocks from the base transversal design contribute a weight of v each. The h blocks outside of the base transversal design contribute a weight of $v-1$ each. The result is a maximum combined weight of

$$\sum_{j=1}^{\nu} \mathrm{wt}\left(F_{i_j}^{(\ell_j)}\right) \leq \nu v + h(v-1) \leq \nu v + \frac{(v-1) - \nu(t-1)}{t}(v-1) = W_{\max}. \tag{5.8}$$

For this particular construction, $W_{\max}$ is an increasing function on ν and is valid for $\nu \leq \frac{v-1}{t-1}$. Nodes can ignore ν and select schedule weights so that the sum of weights is smaller than the smallest possible $W_{\max}$ (with $\nu = 1$). Or, nodes can account for ν and compute a potentially larger $W_{\max}$. Our computation of schedule weights, described in Section 5.4, does the latter.

5.2.4 An Example

Consider the schedules generated from the TD$(3, 3, 3)$ of Table 5.1. Without loss of generality, suppose that schedules derived from $\mathbb{B}_{<a_1,a_0>}$ are assigned to node $i = a_1 v + a_0$. Then, $\mathbb{B}_{<1,2>}$—containing blocks $B_{<0,1,2>}$, $B_{<1,1,2>}$, and $B_{<2,1,2>}$—forms the schedules $F_5^{(1)}$, $F_5^{(2)}$, and $F_5^{(3)}$. These schedules have the potential to intersect

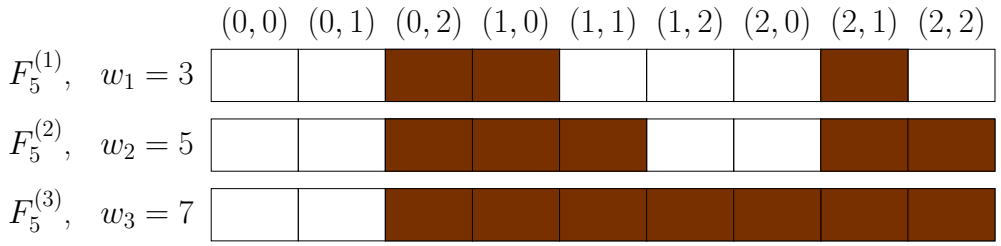


Figure 5.1: $\mathbb{F}_5$ generated from $\mathbb{B}_{<1,2>}$ in the TD$(3, 3, 3)$ of Table 5.1.

the base schedule of another node in 1, 3, and 5 positions, respectively. The three schedules are shown in Figure 5.1. For this small example, the number of usable schedules is $m = \lfloor 1/2 \rfloor + 1$, so $F_5^{(1)}$ is the only usable schedule. For larger v, the construction yields multiple usable schedules.

5.2.5 Existence of Schedules

A TD$(t + 1, v, v)$ exists whenever v is a prime power and $0 \leq t \leq v$. Table 5.2 on page 122 reports properties of the schedules generated from TD$(3, v, v)$s for v equal to prime powers less than or equal to 64. The table includes: frame length, maximum supported neighbourhood size ($\mathrm{D_{max}}$), maximum number of nodes (N), and the number of distinct schedule weights ($m = \lfloor v/2 \rfloor$ when $t = 2$). The table also reports equivalent persistence—the percentage of time a node is permitted to spend transmitting—for the base schedule, the highest weight schedule, and the delta between adjacent schedule weights. The final column reports the guarantee on maximum delay assuming a slot length of 0.0008 seconds (matching the slot length simulated for Section 5.5). In general, as v grows, the flexibility of the schedule (measured in N, $\mathrm{D_{max}}$, and m) improves while the delay guarantee weakens.

Figure 5.2 shows $W_{\max}$ to grow with neighbourhood size. This is because larger neighbourhoods contain more blocks from the base transversal design and fewer blocks from outside of the base transversal design (see Equation 5.7). Because the blocks

v	Frame Length	D_{max}	N	m	Persistence			Maximum Delay (seconds)
					Base	Δ	Max	
2	4	2	4	1	0.500	—	0.500	0.003
3	9	3	9	1	0.333	—	0.333	0.007
4	16	4	16	2	0.250	0.188	0.438	0.013
5	25	5	25	2	0.200	0.160	0.360	0.020
7	49	7	49	3	0.143	0.122	0.388	0.039
8	64	8	64	4	0.125	0.109	0.453	0.052
9	81	9	81	4	0.111	0.099	0.407	0.065
11	121	11	121	5	0.091	0.083	0.421	0.098
13	169	13	169	6	0.077	0.071	0.432	0.136
16	256	16	256	8	0.062	0.059	0.473	0.206
17	289	17	289	8	0.059	0.055	0.446	0.233
19	361	19	361	9	0.053	0.050	0.452	0.291
23	529	23	529	11	0.043	0.042	0.459	0.426
25	625	25	625	12	0.040	0.038	0.462	0.504
27	729	27	729	13	0.037	0.036	0.465	0.587
31	961	31	961	15	0.032	0.031	0.469	0.774
32	1024	32	1024	16	0.031	0.030	0.485	0.825
37	1369	37	1369	18	0.027	0.026	0.474	1.103
41	1681	41	1681	20	0.024	0.024	0.477	1.355
43	1849	43	1849	21	0.023	0.023	0.478	1.490
47	2209	47	2209	23	0.021	0.021	0.479	1.780
49	2401	49	2401	24	0.020	0.020	0.480	1.935
53	2809	53	2809	26	0.019	0.019	0.482	2.264
59	3481	59	3481	29	0.017	0.017	0.483	2.805
61	3721	61	3721	30	0.016	0.016	0.484	2.999
64	4096	64	4096	32	0.016	0.015	0.492	3.301

Table 5.2: Properties of schedules derived from TD$(3, v, v)$s for $v \leq 64$. Delays assume a slot length of 0.0008 seconds.

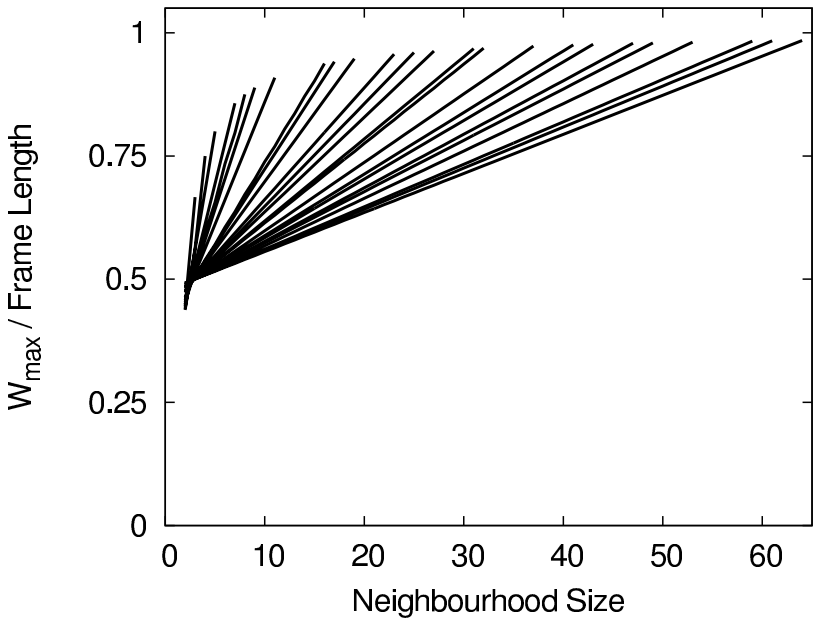


Figure 5.2: The lines define the ratios of $W_{\max}$ to frame length for schedules derived from TD$(3, v, v)$s for $v \leq 64$. The line with the gentlest slope is for $v = 64$, the line with the next gentlest slope is for $v = 61$, and so on.

outside of the base transversal design have a greater potential to intersect other base schedules, their use reduces the total number of blocks that can be employed in a neighbourhood and, therefore, reduces $W_{\max}$ (see Equation 5.8).

5.3 Defining Target Schedule Weights

The variable-weight schedule described in Section 5.1 maintains a guarantee on maximum delay provided the combined weight of the schedules in any neighbourhood is not too large. This raises a critical question: By what criterion are weights selected? We propose the following criterion for the selection of schedule weights, assuming the schedules are topology transparent with parameters N and $W_{\max}$ (per Equation 5.8), the structure defined in Section 5.1, and the construction given in Section 5.2:

1. The sum of schedule weights employed in a neighbourhood should not exceed $W_{\max}$.

2. Nodes should not select schedules with weights larger than they can use.

3. Nodes should not use higher weight schedules at the expense of other nodes using lower weight schedules, *i.e.*, schedule weights should be distributed fairly among the nodes in a neighbourhood.

4. Schedule weights should be maximized subject to the first three criterion.

All four criterion are satisfied by the lexicographic max-min allocation of schedule weights, which we define next. Recall that R contains the set of N receivers and D the set of N transmitters in the network. Receiver $j \in R$ is within range of transmitters in $D_j \subseteq D$ and the transmissions of transmitter i reach the receivers in $R_i \subseteq R$. Transmitters operate with schedule weights selected from $\{w_1, \ldots, w_m\}$. Let d_i and s_i be indices into the vector of schedule weights. Transmitter i desires weight w_{d_i}, is allocated weight w_{s_i}, and employs schedule $F_i^{(s_i)}$. The allocation $\boldsymbol{s} = (s_1, \ldots, s_N)$ is *feasible* if

$$\sum_{i \in D_j} \mathcal{I}\left(F_i^{(s_i)}\right) \leq w_1 - 1$$

for all $j \in R$, and

$$s_i \leq d_i$$

for all $i \in D$. Transmitter $i \in D$ is *satisfied* if $s_i \geq d_i$. A receiver $j \in R$ is *saturated* if the transmitters with maximal weight in D_j cannot simultaneously increase their weight without

$$\sum_{i \in D_j} \mathcal{I}\left(F_i^{(s_i)}\right)$$

exceeding $w_1 - 1$.

Definition 5.2. *[71] A feasible allocation of schedule weights* $\boldsymbol{s}$ *is* lexicographically max-min *if, for every transmitter* $i \in D$*, either transmitter* i *is*

satisfied, or there exists a saturated receiver j where the weight of $i \in D_j$ is maximal among the transmitters in D_j.

A feasible allocation does not exist for neighbourhoods larger than $\mathrm{D}_{\max}$. Rather than stopping transmissions in these neighbourhoods, we allow the nodes unconditional use of their base schedules, regardless of neighbourhood size, and permit use of higher weight schedules as allowed by the allocation $\boldsymbol{s}$.

5.4 Computing the Target Schedule Weights

In Chapter 4, we describe a distributed method for computing lexicographic max-min transmitter persistences to be employed by a random-scheduled MAC protocol. In this section, we review the operation of the algorithm with a focus on the changes required to compute discrete schedule weights, referring to Chapter 4 for a thorough treatment of the algorithm.

To compute lexicographic max-min schedule weights, the meaning of the offers and claims are changed slightly from Chapter 4 where they represent fractions of the communication channel. For this application, each *offer* and *claim* has two parts: There is a discrete part denoted by $\lfloor \mathit{offer} \rfloor$ and $\lfloor \mathit{claim} \rfloor$ and a fractional part denoted by $\{\mathit{offer}\}$ and $\{\mathit{claim}\}$. The discrete part identifies a schedule weight by its index, *i.e.*, $\lfloor \mathit{offer} \rfloor = l$ is offering schedule weight w_l. $\{\mathit{offer}\}$ is a measure of how close an auctioneer is to offering $\lfloor \mathit{offer} \rfloor + 1$ and $\{\mathit{claim}\}$ is a measure of how close a bidder is to claiming $\lfloor \mathit{claim} \rfloor + 1$. The fractional component enables differentiation between offers and claims that are *not* equal, but map to the same discrete schedule weight. This differentiation is required by the auctioneers as they identify the bidders in D_j^* (see Equation 5.9).

Bidder i constrains its claim to be no larger than the smallest offer in R_i or the desired schedule weight d_i. Then the claim of bidder i is

$$claim = \min\left(\{offers[j] : j \in R_i\}, d_i\right)$$

where the comparison between offers (or claims) takes both the discrete and fractional parts into account. *offer* is greater than *offer*$'$ when

$$\lfloor offer \rfloor > \lfloor offer' \rfloor$$

or when

$$\lfloor offer \rfloor = \lfloor offer' \rfloor \text{ and } \{offer\} > \{offer'\}.$$

offer is equal to *offer*$'$ when

$$\lfloor offer \rfloor = \lfloor offer' \rfloor \text{ and } \{offer\} = \{offer'\}.$$

Otherwise, *offer* is less than *offer*$'$. Auctioneer j computes

$$D_j^* = \{i : i \in D_j, claims[i] < offer\}. \tag{5.9}$$

D_j^* holds any bidder in D_j whose claim is either equal to its desired schedule weight, or is constrained elsewhere by another auction; conversely, $|D_j \setminus D_j^*|$ holds the set of bidders whose claims are constrained by the offer of auctioneer j. By accounting for both the discrete and fractional components, auctioneer j is able to identify any claim that is smaller than its offer (*i.e.*, the claim is constrained elsewhere), even if the discrete components of the claim and offer are equal. If $D_j^* = D_j$, then auctioneer j does not limit any bidder and its offer should be larger than the largest claim from D_j:

$$\lfloor offer \rfloor = \max\left(\lfloor claims[i] \rfloor : i \in D_j\right) + 1.$$

If D_j^* is a proper subset of D_j, then $\lfloor offer \rfloor$ should be maximized subject to

$$w_1 - 1 \geq \mathcal{I}\left(\lfloor offer \rfloor\right) \cdot |D_j \setminus D_j^*| + \sum_{i \in D_j^*} \mathcal{I}\left(\lfloor claims[i] \rfloor\right)$$

ensuring that the cost of claims from D_j^* plus the cost of the offers to bidders in $D_j \setminus D_j^*$ does not exceed $w_1 - 1$. Here, $\mathcal{I}(w_\ell)$ denotes the potential intersection of a schedule of weight w_ℓ with any other base schedule. The computation for the fractional part of the offer is

$$\{\mathit{offer}\} = \frac{(w_1 - 1) - \left(\mathcal{I}(\lfloor \mathit{offer} \rfloor) \cdot |D_j \setminus D_j^*| + \sum_{i \in D_j^*} \mathcal{I}(\lfloor \mathit{claims}[i] \rfloor) \right)}{(\mathcal{I}(\lfloor \mathit{offer} \rfloor + 1) - \mathcal{I}(\lfloor \mathit{offer} \rfloor)) \cdot |D_j \setminus D_j^*|}. \tag{5.10}$$

The numerator of Equation 5.10 is what is left after accounting for the claims of bidders in D_j^* and the offers to bidders in $D_j \setminus D_j^*$. The denominator of Equation 5.10 is the cost of increasing $\lfloor \mathit{offer} \rfloor$ by one. The result $\{\mathit{offer}\}$ is a number between 0 and 1; the closer it is to one, the closer *offer* is to offering $\lfloor \mathit{offer} \rfloor + 1$.

The distributed algorithm of Chapter 4 can be extended so that an $\lfloor \mathit{offer} \rfloor$, $\lfloor \mathit{claim} \rfloor$, $\{\mathit{offer}\}$, and $\{\mathit{claim}\}$ are embedded in the MAC header of both data and acknowledgement packets. An efficient representation is critical to maintaining low communication overhead for the algorithm. For the schedules listed in Table 5.2, the discrete values, $\lfloor \mathit{offer} \rfloor$ and $\lfloor \mathit{claim} \rfloor$, can be represented in as few as five bits. If the fractional values, $\{\mathit{offer}\}$ and $\{\mathit{claim}\}$, are represented with eight bits each, then each packet must transport an additional 26 bits. Accounting for both data and acknowledgement packets, this represents an overhead of 0.0089% for the 900 byte packets simulated here.

5.5 Simulation Results

In this section we evaluate both the schedules of Section 5.2 and the distributed computation of schedule weights defined in Section 5.3 with the aim to answer the following questions:

1. How are delay, throughput, and drop rate influenced by use of the variable-weight schedules of Section 5.2?

2. What schedule weight can a node expect to use and how does the selected weight correlate to expected delay and throughput?

3. How do the schedules perform in a dynamic network?

To this end, we propose and simulate the Variable-Weight and Adaptive Topology Transparent (VWATT) MAC protocol. VWATT implements 0.0008 second slots sized to hold a 900 byte payload, MAC header, physical framing, and a return acknowledgement. Slots are organized into frames of length v^2 where v is a prime power. Both slots and frames are synchronized. Schedules are constructed per Section 5.2 from a TD$(3, v, v)$ with the $m = \lfloor v/2 \rfloor$ variable-weight schedules of $\mathbb{F}_i$ assigned to node i. The schedules are constructed using primitive polynomials identified in [25]. VWATT uses the distributed algorithm of Section 4 to select schedule weights according to the criterion of Section 5.3. Receiving nodes acknowledge receipt of a packet immediately within the slot the packet is received. Unacknowledged packets are retransmitted for up to $0.0008 \cdot v^2$ seconds (the length of one frame) to take advantage of the transmission guarantee inherent in the schedule.

Simulations are run using the `ns-2` network simulator. Wireless nodes are equipped with omni-directional antennas with capabilities matching those of the 914 MHz Lucent WaveLAN DSS radio. The data rate for all transmissions is set to 11 megabits/second and the transmission and carrier sense ranges are 250 meters.

We simulate four traffic loads: 10 small demands, 10 large demands, 40 small demands, and 40 large demands. Each demand is modelled by a constant bit rate generator configured to generate 900 byte packets at 75 ± 50 packets/second for small

demands and 500 ± 50 packets/second for large demands. The demands are attached to randomly selected nodes (with no more than one demand per node). UDP provides transport layer services. The destination for each packet is selected at random from the neighbours of the sending node, constraining traffic to be single hop and maintaining fine grain control of network traffic. The four traffic loads combined with random placement of nodes enables the simulation of a wide variety of network scenarios.

5.5.1 The Delay, Drop Rate, and Throughput for VWATT

In the first column of Figure 5.3, delay, drop rate, and throughput are reported for VWATT operating with five different frame lengths: 64, 169, 361, 625, and 961. The number of nodes supported by VWATT is equal to the number of slots in a frame; for these simulations $N = 64, 169, 361, 625, 961$ nodes. However, the simulations are limited to 50-node networks with each node assigned exactly one set of schedules; the remaining $N - 50$ schedule sets are not included in the simulation. Each simulation effectively models part of a potentially larger network.

To provide context, we include results for three additional MAC protocols (see columns 2 through 4 of Figure 5.3). The underlying slot structure and framing match those of VWATT; they differ only in their generation of transmission schedules.

- The first, called the Topology Transparent (TT) MAC, limits nodes to the base schedules described in Section 5.2, effectively implementing the scheme of [16], [53], and [83]. Of the four simulated MAC protocols, TT and VWATT are the only MACs to provide a guarantee on maximum delay. Comparisons between TT and VWATT highlight the effects of variable-weight schedules.

- The second, called the Variable-Weight and Adaptive (VWA) MAC, employs random schedules generated at the start of every frame and upon any change to

the node's allocated schedule weight. The selection of schedule weights matches that of VWATT; because the schedules are random, VWA does not offer a guarantee on maximum delay. Comparisons between VWA and VWATT reveal the effect that topology transparent schedules have on network performance.

- The third MAC is the nominal configuration of ATLAS from Chapter 4 which employs random schedules with persistences matching the lexicographic max-min allocation to transmitters *i.e.*, the TLA allocation. The only change to ATLAS with respect to the simulations of Chapter 4 is in the packet retry mechanism: here, unacknowledged packets are retransmitted for the period of one frame, rather than a maximum count of 10. We include results from ATLAS as a point of reference demonstrating the performance characteristics of a random scheduled MAC that makes no attempt to achieve a delay guarantee.

The results of Figure 5.3 come from simulations of 100 randomly generated network scenarios, 25 of each traffic load (10 small, 10 large, 40 small, and 40 large demands). Each network contains 50 nodes randomly placed in a 1500 meter by 300 meter area for an expected neighbourhood size and standard deviation of 13.8 and 3.8 nodes, respectively. Each network scenario is simulated once for each combination of MAC protocol and frame length. Simulations are run for 15 seconds following a 1 second warm-up interval. Delay is measured per packet. Throughput and drop rate are measured per node over each frame.

The first row in Figure 5.3 reports maximum observed packet delay along with the theoretical delay guarantee provided by the schedule. Each set of bars reports maximum observed delay for the given frame length under all four traffic loads. The theoretical maximum delay is independent of traffic load and is plotted as a step function, increasing with the frame length. VWA and ATLAS do not provide

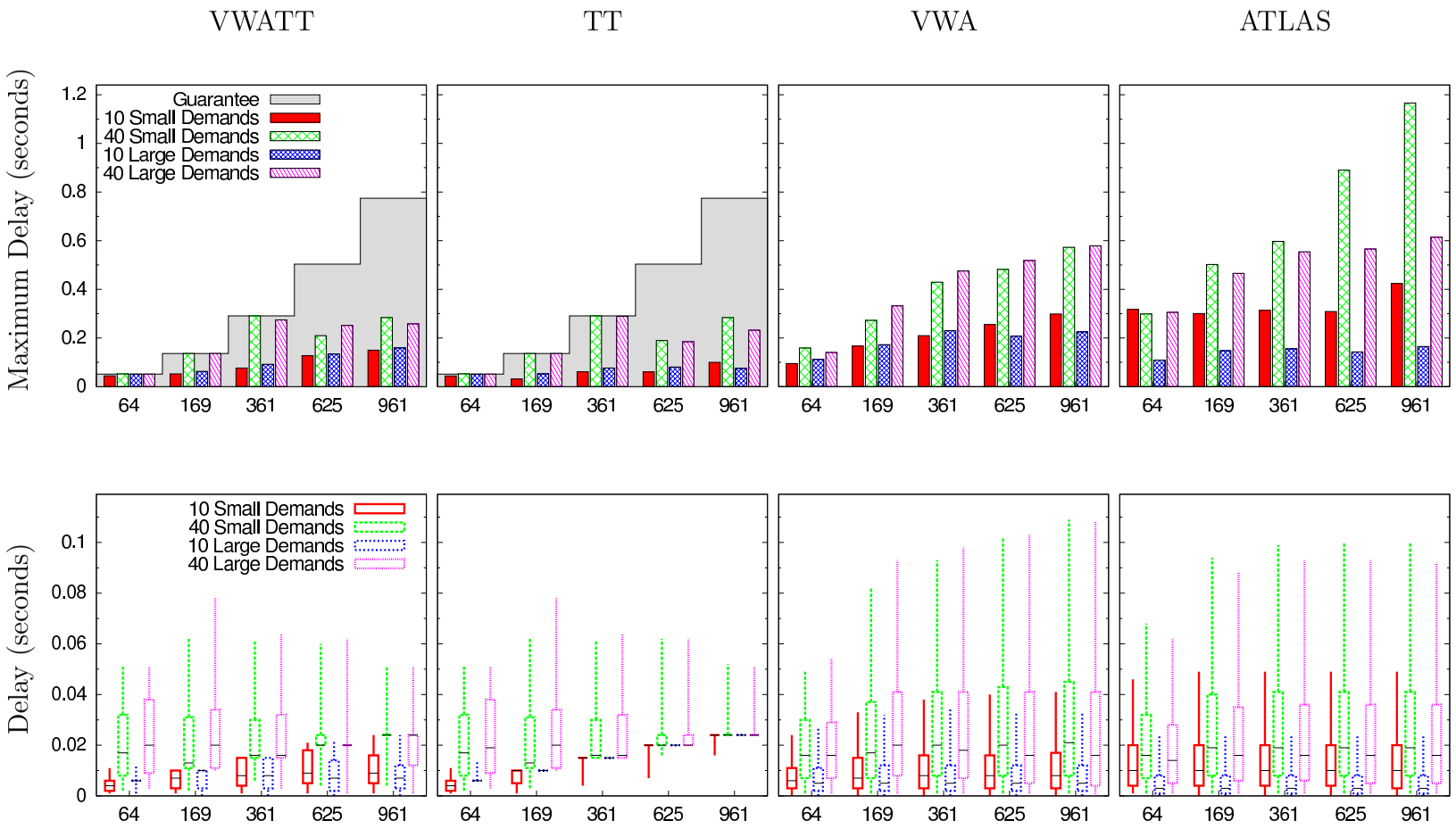


Figure 5.3: Delay, throughput, and drop rate for the four MAC protocols. The x-axis in each plot identifies the frame length of the MAC protocol. This figure is continued on page 132.

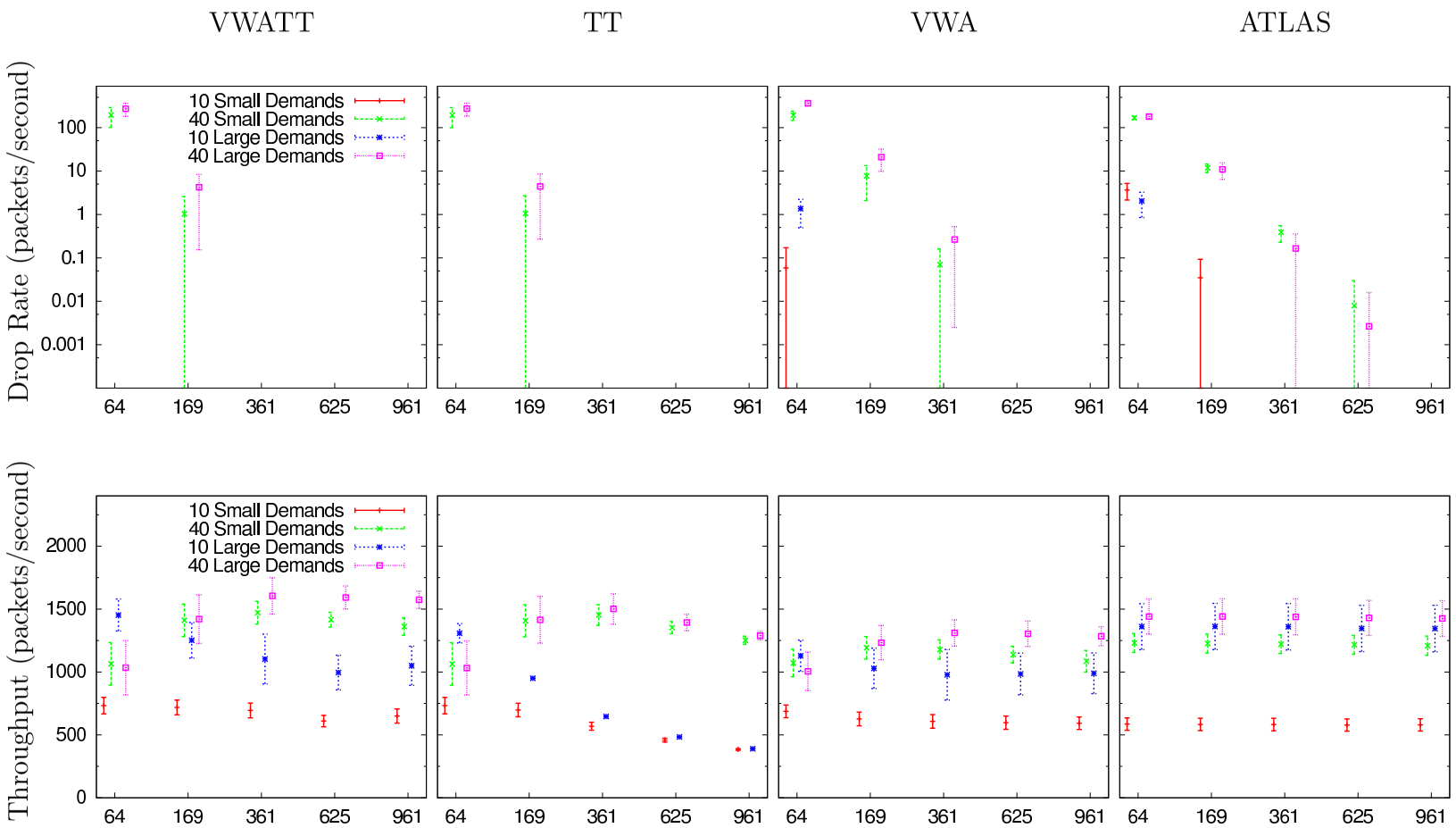


Figure 5.3: Delay, throughput, and drop rate for the four MAC protocols. The figure is continued from page 131.

a guarantee on maximum delay; only observed maximum delays are reported for these two MACs. The box-and-whisker plots in the second row report percentiles for the recorded packet delays. The bottom and top whiskers identify the 5^{th} and 95^{th} percentiles; the bottom and top of the boxes identify the 25^{th} and 75^{th} percentiles; the line in the box identifies the median packet delay. The third row reports MAC drop rate and the fourth reports MAC throughput. Here, and for the rest of the chapter, error bars identify standard deviation from the mean.

The first column in Figure 5.3 reports on the performance of VWATT. Neighbourhood sizes often exceed D_{max} for frame lengths of 64 and 169. This is evident in the maximum recorded delays, large drop rates, and lower throughputs recorded for networks loaded with 40 demands (small or large). At a frame length of 361, D_{max} is large enough to accommodate the majority of neighbourhoods in the network; dropped packets are mostly avoided, resulting in improved throughput.

The second column of Figure 5.3 shows the performance of the TT MAC. Delays above the median (including maximum observed delays) change very little compared to those of VWATT. However, delays below the median increase sharply because the shorter delays tend to come from higher weight schedules that are not employed by TT. Drop rates remain unchanged. Throughput degrades slightly for networks with 10 small, 40 small, and 40 large demands; the degradation is dramatic for networks with 10 large demands. Loss of throughput is more pronounced at larger frame lengths as the base schedule weight of v gets smaller relative to the frame length of v^2.

Columns three and four of Figure 5.3 report on the two random scheduled MACs: VWA and ATLAS. Delays in the upper quartile are noticeably higher when random schedules are employed, confirming the damping effect that topology transparent

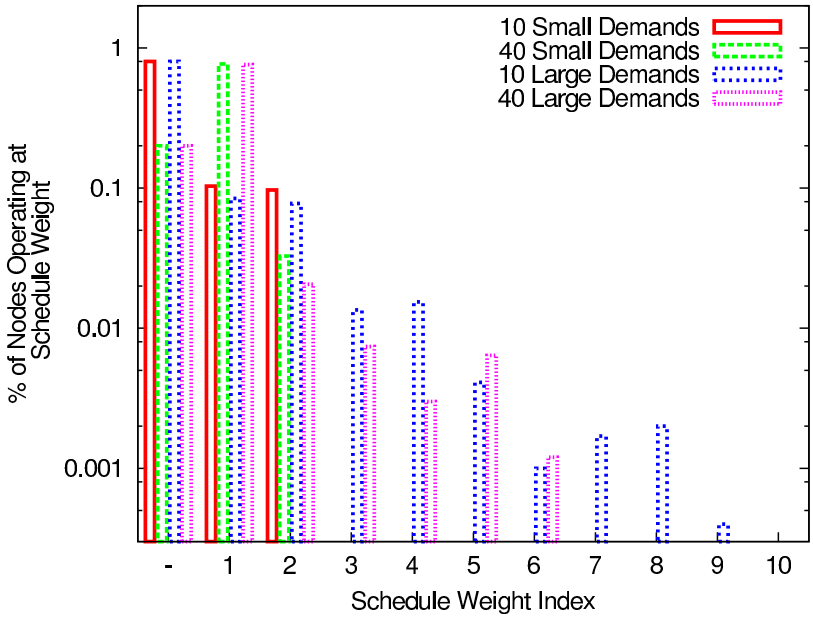
schedules have on large delays. The random schedules also result in an increase in drop rate for frame lengths of 64, 169 and 361. The throughput of VWA is lower than that of VWATT for all four traffic loads. ATLAS and VWATT achieve comparable throughput for all network scenarios except those loaded with 10 large demands. For this traffic load, ATLAS is able to implement higher schedule weights, yielding better expected throughput.

5.5.2 Understanding the Schedule Weights

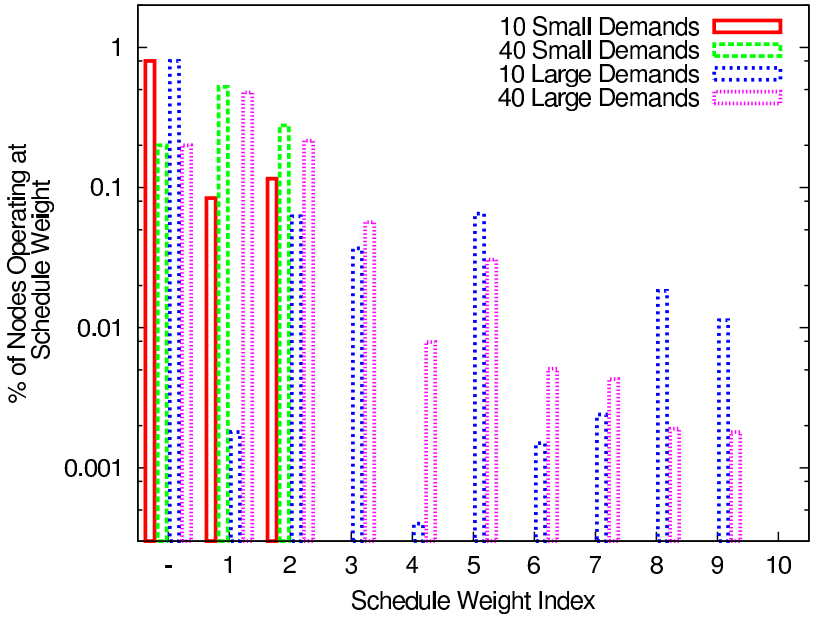
Figure 5.4 shows the distribution of schedule weights for VWATT configured with a frame length of 361 slots. The distribution in Figure 5.4a comes from 800 dense network scenarios, 200 of each traffic load, with each scenario consisting of 50 nodes randomly placed in a 300 meter by 1500 meter area for an expected neighbourhood size of 13.8 nodes with standard deviation of 3.8 nodes. The distribution in Figure 5.4b comes from 800 sparse network scenarios, 200 of each traffic load; these scenarios consist of 50 nodes randomly placed in a 300 meter by 4500 meter area for an average neighbourhood size of 5.5 nodes and standard deviation of 2.1 nodes. The largest schedule weight index is $m = \lfloor 19/2 \rfloor = 9$. The leftmost set of bars, labelled with a hyphen on the x-axis, represents the distribution of unloaded nodes. The y-scale is logarithmic.

For both sparse and dense topologies, nodes loaded with small demands employ schedule weights w_1 or w_2; these small demands do not justify a higher schedule weight. Nodes loaded with large demands occasionally employ larger weight schedules, most often in sparse topologies and networks with only a few large demands.

Figure 5.5 reports expected delay and throughput by schedule weight. The data comes from simulations of all 1600 network scenarios of Figure 5.4 and is collected for 15

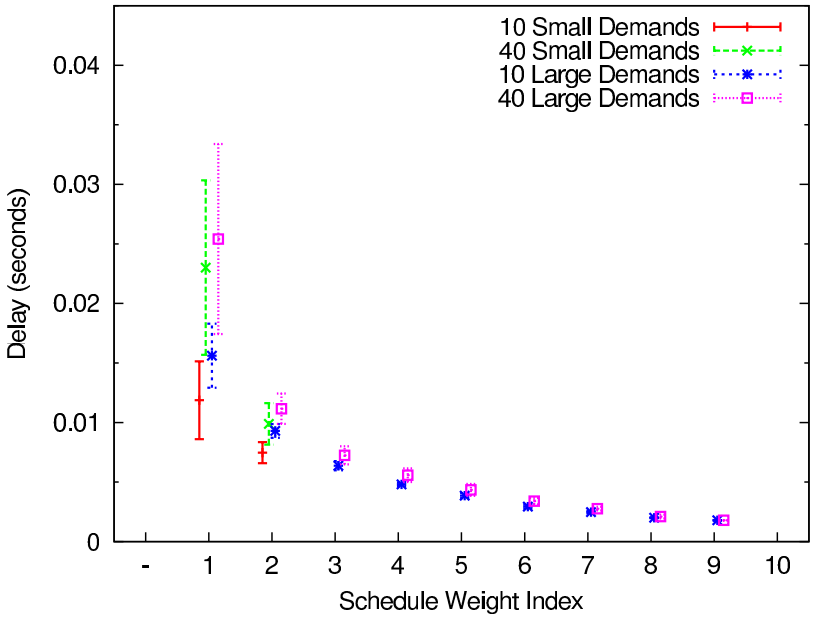


(a) 50 nodes in a 300 meter by 1500 meter area.

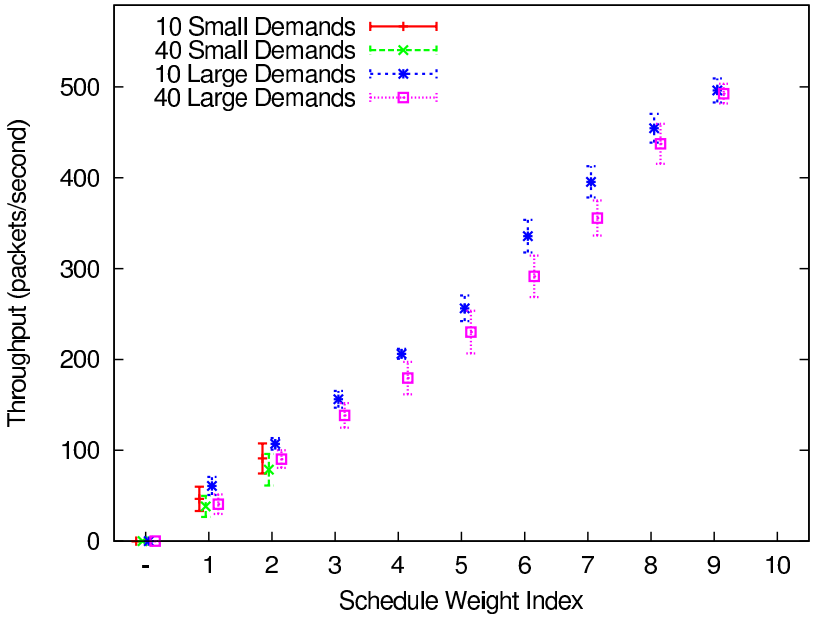


(b) 50 nodes in a 300 meter by 4500 meter area.

Figure 5.4: Distribution of schedule weights for VWATT with a frame length of 361 slots.



(a) Average packet delay.



(b) Average per node throughput.

Figure 5.5: Throughput and delay organized by schedule weight for VWATT with a frame length of 361 slots.

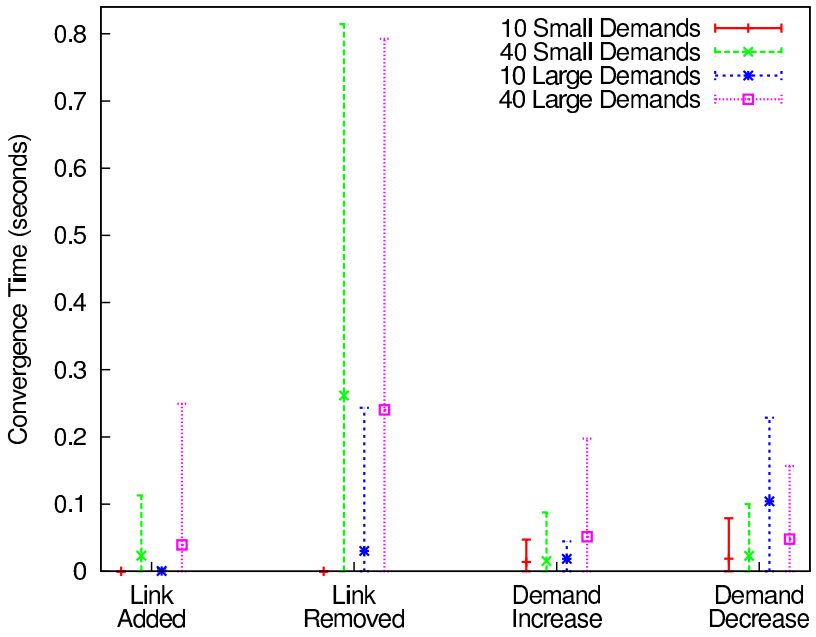


Figure 5.6: Convergence times after a change in the network for VWATT with a frame length of 361 slots.

seconds following a 1 second warm up period. The simulations show a remarkably tight correlation between schedule weight and expected delay and throughput, suggesting schedule weight may be a good predictor of MAC-layer performance; it could be used to inform admission control and routing decisions made by the upper layers of the network stack.

5.5.3 Adapting to Changes in the Network

Figure 5.6 reports the time required for VWATT to converge to the lexicographic max-min schedule weights following a change in the network. Four types of changes are simulated: addition of a link, removal of a link, a demand increase, and a demand decrease. Each type of change is simulated 400 times, 100 for each traffic load. Networks contain 50 nodes randomly placed within a 300 meter by 1500 meter simulation area. For simulations of added and removed links, 49 of the 50 nodes are statically placed; the remaining node is given a location and trajectory to create, or

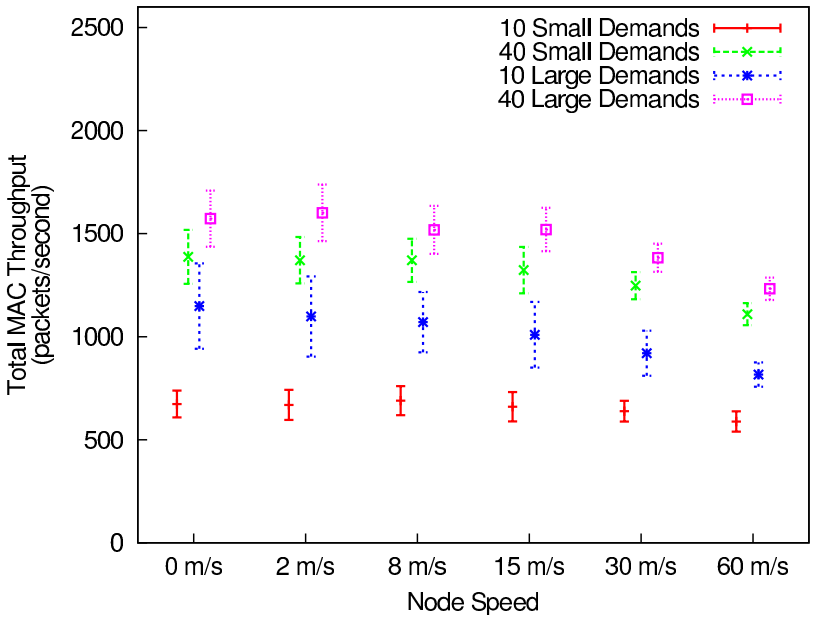


Figure 5.7: Total throughput of VWATT at various levels of mobility.

break, a single link in the network topology. For simulations with a single demand increase, one node is initialized with a demand of 75 packets/second which is increased to 500 packets/second; for a single demand decrease, the node's demand is initialized to 500 packets/second and reduced to 75 packets/second. The network is given time to stabilize on the lexicographic max-min schedule weights before the network change occurs. Convergence times are measured from the time of the network change to the time the nodes converge on their new lexicographic max-min schedule weights.

The simulations show VWATT to respond quickly to changes in the network, in under 0.1 seconds in most cases. For removed links, the longer convergence times are dominated by the time required to detect the lost neighbours, which only happens after a lapse of 1.44 seconds in communication (equal to five frames). We refer to Section 4.4 starting on page 88 for an in-depth evaluation of the distributed algorithm used to compute the lexicographic max-min schedule weights.

While Figure 5.6 shows how the MAC responds to a single change in the network, Figure 5.7 and Figure 5.8 show how it performs in a continuously changing environment. In Figure 5.7, throughput is reported for networks with varying degrees of mobility: from 0 meters/second to 60 meters/second. We simulate 200 network scenarios for each node speed, 50 of each traffic load. Node movements are generated using the steady-state mobility model generator of [49] configured with a pause time of zero and a 300 meters by 1500 meters simulation area. For each simulation, throughput is measured for 15 seconds following a 1 second warm up interval. VWATT's ability to handle node mobility is striking; the throughput of networks with nodes moving at 60 meters/second degrades by less than 28% compared to the throughput of static networks. Figure 5.8 reports delay for nodes in the network scenarios of Figure 5.7 with node speeds set to 30 meters/second. Each point identifies the expected (x-axis) and variation (y-axis) in packet delay for a single node. The network scenarios are simulated twice, once with VWATT and once with ATLAS. The scatter plot shows VWATT to reduce both expectation and variation in delay.

5.5.4 Review of Simulation Results

The simulations discussed in Section 5.5.1, Section 5.5.2, and Section 5.5.3 are motivated by the three questions posed on page 127. We repeat them here, summarizing answers to each:

1. How are delay, throughput, and drop rate influenced by use of the variable-weight schedules of Section 5.2?

   The results of Section 5.5.1 show the variable weight schedules to maintain both delay and drop rate while improving throughput compared to constant-weight topology transparent schedules. Compared to ATLAS, the variable-

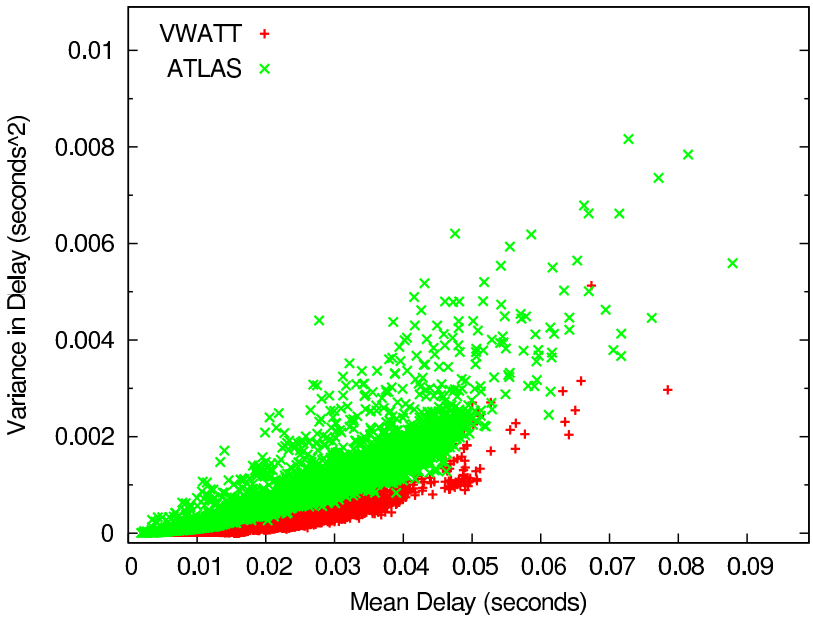


Figure 5.8: Average delay vs. variation in delay for VWATT with nodes moving at 30 meters/second.

weight topology-transparent schedules maintain throughput while reducing delay variance and worst-case delay.

2. What schedule weight can a node expect to use and how does the selected weight correlate to expected delay and throughput?

   The results of Section 5.5.2 suggest that lower weight schedules are used most frequently. However, higher weight schedules are employed in sparse networks with large demands. Furthermore, both delay and throughput are highly correlated to schedule weight; delay decreases and throughput increases as schedule weight increases. The tight correlation may allow nodes to predict their performance based on their schedule weight.

3. How do the schedules perform in a dynamic network?

   In Section 5.5.3, VWATT is observed to adapt to changes in the network with convergence times comparable to those of ATLAS. Throughput is maintained in

mobile networks, even those high node speeds. Compared to ATLAS, VWATT further reduces the already low variation in delay experienced at each node.

5.6 Discussion

In this section, we discuss limitations of the variable-weight topology transparent schedules of Section 5.2.

5.6.1 Large Maximum Delays

For schedules with a large D_{max}, the maximum delays reported in Table 5.2 are of limited value. However, these delays are directly related to the slot length, a function of data rate and packet size. The 0.0008 second slots simulated here are sized to hold 900 byte packets over an 11 megabits/second channel. At higher data rates, 900 byte packets can fit into smaller slots, tightening the bound on maximum delay. Given that data rates for existing technologies are approaching 600 megabits/second [47], it is reasonable to expect a significant reduction in slot size and a corresponding improvement in maximum delay.

5.6.2 A Guarantee in Neighbourhoods larger than D_{max}

A limitation of our variable-weight schedules (and of topology transparent scheduling in general) is the loss of a delay guarantee in neighbourhoods larger than D_{max}. It is possible to extend the guarantee to neighbourhoods larger than D_{max} by adding a TDMA schedule to the set of schedules assigned to each node. For schedules derived from TD$(3, v, v)$s, the number of supported nodes and the frame length are both v^2, allowing for seamless integration of TDMA. Nodes in neighbourhoods larger than D_{max} can alternate between their TDMA and base schedules to maintain a guarantee both inside and outside the dense neighbourhood. The maximum delay for nodes

alternating between schedules is equal to the length of two frames (as opposed to one) and the throughput of nodes running TDMA in large networks is poor. However, the guarantee is maintained across the network, regardless of neighbourhood size.

5.6.3 Intersection Cost of Higher Weight Schedules

In Section 5.5.1, we observe VWATT to outperform ATLAS in throughput for all network scenarios except those loaded with only 10 large demands (*i.e.*, sparsely distributed large demands). The lower throughput for these scenarios is a consequence of the progressive potential for intersection of higher weight schedules; nodes in neighbourhoods with only a few large demands are not permitted use of schedules with weights as large as they could otherwise use. A potential area for future research is the design of schedules with lower intersection properties to achieve better performance in these scenarios.

5.6.4 Complications of Fixed Schedules

The VWATT MAC has two limitations common to all fixed schedule MAC protocols, but not shared by either contention-based schemes such as IEEE 802.11 [45] or random scheduled schemes such as ATLAS: VWATT limits the number of nodes in the network to a maximum of v^2 and presumes a priori provisioning of schedules among the nodes. Both limitations might be mitigated through a mechanism for dynamic provisioning and spatial reuse of schedules.

5.7 Summary

In this chapter, we have given a construction for a variable-weight topology transparent schedule that enables the dynamic selection of schedule weights to accommodate variations in topology and traffic load. The schedules are integrated into ATLAS to

give an adaptive topology transparent MAC called VWATT. Simulation results show VWATT to improve delay, throughput, and drop rate compared to random scheduled MACs, such as ATLAS, while offering a guarantee on maximum delay. In the next chapter, we summarize the contributions and discuss avenues for potential future work.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Summary

The contributions of this thesis are threefold: (1) The TLA allocation is proposed as a target channel allocation for multi-hop networks. (2) An adaptive and distributed computation of the TLA allocation is described and integrated into ATLAS, a random-scheduled MAC that implements persistences according to the TLA allocation. (3) A variable-weight topology transparent schedule is constructed that can adapt to network conditions while maintaining a guarantee on maximum delay. In this section, we review the specifics of each contribution.

The TLA Allocation: Prior work on MAC protocol design has focussed on specific challenges such as collision avoidance, minimization of delay, and spatial reuse. Observing that all MAC protocols, regardless of end goal or underlying strategy, determine the operating persistences of transmitters, we argue that persistences should follow the TLA allocation, a lexicographically max-min allocation that treats transmitters as demands and receivers as resources. With this in mind, a straightforward centralized algorithm along with an auction-based distributed algorithm are proposed for computing TLA persistences. The correctness of both algorithms is established. The auction-based technique can be implemented with minimal communication that can be piggybacked on existing traffic. Simulation results demonstrate that the distributed algorithm succeeds in finding TLA persistences in a reasonable period of time. When equipped with these persistences, a simple persistence-based MAC protocol operating with TLA persistences can outperform IEEE 802.11. As developed initially, the determination of TLA persistences is undertaken only once, and applies only as long as the topology and demands remain unchanged.

Adaptive Topology- and Load-Aware Scheduling: To enable use of the TLA allocation in real wireless networks, we give a distributed auction-based algorithm that converges continuously on the TLA allocation, adapting to changes in both topology and traffic load. The utility of the algorithm is demonstrated through integration into ATLAS which we simulate under a wide variety of network settings and scenarios. The new MAC performs well in simulations of mobile networks and provides MAC layer services supporting multi-hop TCP. It remains to be seen how ATLAS handles a more realistic physical medium where transmitters interfere beyond the range of their transmissions and asymmetric communication is common. Nevertheless, the body of results presented suggests that the distributed algorithm, when employed at the wireless MAC layer, can effectively inform the selection of transmitter persistences, enabling a simple MAC protocol to provide robust, reliable, and scalable services. The application of the distributed algorithm is not restricted to the computation of transmitter persistences. It has the potential to inform routing and admission control decisions and to enable differentiation of service at the MAC layer. In this context, the algorithm provides a potential solution to the immediate challenge of medium access control, but also shows promise as a tool for use in network protocol design in general.

Variable-Weight and Adaptive Topology Transparent Scheduling: Use of a variable-weight schedule requires two questions to be answered: (1) Assuming a node can choose between multiple schedules of different weights, what weight should it choose? (2) How can a node arrive at the correct schedule weight in an ad hoc network that lacks centralized control? The first two contributions of this thesis—the definition of the TLA allocation and the design of ATLAS—lay the framework for answering these questions. The final contribution of this thesis is the construction and evaluation of a variable-weight topology transparent schedule, the first of its kind. The variable-weight

schedule and the distributed algorithm from ATLAS are integrated to make VWATT, a MAC protocol that adjusts schedule weights to accommodate local topology and traffic load while maintaining a guarantee on maximum delay. Simulations show VWATT to reduce delay (maximum and expected) relative to random schedules, to increase throughput compared to constant-weight topology transparent schedules, and to adapt quickly to changes in topology and traffic load. The results reported here suggest variable-weight topology transparent scheduling as a viable approach to medium access control, and a promising area for future work.

6.2 Future Work

The contributions of this thesis offer a number of potential topics for future research. We discuss several here.

6.2.1 Evaluation of ATLAS in a Testbed

Given the remarkable performance ATLAS has achieved in simulation, a natural next step is to evaluate the protocol in a testbed of real radios. Such an experiment is necessary to understand how ATLAS responds to a wireless channel that exhibits asymmetric communication, interference beyond the range of transmission, and temporal fading. The primary difficulty in testing ATLAS on real hardware is finding a testbed that (1) enables replacement/modification of the medium access control layer and (2) maintains sufficiently accurate time to support synchronization of slot boundaries. The MAC processors described in [90] may be sufficient to enable a testbed evaluation of ATLAS.

6.2.2 Differentiated Service using ATLAS

With its low variation in delay and throughput, and almost zero drop rate (see Figure 3.11, Figure 3.12, Figure 3.13, and Figure 4.10), ATLAS in itself promises an improved quality of service (QoS) at the MAC layer. The weighted-TLA allocation—computed by weighted bidders per Section 4.2.3—could be used to further improve MAC layer QoS.

An intriguing application is the implementation of differentiated service at the MAC layer. IEEE 802.11e [46] enhances the distributed coordination function by implementing four access categories; from lowest to highest priority they are Background, Best Effort, Video, and Voice. Four instances of the back-off algorithm are run, one per access category, each with its own queue. The probability of transmission of each access category is manipulated independently through selection of contention window size and inter-frame space, leveraging the inherent unfairness of IEEE 802.11 to enhance the network's quality of service. Higher priority traffic is permitted (encouraged) to capture the channel from lower priority traffic flows.

Similar results can be achieved by four instances of the distributed algorithm of Section 4, each computing the allocation for a single access category. Prioritization is achieved through dynamic coordination of the four auction capacities at each node. A potential strategy sets the capacity for each access category equal to one minus the allocation to higher priority access categories. As a result, higher priority auctions are permitted to starve lower priority auctions of capacity, effectively distributing channel access to high priority traffic. Alternatively, auction capacities can be selected to ensure a minimum or maximum percentage of the channel is offered to an access category. Although traffic within an access category follows the lexicographic max-

min allocation, bidder's can weight their claims to prioritize traffic *within* the access category.

Regardless of the approach taken, QoS remains a cross-layer problem requiring a cross-layer solution. An interesting area of research is the integration of ATLAS (or a variant of ATLAS) into a cross-layer approach to QoS.

6.2.3 The TLA Allocation and the IEEE 802.11 Contention Window

We have used the distributed algorithm of Chapter 4 to compute persistences to be employed within two slotted MAC protocols (ATLAS and VWATT). Alternatively, the TLA allocation can be computed by running the distributed algorithm on top of the IEEE 802.11 MAC, embedding the bidder claims and auction offers in the headers of existing RTS, CTS, data, and acknowledgement messages. Given knowledge of their TLA allocations, nodes can set their contention window sizes directly to achieve their desired TLA persistence, eliminating the need for (and negative side effects of) binary exponential back off. An advantage of this approach is its heavy reuse of an existing and pervasive MAC layer protocol. Depending on the platform, it may be possible to experiment using off-the-shelf hardware, passing offers and claims through extended headers and manipulating the contention window by controlling the minimum and maximum window size.

6.2.4 Intersection Properties of VWATT Schedules

In Chapter 5, we give a construction for a variable-weight topology transparent schedule based on transversal designs. One weakness of this construction is the progressive potential to intersect other schedules seen in the higher weight schedules. Any base schedule can intersect another base schedule in at most one slot; each increase to the schedule weight comes at a cost of two more intersections. This limits the use of

higher weight schedules, reducing the throughput achieved in networks sparsely loaded with large demands. An area for potential research is the construction of schedules with lower intersection to improve throughput and to provide a richer set of possible weights while maintaining the same delay guarantee.

6.2.5 The TLA Allocation and Topology-Aware Scheduling

In Chapter 5, we employ the distributed algorithm of Chapter 4 to compute schedule weights for both random and topology transparent schedules. An alternative, and perhaps more ambitious, approach is to employ the distributed algorithm to inform the number of slots assigned to each node in a topology-aware MAC that computes collision-free schedules. Only a few of the topology-aware schemes allow a node to reserve more than one slot in a frame (*i.e.*, [98], [87], [50]), and those do not define how many slots a node *should* be permitted to reserve. It is possible for a node to unfairly reserve more than its share, preventing others from communicating. The TLA allocation might be used to inform the topology-aware scheduler of a permissible number of slots to be reserved for a node, given the current topology and traffic load.

6.3 Concluding Remarks

Our motivation for this work has been the design of medium access control techniques that embody the strengths of contention- and schedule-based schemes without succumbing to the weaknesses of either. To a large extent, ATLAS and VWATT accomplish this goal. Both protocols achieve remarkably small variations in delay, throughput, and drop rate, while maintaining expected delay, throughput, and drop rate compared to IEEE 802.11. In addition, VWATT offers a guarantee on maximum delay when neighbourhood sizes are not too large, an accomplishment that is not possible for purely contention-based MAC protocols. Combined with their ability to

adapt quickly to changes in the network, ATLAS and VWATT have the potential to achieve both the stable performance characteristics common to schedule-based protocols and the agility found in contention-based schemes. It remains to be seen how ATLAS and VWATT handle a more realistic physical medium where transmitters interfere beyond the range of their transmissions and asymmetric communication is common. Nevertheless, the body of results presented here suggest that both protocols have the potential to perform well in mobile ad hoc networks, opening doors for a number of interesting avenues for future work.

REFERENCES

[1] *RFC 768: User Datagram Protocol*, 1980.

[2] *RFC 793: Transmission Control Protocol*, 1981.

[3] *RFC 826: Ethernet Address Resolution Protocol*, 1982.

[4] *RFC 765: File Transfer Protocol*, 1985.

[5] *RFC 2581: TCP Congestion Control*, 1991.

[6] *RFC 1323: TCP Extensions for High Performance*, 1992.

[7] *RFC 2018: TCP Selective Acknowledgement Options*, 1996.

[8] *RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing*, 2003.

[9] N. Abramson. The ALOHA system - Another alternative for computer communications. In *Proceedings of the Fall Joint Computer Conference*, pages 281–285, 1970.

[10] Apple. The MacBook Air. http://www.apple.com/macbook-air/.

[11] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[12] L. Bao. MALS: Multiple access scheduling based on latin squares. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 315–321, 2004.

[13] N. Bayer, B. Xu, and S. Hischke. An architecture for connecting ad hoc networks with the IPv6 backbone (6Bone) using a wireless gateway. In *Proceedings of the European Wireless Conference*, 2004.

[14] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A medium access protocol for wireless LANs. In *Proceedings of the ACM Conference on Communications Architectures, Protocols and Applications (SIGCOMM'94)*, pages 212–225, 1994.

[15] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proceedings of the 11th Annual*

International Conference on Mobile Computing and Networking (MOBICOM'05), pages 31–42, 2005.

[16] I. Chlamtac and A. Faragó. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Transactions on Networking*, 2(1):23–29, 1994.

[17] I. Chlamtac and A. Faragó. Time-spread multiple-access TSMA protocols for multihop mobile radio networks. *IEEE/ACM Transactions on Networking*, 5(6):804–812, 1997.

[18] I. Chlamtac, A. Faragó, and H. Y. Ahn. A topology transparent link activation protocol for mobile CDMA radio networks. *IEEE Journal on Selected Areas in Communications*, 12(8):1426–1433, 1994.

[19] I. Chlamtac, A. Faragó, A. D. Myers, V. R. Syrotiuk, and G. V. Záruba. ADAPT: A dynamically self-adjusting media access control protocol for ad hoc networks. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'99)*, pages 11–15, 1999.

[20] I. Chlamtac, A. D. Myers, V. R. Syrotiuk, and G. V. Záruba. An adaptive medium access control (MAC) protocol for reliable broadcast in wireless networks. In *Proceedings of the IEEE International Conference on Communications (ICC'00)*, pages 1692–1696, 2000.

[21] I. Chlamtac and S. S. Pinter. Distributed nodes organization algorithm for channel access in a multihop dynamic radio network. *IEEE Transactions on Computers*, C-36(6):728–737, 1987.

[22] W. Chu, C. J. Colbourn, and V. R. Syrotiuk. The effects of synchronization on topology-transparent scheduling. *Wireless Networks*, 12(6):681–690, 2006.

[23] W. Chu, C. J. Colbourn, and V. R. Syrotiuk. Slot synchronized topology-transparent scheduling for sensor networks. *Computer Communications*, 29(4):421–428, 2006.

[24] F. R. K. Chung, J. A. Salehi, and V. K. Wei. Optical orthogonal codes: Design, analysis, and applications. *IEEE Transactions on Information Theory*, 35(3):595–604, 1989.

[25] R. Church. Tables of irreducible polynomials for the first four prime moduli. *Annals of Mathematics*, 36(1):198–209, 1935.

[26] Cisco. Connections Counter: The Internet of Everything in Motion. http://newsroom.cisco.com/feature-content?articleId=1208342.

[27] C. J. Colbourn and J. H. Dinitz. *Handbook of Combinatorial Designs*. CRC Press, second edition, 2007.

[28] C. J. Colbourn, A. C. H. Ling, and V. R. Syrotiuk. Cover-free families and topology-transparent scheduling for MANETs. *Designs, Codes and Cryptography*, 32:65–95, 2004.

[29] C. J. Colbourn and V. R. Syrotiuk. Scheduled persistence for medium access control in sensor networks. In *Proceedings from the First IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS'04)*, pages 264–273, 2004.

[30] C. J. Colbourn, V. R. Syrotiuk, and A. C. H. Ling. Steiner systems for topology-transparent access control in MANETs. In *Proceedings of the Second International Conference on Ad Hoc Networks and Wireless (ADHOC-NOW'03)*, pages 247–258, 2003.

[31] A. Colvin. CSMA with collision avoidance. *Computer Communications*, 6(5):227–235, 1983.

[32] J. Deng, P. K. Varshney, and J. Haas. A new backoff algorithm for the IEEE 802.11 distributed coordination function. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulations (CNDS'04)*, 2004.

[33] P. Erdös, P. Frankl, and Z. Rüredi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51(1):79–89, 1985.

[34] S. Even, O. Goldreich, S. Moran, and P. Tong. On the np-completeness of certain network testing problems. *Networks*, 14(1):1–24, 1984.

[35] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[36] J. J. Garcia-Luna-Aceves and A. Tzamaloukas. Receiver-initiated collision avoidance in wireless networks. *Wireless Networks*, 8:249–263, 2002.

[37] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang. TCP over wireless multi-hop protocols: Simulation and experiments. In *Proceedings of the 1999 IEEE International Conference on Communication (ICC'99)*, pages 1089–1094, 1999.

[38] C. Gomez and J. Paradells. Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, 48(6):92–101, 2010.

[39] Google. Google Glass. http://www.google.com/glass/start/.

[40] Z. Haas and J. Deng. On optimizing the backoff interval for random access schemes. *IEEE Transactions on Communications*, 51(12):2081–2090, December 2003.

[41] J. Hastad, T. Leighton, and B. Rogoff. Analysis of backoff protocols for multiple access channels. In *Proceedings of the 19th annual ACM Symposium on Theory of Computing (STOC'87)*, pages 740–744, 1987.

[42] A. S. Hedayat, N. J. A. Sloane, and J. Stufken. *Orthogonal Arrays: Theory and Application*. Springer, first edition, 1999.

[43] X. L. Huang and B. Bensaou. On max-min fairness and scheduling in wireless ad-hoc networks: Analytical framework and implementation. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'01)*, pages 221–231, 2001.

[44] IEEE. *IEEE 802.3, CSMA/CD Ethernet access method*, 1983.

[45] IEEE. *IEEE 802.11, Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, 1997.

[46] IEEE. *IEEE 802.11e, Enhancements: QoS, including packet bursting*, 2007.

[47] IEEE. *IEEE 802.11ac D4.0: Enhancements for Very High Throughput for Operation in Bands below 6 GHz*, 2012.

[48] Intel. Ultrabook. www.intel.com/Ultrabook/.

[49] J. Boleng, N. Bauer, T. Camp, and W. Navidi. Random Waypoint Steady State Mobility Generator (mobgen-ss). http://toilers.mines.edu/.

[50] G. Jakllari, M. Neufeld, and R. Ramanathan. A framework for frameless TDMA using slot chains. In *Proceedings of the 9th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS'12)*, 2012.

[51] L. B. Jiang and S. C. Liew. Proportional fairness in wireless LANs and ad hoc networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'05)*, pages 1551–1556, 2005.

[52] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imielinski and H. F. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[53] J. Ju and V. O. K. Li. An optimal topology-transparent scheduling method in multihop packet radio networks. *IEEE/ACM Transactions on Networking*, 6(3):298–305, 1998.

[54] J.-H. Ju and V. O. K. Li. TDMA scheduling design of multihop packet radio networks based on latin squares. *IEEE Journal on Selected Areas in Communication*, 17(8):1345–1352, 1999.

[55] P. Karn. MACA - a new channel access method for packet radio. In *Proceedings from the ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pages 134–140, 1990.

[56] V. Kawadia and P. R. Kumar. Experimental investigations into TCP performance over wireless multihop networks. In *Proceedings of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis (E-WIND'05)*, pages 29–34, 2005.

[57] D. Koutsonikolas, J. Dyaberi, P. Garimella, S. Fahmy, and Y. C. Hu. On TCP throughput and window size in a multihop wireless network testbed. In *Proceedings of the 2nd ACM International Workshop on Wireless network testbeds, experimental evaluation and characterization (WiNTECH'07)*, 2007.

[58] K. Leung and V. O. K. Li. Transmission control protocol (TCP) in wireless networks: Issues, approaches, and challenges. *IEEE Communications Surveys & Tutorials*, 8:64–79, 2006.

[59] Y. Liu, L. Zhang, V. O. K. Li, K.-C. Leung, and W. Zhang. Topology-transparent scheduling in mobile ad hoc networks supporting heterogeneous quality of service guarantees. In *Proceedings of the 46th Annual Conference on Information Sciences and Systems (CISS 2012)*, pages 1–6, 2012.

[60] E. L. Lloyd and S. Ramanathan. On the complexity of distance-2 coloring. In *Proceedings of the 4th International Conference on Computing and Information (ICCI'92)*, pages 71–74, 1992.

[61] J. Lutz, C. J. Colbourn, and V. R. Syrotiuk. Apples and oranges: Comparing schedule- and contention-based medium access control. In *Proceedings of the 13th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'10)*, pages 319–326, 2010.

[62] J. Lutz, C. J. Colbourn, and V. R. Syrotiuk. Variable weight sequences for adaptive scheduled access in MANETs. In *Proceedings of Sequences and their Applications (SETA'12)*, pages 53–64, 2012.

[63] J. Lutz, C. J. Colbourn, and V. R. Syrotiuk. ATLAS: Adaptive topology- and load-aware scheduling. *under review*, 2013.

[64] J. Lutz, C. J. Colbourn, and V. R. Syrotiuk. Topological persistence for medium access control. *IEEE Transactions on Mobile Computing*, 12(8), 2013.

[65] J. Lutz, C. J. Colbourn, and V. R. Syrotiuk. Variable-weight topology transparent scheduling. *under review*, 2013.

[66] A. D. Myers, G. V. Záruba, and V. R. Syrotiuk. An adaptive generalized transmission protocol for ad hoc networks. *Mobile Networks and Applications*, 7:493–502, 2002.

[67] T. Nandagopal, T.-E. Kim, X. Gao, and V. Bharghavan. Achieving MAC layer fairness in wireless packet networks. In *Proceedings of the 6th ACM Annual International Conference on Mobile Computing and Networking (MOBICOM'00)*, pages 87–98, 2000.

[68] W. Navidi, T. Camp, and N. Bauer. Improving the accuracy of random waypoint simulations through steady-state initialization. In *Proceedings of the 15th International Conference on Modelling and Simulation (MS'04)*, 2004.

[69] The Network Simulator. http://nsnam.isi.edu/nsnam/index.php/Main_Page.

[70] The `ns` Manual. http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf.

[71] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Elsevier Inc., 2004.

[72] R. Ramanathan. A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks. In *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'97)*, pages 900–907, 1997.

[73] C. H. Rentel and T. Kunz. Reed-solomon and hermitian code-based scheduling protocols for wireless ad hoc networks. In *Proceedings of the Fourth International Conference on Ad Hoc Networks and Wireless (ADHOC-NOW'05)*, pages 221–234, 2005.

[74] C. H. Rentel and T. Kunz. Bounds and parameter optimization of medium access control coding for wireless ad hoc and sensor networks. *Ad Hoc Networks*, 10(1):128–143, 2012.

[75] I. Rhee, A. Warrier, M. Aia, J. Min, and M. L. Sichitiu. Z-MAC: A hybrid MAC for wireless sensor networks. *IEEE Transactions on Networking*, 16(3):511–524, 2008.

[76] I. Rhee, A. Warrier, J. Min, and L. Xu. DRAND: Distributed randomized TDMA scheduling for wireless ad-hoc networks. *IEEE Transactions on Mobile Computing*, 8(10):1384–1396, 2009.

[77] R. Rozovsky and P. R. Kumar. SEEDEX: A MAC protocol for ad hoc networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'01)*, pages 67–75, 2001.

[78] J. L. Sobrinho and A. S. Krishnakumar. Distributed multiple access procedures to provide voice communications over IEEE 802.11 wireless networks. In *Proceedings of the Global Telecommunications Conference, (GLOBECOM'96)*, pages 1689–1694, 1996.

[79] D. R. Stinson. *Combinatorial Designs: Construction and Analysis*. Springer-Verlag, 2004.

[80] Y.-S. Su, S.-L. Su, and J.-S. Li. Topology-transparent node activation scheduling schemes for multihop TDMA ad hoc networks. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'04)*, pages 68–73, 2004.

[81] Y.-S. Su, S.-L. Su, and J.-S. Li. Topology-independent link activation scheduling schemes for mobile CDMA ad hoc networks. *IEEE Transactions on Mobile Computing*, 7(5):599–616, 2008.

[82] Q. Sun, O. K. Li, and K. Leung. Adaptive topology-transparent distributed scheduling in wireless networks. In *Proceedings of the International Conference on Communications (ICC 2010)*, pages 1–5, 2010.

[83] V. R. Syrotiuk, C. J. Colbourn, and A. C. H. Ling. Topology-transparent scheduling in MANETs using orthogonal arrays. In *Proceedings of the DIAL-M/POMC Joint Workshop on Foundations of Mobile Computing*, pages 43–49, 2003.

[84] V. R. Syrotiuk, C. J. Colbourn, and S. Yellamraju. Rateless forward error correction for topology-transparent scheduling. *IEEE/ACM Transactions on Networking*, 16(2):464–472, 2008.

[85] F. Talucci, M. Gerla, and L. Fratta. MACA-BI (MACA by invitation) – a receiver oriented access protocol for wireless multihop networks. In *Proceedings of the IEEE 6th International Conference on Universal Personal Communications Record*, pages 913–917, 1997.

[86] A. S. Tanenbaum. *Computer Networks*. McGraw Hill, fourth edition, 2003.

[87] Z. Tang and J. J. Garcia-Luna-Aceves. A protocol for topology-dependent transmission scheduling in wireless networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference, (WCNC'99)*, pages 1333–1337, 1999.

[88] L. Tassiulas and S. Sarkar. Maxmin fair scheduling in wireless networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, pages 763–772, 2002.

[89] Y. Tian, K. Xu, and N. Ansari. TCP in wireless environments: Problems and solutions. *IEEE Communications Magazine*, 43:27–32, 2005.

[90] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli. Wireless MAC processors: Programming MAC protocols on commodity hardware. In *Proceedings of the 31st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'12)*, pages 1269–1277, 2012.

[91] X. Wang and K. Kar. Distributed algorithms for max-min fair rate allocation in ALOHA networks. In *Proceedings of the 42nd Annual Allerton Conference on Communication, Control, and Computing*, 2004.

[92] X. Wang, K. Kar, and J.-S. Pang. Lexicographic max-min fairness in a wireless ad hoc network with random access. In *Proceedings of the 45th IEEE Conference on Decision and Control (CDC'06)*, pages 1284–1300, 2006.

[93] M. Weiser. The computer of the 21st century. *Scientific American*, 265(3):94–104, 1991.

[94] D. B. West. *Introduction to Graph Theory*. Prentice Hall, second edition, 2001.

[95] G.-C. Yang. Variable-weight optical orthogonal codes for CDMA networks with multiple performance requirements. *IEEE Transactions on Communications*, 44(1):47–55, 1996.

[96] J.-H. Youn and B. Bose. A topology-independent transmission scheduling in multihop packet radio networks. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'01)*, pages 1918–1922, 2001.

[97] C. Zhu and M. S. Corson. An evolutionary-TDMA scheduling protocol for mobile ad hoc networks. Technical Report TR 1998-32, Institute for Systems Research, University of Maryland, 1998.

[98] C. Zhu and M. S. Corson. A five-phase reservation protocol (FPRP) for mobile ad hoc networks. *Wireless Networks*, 7(4):371–384, 2001.

Appendix A

SUPPLEMENTAL MATERIALS

A.1 The Persistence of IEEE 802.11

For scheduled p-persistence, we define the persistence of a node to be the probability with which it transmits during a single slot. Since IEEE 802.11 does not divide time into transmission slots, this definition is not directly applicable. Yet, under a slightly more general framework, it is possible to define persistence for both scheduled p-persistence and IEEE 802.11 in a way that allows for a meaningful comparison between protocols. Here, we present such a framework and use it to measure node persistence in simulations of both protocols.

The persistence value p defines the number of expected transmission attempts made during a fixed number of slots (a fundamental point made in [29]). For example, we can expect $100p$ transmission attempts within any 100 contiguous slots. From this point of view, persistence is

$$p = \frac{\text{\# Transmission Attempts}}{\text{\# Transmission Slots}}.$$

When network packets are of equal size and the slot length is appropriately selected to contain the transmission of a single packet, persistence also defines the percentage of time a node spends transmitting. In other words, persistence is

$$p = \frac{\text{Time Spent Transmitting}}{\text{Total Time}}. \tag{A.1}$$

It is then possible to consider the persistence of any MAC protocol, without the notion of a protocol "slot," including IEEE 802.11.

One further modification to the definition of persistence is required to avoid bias caused by nodes sitting idle with an empty transmission queue. Nodes in an idle state have nothing to send and consequently do not access the channel. Inactivity due to an empty transmission queue should not influence persistence measurements. In order to eliminate bias due to idle nodes, the Total Time in Equation A.1 is revised

to be the total time during which a packet is queued for transmission. *Persistence* is then equal to

$$p = \frac{\text{Time Spent Transmitting}}{\text{Total Time while not Idle}}. \tag{A.2}$$

Under the persistence of Equation A.1 or Equation A.2, the maximum achievable persistence of any protocol is less than one. The limitations of the physical layer alone, such as receive and transmit switching times, make 100% channel utilization impossible. Possibly more significant is the overhead of the MAC protocol itself. At a minimum, slotted protocols must oversize the length of each transmission slot to allow for network clock drift. The slot length may need to be increased further if the protocol is expected to sense the channel before transmitting. Even in IEEE 802.11 MAC, DCF Inter Frame Space (DIFS) and Short Inter Frame Space (SIFS) result in unused transmission capacity.

A.1.1 Challenges in Measuring Persistence

As with all metrics, measurements must be performed on single, well defined events. For persistence, the lowest level measurement is the *instantaneous persistence* of a single packet transmission defined as

$$p_{\text{inst}} = \frac{\text{Transmit Time}}{\text{MAC Latency} + \text{Transmit Time}}. \tag{A.3}$$

where Transmit Time is the time required to transmit the packet and MAC Latency is the time the packet spent queued for transmission at the MAC layer. In our simulations all packets are equally sized making the numerator of Equation A.3 a constant. All variation in the metric is driven by changes in the denominator. In order to average these ratios, one would normally employ a geometric mean. Because the numerator is a constant, we instead compute the average and standard deviation for the inverse persistence values using the computationally easier arithmetic mean.

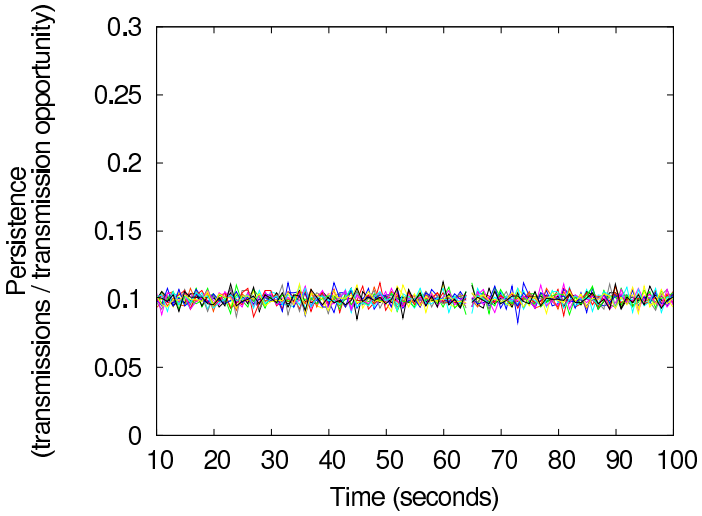
Average persistence is then taken to be the inverse of the average of the inverse persistence values.

A.1.2 Persistence Results

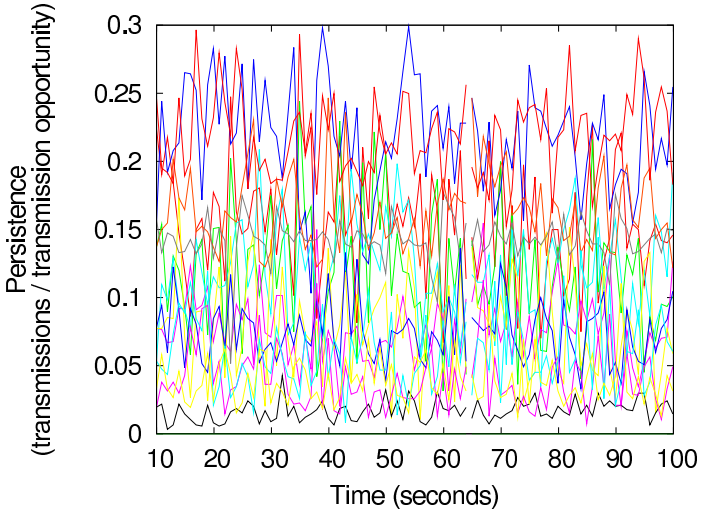
Traces of persistence in scheduled p-persistence and IEEE 802.11 are shown in Figure A.1a and Figure A.1b, respectively. Scheduled p-persistence is configured with $p = 0.1$ and a frame length of 20. The persistence data points in the traces are estimated by counting transmissions over 0.1 second intervals. As expected, the measured persistence of nodes running scheduled p-persistence remains stable and relatively unchanged while the persistence of IEEE 802.11 changes dramatically throughout the simulation. This does not come as a surprise since IEEE 802.11 is constantly adjusting the size of its contention window at each node.

Figure A.2a and Figure A.2b plot the average and standard deviation for the observed MAC latencies (magnitudes keyed on the left). Scheduled p-persistence is configured with a persistence of 0.1 and a frame length of 20. The error bars indicate the deviation from its expected value. Hence using Equation A.3 they also provide a general idea of how persistence varies as well. The bars in both figures give the average persistence associated with each network scenario (magnitudes keyed on the right). Figure A.2a demonstrates the accuracy of our measurements with nearly identical theoretical and measured persistence values. Figure A.2b agrees with the expectation that average persistence of IEEE 802.11 decreases as network load increases.

The method used to collect the persistence results in Figure A.1 and Figure A.2 has a significant weakness. The measurements only take into account persistence due to data packets. The control packets, such as RTS, CTS, and ACK in IEEE 802.11, along with the ACKs of scheduled p-persistence are ignored. This inaccuracy

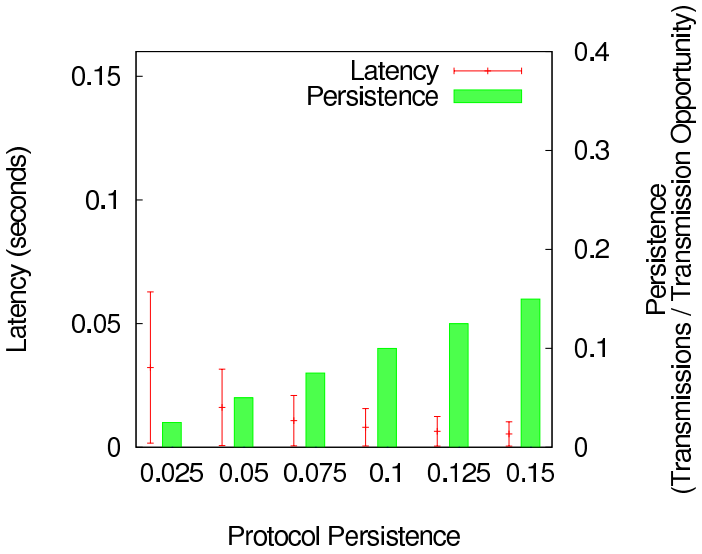


(a) Scheduled p-persistence with frame length of 20 and $p = 0.1$.

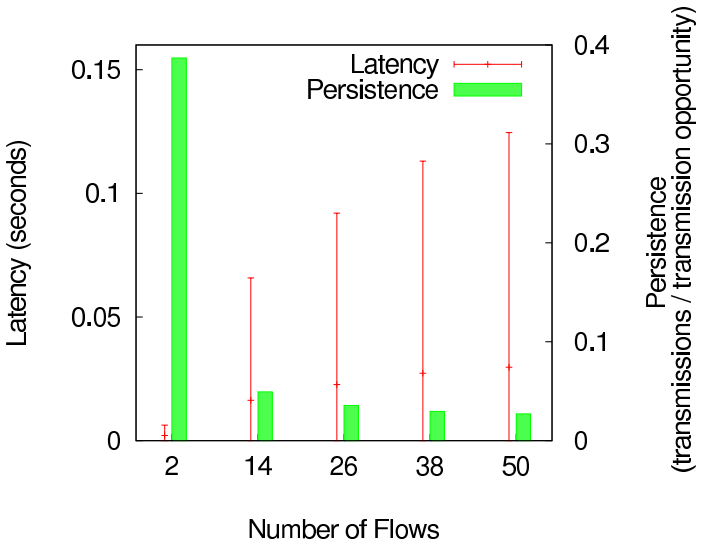


(b) IEEE 802.11.

Figure A.1: Persistence traces of IEEE 802.11 for five randomly selected nodes.



(a) Scheduled p-persistence with frame length of 20 and $p = 0.1$.



(b) IEEE 802.11.

Figure A.2: Average latency and persistence for IEEE 802.11.

limits the ability to make precise claims regarding the persistence of either protocol. Nevertheless, the current implementation is sufficient to demonstrate the volatility of IEEE 802.11 persistence.

A.2 Frame Length and Packet Delay

A.2.1 Scheduled **p**-Persistence Delay Variation

We define *delay* for a packet at the MAC layer to be the elapsed time from when the packet is queued for transmission at the source node's MAC layer to the time at which the packet is successfully received by the next hop destination's MAC layer. The structure imposed by the frame in scheduled (k, v)-persistence constrains the time between consecutive transmission attempts [29]. Indeed, the time between transmission slots is never larger than $v - k$ slots. In contrast, there is no worst case bound for pure p-persistence. In general, when $p = k/v$ is fixed and v increases, the maximum time that could arise between transmission slots also increases; p-persistence can be viewed as the limiting behaviour.

To demonstrate this, a series of `ns-2` simulations of a network running the scheduled p-persistence MAC protocol with a persistence of 0.1 for frame lengths of 10 (1 out of 10) and 100 (10 out of 100) slots is employed. Figure A.3 shows a scatter plot of delay for both frame lengths alongside delay from pure p-persistence. The distribution of expected delay is nearly identical regardless of MAC protocol or frame length. Variation in delay, on the other hand, increases with frame length. Because it has an effective frame length of infinity, the p-persistence MAC exhibits the highest variation in delay.

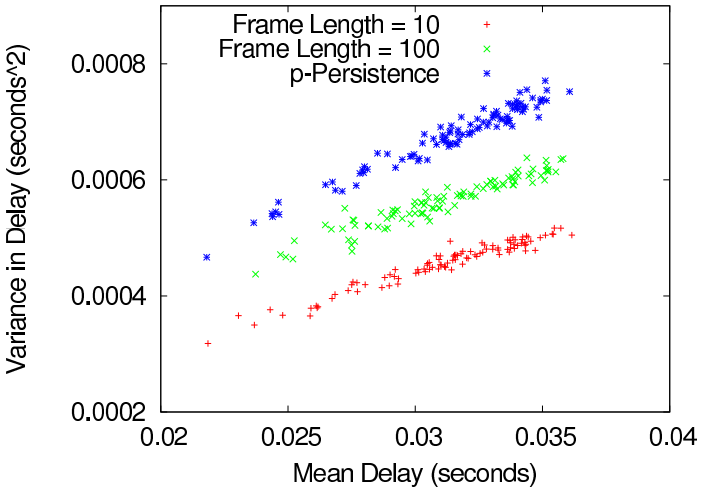
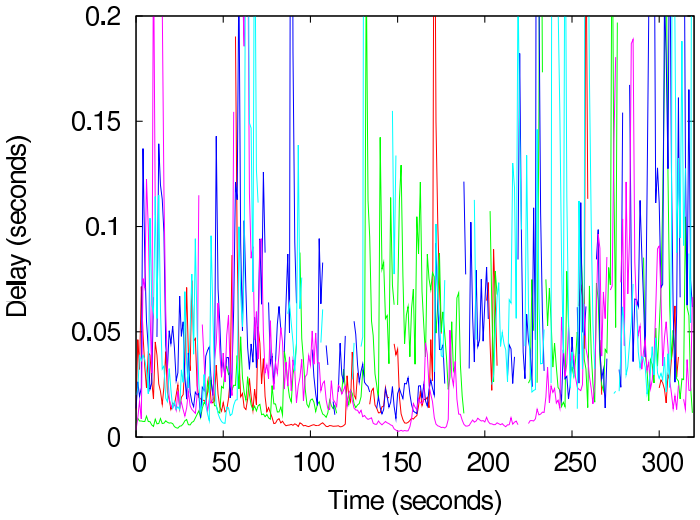


Figure A.3: Delays for scheduled p-persistence with frame lengths 10 and 100, and for 0.1-persistence. Expected delay is on the x-axis and variance in delay on the y-axis.

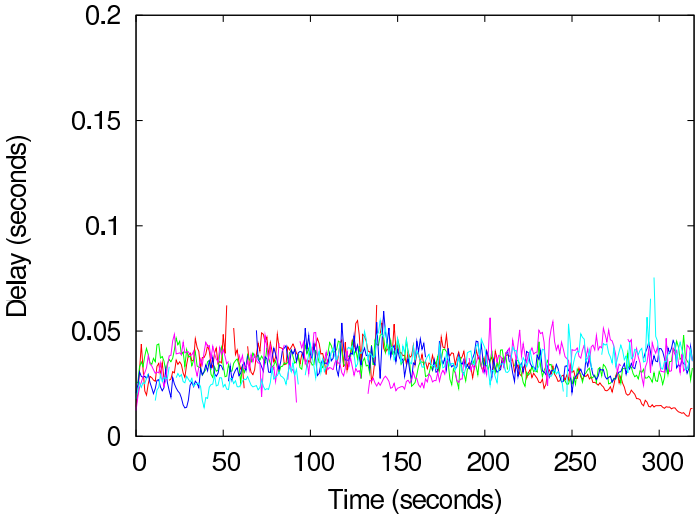
A.2.2 Instantaneous Delay Traces

One of the more striking differences between the slotted protocols and IEEE 802.11 is demonstrated by the delay traces shown in Figure A.4. Figure A.4a and Figure A.4b plot MAC delay as a function of simulation time for five randomly selected nodes from the 50 nodes in the network. For both simulations, the network was loaded with 50 traffic flows each generating 80 packets per second. From the perspective of the MAC layer, the flows appear to be elastic because the persistence levels assigned to the nodes are lower than the rate at which the CBR generators create packets. Scheduled p-persistence was run with $p = 0.1$ and a frame length of 20 slots.

The traces of Figure A.4a suggest that MAC layer delay in an IEEE 802.11 network varies widely over time and between nodes. In fact, the spikes in delay shown in Figure A.4a are truncated with the magnitudes of several approaching 1.6 seconds. This should not come as a surprise since nodes running the IEEE 802.11 MAC protocol adapt their back-off windows (adjusting the expected time between transmissions) in response to perceived network conditions. Nodes running scheduled p-persistence do



(a) IEEE 802.11.



(b) Scheduled p-persistence with frame length of 20 and $p = 0.1$.

Figure A.4: Instantaneous delay traces for five randomly selected nodes.

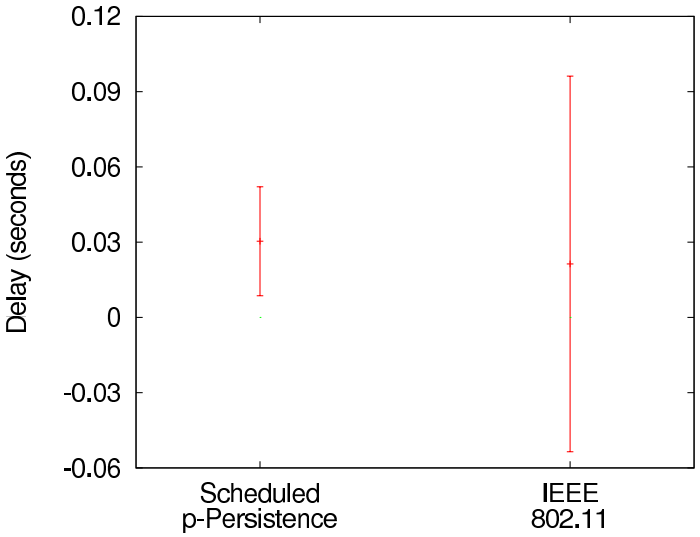


Figure A.5: System wide delay.

not adapt, but rather continue to transmit with the same persistence regardless of network topology or traffic congestion.

A.2.3 System Wide Delay

While the delay traces give a general indicator, they focus on the performance characteristics of a few specific points in the network and fail to describe system wide behaviour. Figure A.5 reports average MAC delay for both scheduled p-persistence ($p = 0.1$ and a frame length of 20) and IEEE 802.11. The error bars mark the standard deviation for the sample set containing all MAC layer delay times associated with successful MAC transmissions. While there is no significant difference between expected delays for the two protocols, the variation in delay (shown by the error bars) is much larger for IEEE 802.11.

A.2.4 MAC Layer Throughput

Another important metric is MAC layer throughput, defined as the rate (in packets per second) at which a node successfully transmits to its neighbours. Throughput is

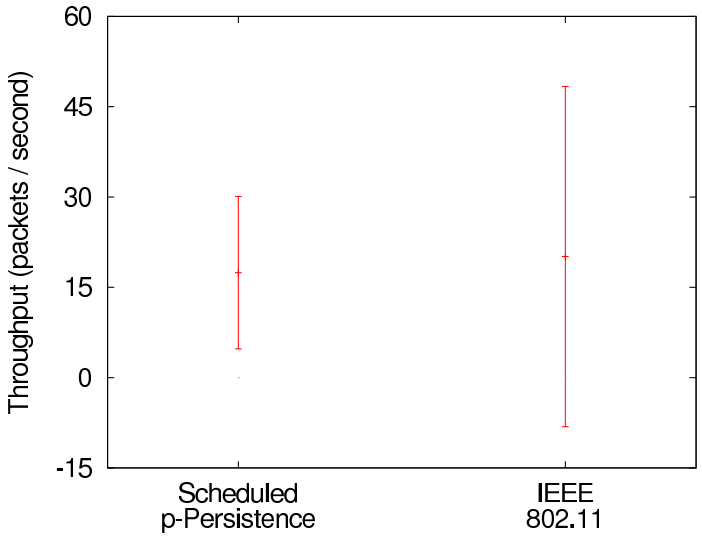


Figure A.6: System wide throughput.

measured at a given node by counting the successful transmissions performed by the node within a fixed period of time. In our simulations, we divide time into one second intervals over which throughput is computed for each node. These measurements are then used as the sample set for average and standard deviation computations. Figure A.6 shows system wide average and standard deviation in throughput for both scheduled p-persistence ($p = 0.1$ and a frame length of 20) and IEEE 802.11.

IEEE 802.11 shows a slightly higher average throughput but with much higher variation. Scheduled p-persistence appears to perform well in this particular scenario with competitive overall throughput and limited variation. However, one should not be misled by this single data point. The value $p = 0.1$ happens to be a reasonable choice for this specific network load. A different choice for p or a change in network load could significantly change the overall throughput achieved by the scheduled p-persistence protocol.

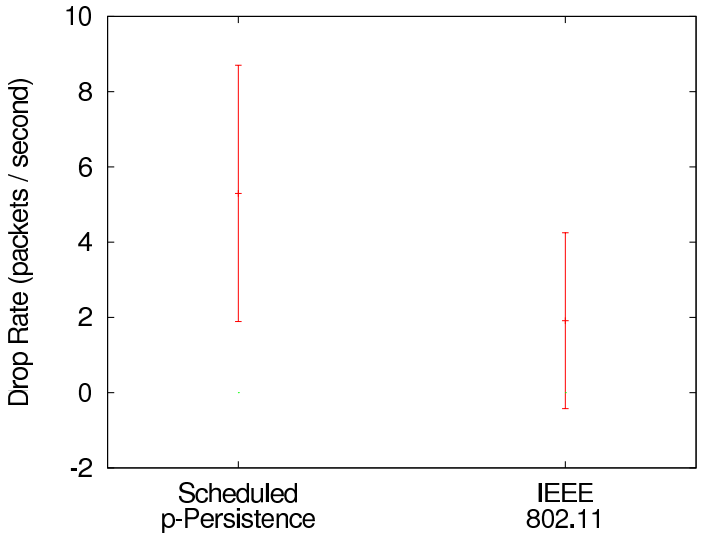


Figure A.7: System wide drop rate.

A.2.5 MAC Layer Drop Rate

The lower variation in throughput and delay achieved by scheduled p-persistence comes at the cost of increased drop rates; see Figure A.7. The rate at which packets are dropped by scheduled p-persistence is more than double that of IEEE 802.11. This is a direct consequence of adaptation in IEEE 802.11. The longer timeout period in IEEE 802.11 reduces the likelihood that a packet is dropped.

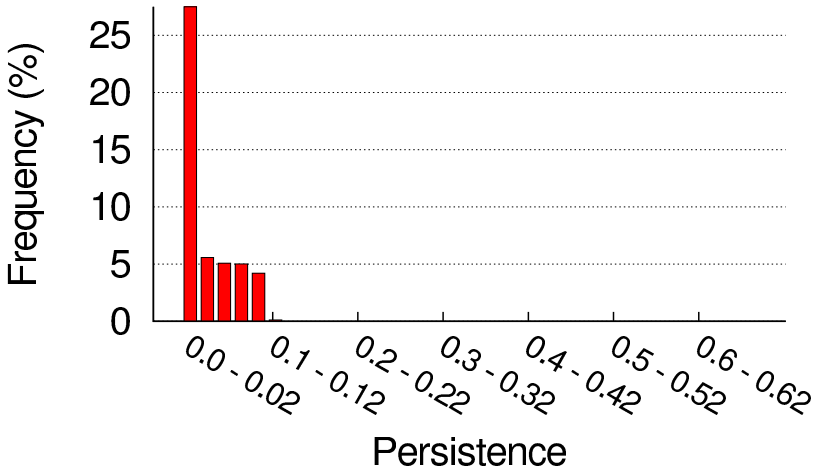
The delay results for scheduled p-persistence (Figure A.5) must be understood in the context of the higher drop rates in Figure A.7. Every dropped packet spends the maximum possible time queued for transmission before finally being dropped. Since the delay statistics do not include measurements associated with dropped messages, they can be misleading when treated in isolation. Drop statistics must always be considered in the proper context.

A.3 Analysis of Variable-Weight Schedules

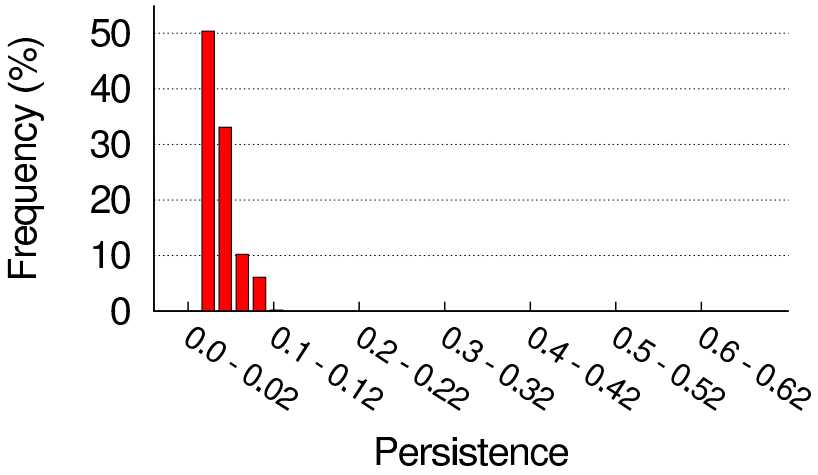
In order to determine what weights of schedules may perform well, we first investigate the distribution of persistences for the TLA allocation in a number of scenarios. Topologies are constructed by randomly placing 50 nodes, each with a transmission range of 250 meters, in a 1500 meter by 300 meter area for *dense* topologies and in a 4500 meter by 300 meter area for *sparse* topologies. *Few demands* means that 10 of the 50 nodes, chosen at random, have traffic to be carried on a single hop to a neighbour, again chosen at random; *many demands* means that all 50 nodes originate such a one-hop flow. *Small demands* arise when each node originating traffic desires a persistence selected uniformly at random from the range $[0.02, 0.10]$; *large demands* are instead selected uniformly at random from the range [0.60,0.68]. These reflect the rate at which the node would transmit in order to meet its demand, not the persistence at which it may be permitted to transmit. The eight resulting combinations reflect a wide variety of network topologies and traffic loads.

We undertook simulations using the network simulator `ns-2` on each of these eight combinations, using 100 randomly selected topologies for each, and computed the TLA allocation for persistence using the centralized method in Chapter 3. The bars in the histograms of Figure A.8 for dense topologies, and Figure A.9 for sparse topologies, identify the percentage of nodes with persistence in each range specified on the x axis.

Persistences less than 0.02 arise from nodes that originate no flow, and the corresponding bar in the histograms shown is truncated. All other nodes are assigned a persistence greater than 0.02. For the dense topologies, the mean persistence is 0.041, and the standard deviation is 0.053. For the sparse topologies, the mean persistence is

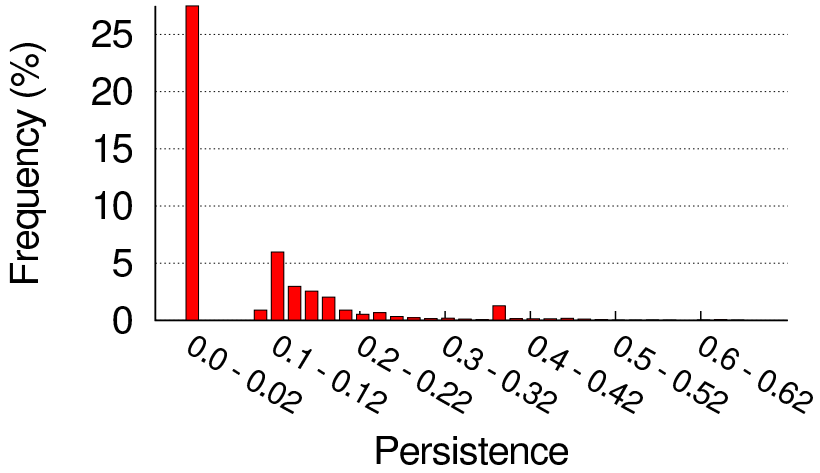


(a) A few small demands.

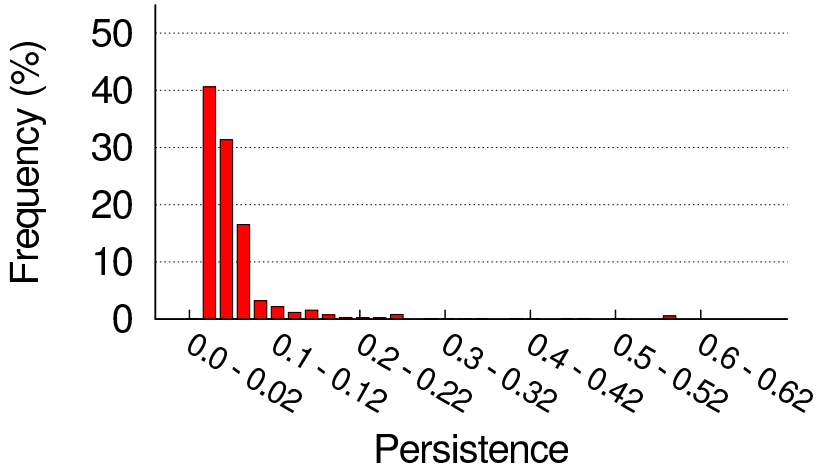


(b) Many small demands.

Figure A.8: Distribution of TLA persistences over 400 dense topologies with an average neighbourhood size of 18.7 nodes. This figure is continued on page 174.

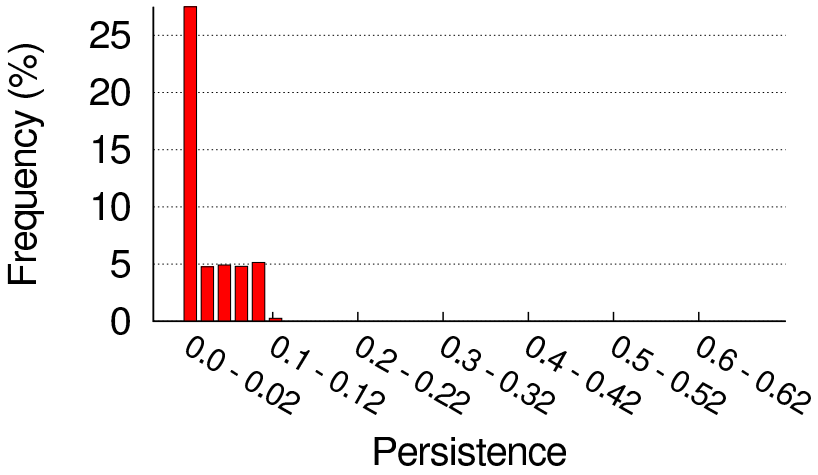


(c) A few large demands.

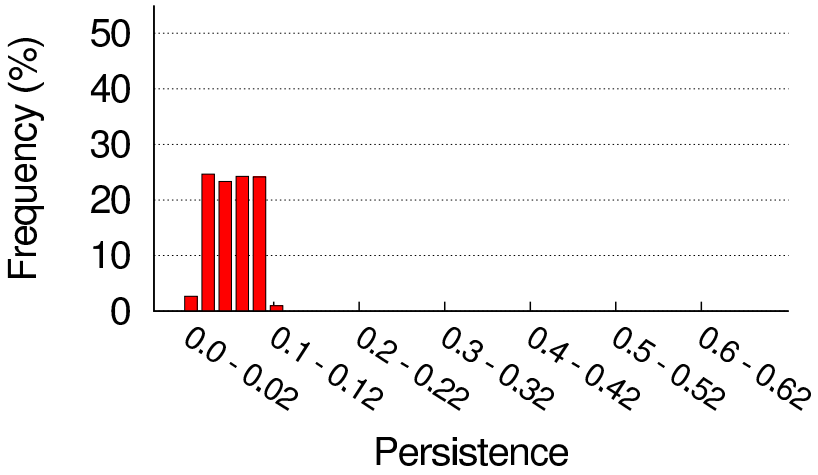


(d) Many large demands.

Figure A.8: Continued from page 173.

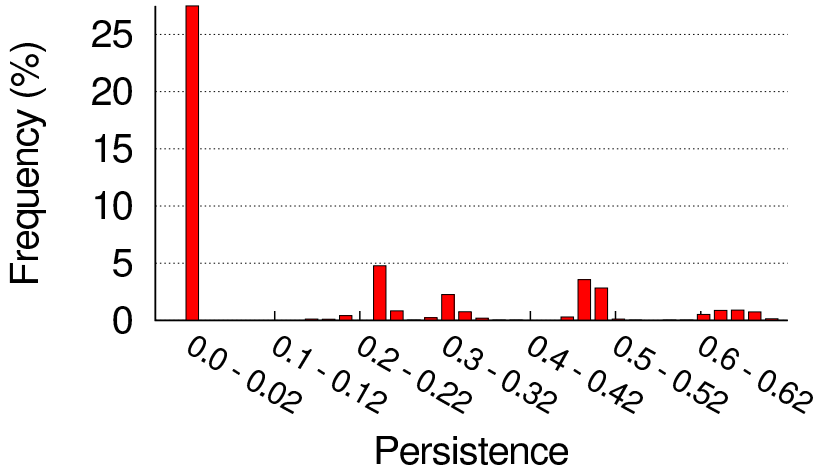


(a) A few small demands.

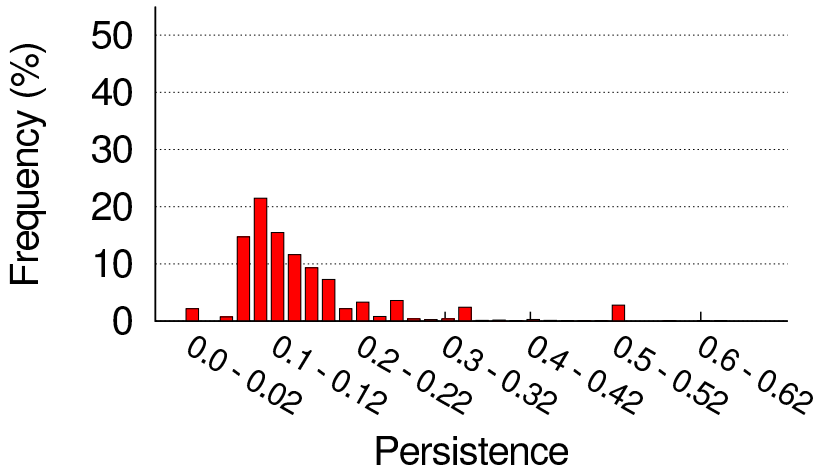


(b) Many small demands.

Figure A.9: Distribution of TLA persistences over 400 sparse topologies with an average neighbourhood size of 7.3 nodes. This figure is continued on page 176.



(c) A few large demands.



(d) Many large demands.

Figure A.9: Continued from page 175.

Table A.1: Sets of schedule weights for dense and sparse topologies.

	Dense $\mu = 0.04$ $\sigma = 0.05$	Sparse $\mu = 0.08$ $\sigma = 0.11$	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7
very small	0.01	0.01		√	√	√	√	√	√
$\mu - \frac{1}{2}\sigma$	0.02	0.03			√	√	√	√	√
μ	0.04	0.08	√	√	√	√	√	√	√
$\mu + \frac{1}{2}\sigma$	0.07	0.14					√	√	√
$\mu + \sigma$	0.09	0.19				√	√	√	√
$\mu + 2\sigma$	0.14	0.30						√	√
$\mu + 4\sigma$	0.24	0.51							√

higher, 0.076, and the standard deviation is 0.105. Most of the computed persistences are small. These distributions can be used to assist in the selection of suitable weights.

To demonstrate the effect of a variable-weight frame schedule, we simulate scheduled-vector persistence (see Section 2.4) which employs a vector $\mathbf{p} = (p_1, p_2, \ldots, p_N)$ of persistences, allowing each node to operate with a unique persistence. The TLA allocation is achieved by setting the vector of persistences equal to the TLA allocation. Similarly, a single weight is achieved by setting the elements of $\mathbf{p}$ to a constant. To accommodate a number of fixed weights, say $\{w_1, w_2, \ldots, w_m\}$, we simply select the nearest weight available to the one specified by the TLA persistence. Naturally, different mappings to the allowed weights are possible, but these suffice for our investigation.

Using the sets of weights in Table A.1, Figure A.10 and Figure A.11 show the effect of the number of weights employed in the schedule on throughput for the eight scenarios. For few large demands, the increase in the throughput is striking as the number of weights increases from one to seven. Schedules of variable weight appear

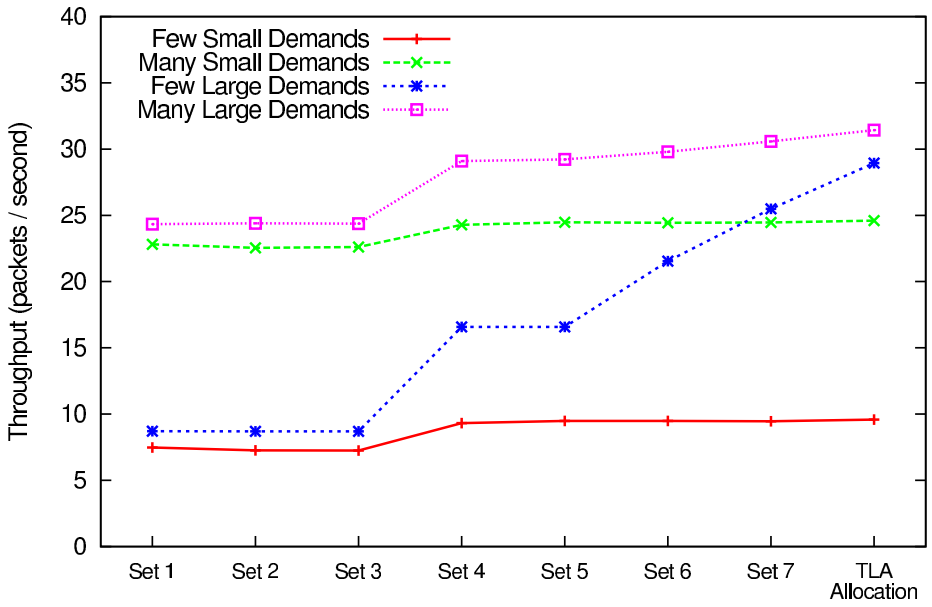


Figure A.10: Expected throughput in dense topologies when the persistences of the TLA allocation are mapped to a finite set of weights, shown in Table A.1.

to be useful because they have the potential to improve throughput, especially when the demands on the network are small. At first, it is disappointing that in this study, providing an additional weight may have no appreciable effect, as shown when moving from one to two weights, and from four to five weights. It remains unclear whether one needs many weights, or whether a few carefully chosen weights could suffice.

We therefore repeated the experiments with six pairs of weights: $\{\mu - \frac{1}{2}\sigma, \mu\}$, $\{\mu - \frac{1}{2}\sigma, \mu + \frac{1}{2}\sigma\}$, $\{\mu - \frac{1}{2}\sigma, \mu\}$, $\{\mu, \mu + \frac{1}{2}\sigma\}$, $\{\mu, \mu + \sigma\}$, and $\{\mu, \mu + 2\sigma\}$. (This selection is intended only to explore variation, not to find the very best.) We do not report complete results here, but rather summarize them. For each of the four load conditions on sparse networks, the pair $\{\mu, \mu + 2\sigma\}$ performs at least as well as the other five. Indeed for few large demands, employing only two weights, $\{\mu, \mu + 2\sigma\}$, yields a throughput of 50.5 packets per second. While still less than the 63.6 packets per second obtained with the TLA allocation, it is nonetheless a dramatic improvement on the 18 packets per second using the pair $\{\mu - \frac{1}{2}\sigma, \mu\}$. A similar pattern is observed for

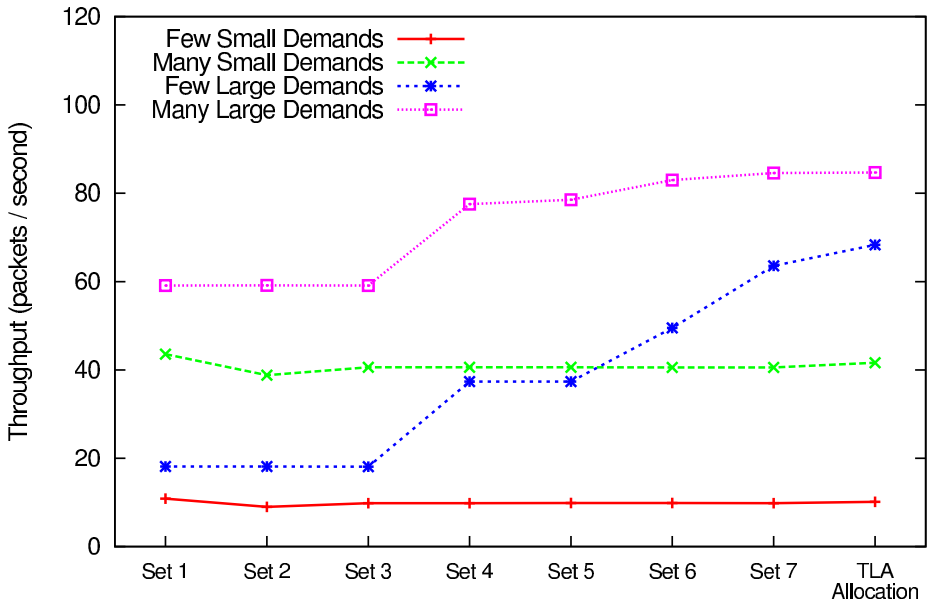


Figure A.11: Expected throughput in sparse topologies when the persistences of the TLA allocation are mapped to a finite set of weights, shown in Table A.1.

few large demands in dense networks: Again the pair $\{\mu, \mu + 2\sigma\}$ is the clear winner. For other load conditions in dense networks, the trends are less dramatic. While the pair $\{\mu - \frac{1}{2}\sigma, \mu\}$ performs the worst in each, the remaining pairs (the ones with at least one weight exceeding μ) yield very similar results, with the pair $\{\mu, \mu + \sigma\}$ having a slight advantage over the remainder.

A.4 Adding TDMA to the Variable-Weight Schedules of Chapter 5

TDMA can be incorporated into schedules derived from TD$(3, v, v)$s. For these designs, there are v^2 sets of schedules, $\mathbb{F}_1, \ldots, \mathbb{F}_{v^t}$, and a frame length of v^2 slots. A unique TDMA schedule, denoted $F_i^{(0)}$, can be added to each schedule set $\mathbb{F}_i$ for $1 \leq i \leq v^2$. The weight of each TDMA schedule is equal to one.

$$\text{wt}(F_i^{(0)}) = 1$$

While TDMA schedules maintain zero intersection with each other, they can intersect higher weight schedules in at most one position. Therefore,

$$\mathcal{I}\left(F_i^{(0)}\right) = \begin{cases} 0, & \text{in TDMA neighbourhoods,} \\ 1, & \text{otherwise.} \end{cases}$$

The TDMA schedules can maintain a guarantee on maximum delay in neighbourhoods as large as v^2 nodes provided that all nodes in the neighbourhood employ their TDMA schedules. The guarantee is lost if any node in the neighbourhood uses a higher weight schedule. In a multi-hop network, a node often operates in more than one neighbourhood concurrently and may be required to employ a TDMA schedule to maintain the delay guarantee in one neighbourhood and a higher weight schedule to maintain the delay guarantee in another. To maintain the delay guarantee in both neighbourhoods, a node can alternate between $\mathbb{F}_i^{(0)}$ and $\mathbb{F}_i^{(1)}$. Alternation between schedules must be synchronized across the network.

Use of TDMA comes at a cost to maximum delay, expected delay, and expected throughput. Maximum delay increases to the length of two frames (due to the alternation between schedules). In large networks, the small persistence of the TDMA schedules can significantly degrade expected throughput and delay. However, a delay guarantee is still maintained which may allow the network to resolve its congestion problem through topology/admission control to eventually eliminate the need to use TDMA.

Appendix B

LIST OF PUBLICATIONS

B.1 Journal Publications

1. **J. Lutz**, C. J. Colbourn, and V. R. Syrotiuk, "Topological persistence for medium access control", *IEEE Transactions on Mobile Computing*, vol. 12 no. 8, pp. 1598–1612, 2013.

2. **J. Lutz**, C. J. Colbourn, and V. R. Syrotiuk, "ATLAS: Adaptive topology- and load-aware scheduling", under review.

B.2 Conference Publications

1. **J. Lutz**, C. J. Colbourn, and V. R. Syrotiuk, "Apples and oranges: Comparing schedule- and contention-based medium access control", In *Proceedings of the 13th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'10)*, 2010, pp. 319–326.

2. J. Lutz, **C. J. Colbourn**, and V. R. Syrotiuk, "Variable weight sequences for adaptive scheduled access in MANETs", In *Proceedings of Sequences and their Applications (SETA'12)*, 2012, pp. 53–64 (invited paper).

3. **J. Lutz**, C. J. Colbourn, and V. R. Syrotiuk, "Variable-weight topology transparent scheduling", under review.