The Thesis Committee for Rajaganesh Ganesh
Certifies that this is the approved version of the following thesis:

SQ-CSMA: Universally Lowering the Delay of Queue-based CSMA/CA

APPROVED BY

SUPERVISING COMMITTEE:

Sujay Sanghavi, Supervisor

Constantine Caramanis

SQ-CSMA: Universally Lowering the Delay of Queue-based CSMA/CA

by

Rajaganesh Ganesh, B.E.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2010

Dedicated to my parents.

Acknowledgments

I wish to thank my advisor, Dr. Sujay Sanghavi without whom, I would not have been able to complete this work in a timely and efficient manner. I would also like to take this opportunity to thank my parents, for standing by me all the time and my friends for their constant encouragement.

SQ-CSMA: Universally Lowering the Delay of Queue-based CSMA/CA

Rajaganesh Ganesh, M.S.E.
The University of Texas at Austin, 2010

Supervisor: Sujay Sanghavi

Recent works show that, by incorporating queue length information, CSMA/CA multiple access protocols can achieve maximum throughput in general ad-hoc wireless networks. In all of these protocols, the aggressiveness with which a link attempts to grab the channel is governed solely by its own queue, and is independent of the queues of other interfering links. While this independence allows for minimal control signaling, it results in schedules that change very slowly. This causes starvation and delays - especially at moderate to high loads.

In this work we add a very small amount of signaling - an occasional few bits between interfering links. These bits allow us a new functionality: switching - a link can now turn off its interfering links with a certain probability. The challenge is ensuring maximum throughput and lower delay via the use of this new functionality. We develop a new protocol - Switch-enabled Queue-based CSMA (SQ-CSMA) - that uses switching to achieve both of these objectives.

This simple additional functionality, and our protocol to leverage it, can be "added on" to every existing CSMA/CA protocol that uses queue lengths. Interestingly, we see that in every case it has a significant positive impact on delay, universally furthering the performance of existing protocols.

Table of Contents

List of Figures

Chapter 1

Introduction

Distributed link scheduling is a central problem in ad-hoc wireless networks. Several popular approaches, and implemented protocols, are based on CSMA/CA, which combines channel sensing with collision avoidance via handshaking. Broadly speaking, links in these protocols continuously sense the channel, and contend for access when they have data to send. Until recently, the throughput of these systems for general interference networks was not known to be optimal. A recent set of results ([1],[2],[3]) shows that if queue lengths are used to modulate the aggressiveness of channel access by the links, then it is possible to make the protocols – which we will broadly call Q-CSMA - throughput-optimal. While these protocols are appealing due to their very low control overheads, they suffer from very high delays. Our main focus in this work is to reduce the long term average delay experienced by each node, while still maintaining the throughput optimality.

1.1 Previous Work

There are several existing centralized scheduling schemes which achieve throughput optimality and a lower delay. These protocols require a central governing node to monitor the queue lengths of all the nodes in the network and perform scheduling. For example, in the Maximal Weighted Scheduling (MWS) scheme, from the set of all non-interfering links, the set with maximum total queue length is chosen as the schedule in current time slot. MWS is known to be throughput optimal [4]. However, finding the maximal-weighted independent set in each time slot is NP-complete in general, and is hard even for centralized algorithms. The Greedy Maximal Scheduling, or, the Longest Queue First scheduling is a lower complexity alternative to maximal weight scheduling. This algorithm proceeds in a recursive manner by sequentially scheduling the link with the longest queue length while switching off all its interfering links. Though the GMS algorithm has a lower delay and is throughput optimal under certain constraints [5], for a general network topology, it achieves only a fraction of the capacity region [6],[7]. The signalling overhead also scales with the network size.

The CSMA (Carrier Sense Multiple Access) algorithms are a class of algorithms which are distributed random access algorithms. Under CSMA, a node (sender) will sense whether the channel is busy before it transmits a packet. When the node detects that the channel is busy, it will wait for a random amount of backoff time. Else, the node will transmit the packet. Since CSMA-type algorithms can be easily implemented in a distributed manner, they are widely used in practice in wireless networks. Under an idealized CSMA model (which assumes zero propagation/sensing delay and no hidden terminals) where collisions can never occur, it was shown that the Markov chain describing the evolution of schedules has a product-form stationary distribution [8].

Based on these results, Jiang and Warland [1] proposed a distributed algorithm to adaptively choose the CSMA parameters to meet the traffic demand, without explicitly knowing the arrival rates. The authors make a time-scale separation assumption, whereby the CSMA Markov chain converges to its steady-state product form distribution instantaneously when compared to the time-scale of adaptation of the CSMA parameters. They proved that this algorithm is throughput-optimal.

In [9], the authors study distributed algorithms for optical networks. In [2], a slightly modified version of the algorithm proposed in [9] was shown to be throughput-optimal. The key idea in [2] is to choose the link weights to be a specific function of the queue lengths to essentially separate the time scales of the link weights and the CSMA dynamics. The authors also assume that the maximum queue length in the network is known, through a distributed message-passing procedure.

Srikant and Ni [3] proposed a discrete-time version of the CSMA-type random access algorithm called the Q-CSMA which achieves the product-form distribution. In their work, given that the channel is free, the probability that a link grabs the channel in a given time slot depends solely on its own queue length.

1.2 Key Results

While the protocols discussed above are appealing due to their very low control overheads, they suffer from very high delays. As we will see, this is due to the fact that, at moderate loads, schedules change over very slow time scales. This causes starvation, which results in high delay. It would be preferable to have much lower delays at the cost of some additional control overhead. In this thesis, we take the very first step in this direction; we develop a protocol that adds a few bits of messaging between neighboring links. These additional bits can be used to enable links that have been starved (and hence have high queues) to pre-emptively switch off their interferers. This simple additional functionality, and our protocol to leverage it, can be "added on" to every existing CSMA/CA protocol that uses queue lengths. Interestingly, this makes Q-CSMA vary schedules much faster and we see that in every case it has a significant positive impact on delay, universally furthering the performance of existing protocols. The smartness in our design is finding a way to do this that still retains the throughput-optimality property. The signalling information is exchanged in a control slot and the schedule is transmitted in the transmission slot. Our protocol allows for collisions in the control slot and generates collision free transmission schedules in the data slot, thus relaxing the idealized CSMA condition.

Intellectually, our protocol borrows from the theory of faster-mixing Markov chains; in particular the mixing time of our scheduling algorithm captures how quickly the schedule changes with time. By performing switching in a way that leads to faster mixing, we retain throughput optimality while obtaining noticeable gains in delays. We introduce the network model and some nomenclature in the next chapter and the actual algorithm and results in the following chapters.

Chapter 2

Network Model

In this chapter, we present the network model and nomenclature that has been used throughout our work.

2.1 System Parameters

We model a single channel wireless network by an interference graph or a link contention graph, $G = (V, E)$ where V denotes the set of links and E denotes the edge set (Fig. 2.1). Each link consists of a transmitter-receiver pair. The edges model the radio interference between the links. Two links which are connected by an edge interfere with each other and hence cannot transmit simultaneously. For any link $i \in V$ we use $\mathcal{N}(i)$ to denote the set of all neighbors (conflicting links) of i. If at least one link in the set $\mathcal{N}(i)$ is active, the link i cannot be active. We assume symmetry in the conflict set so that if $j \in \mathcal{N}(i)$, then $i \in \mathcal{N}(j)$.

We assume a time slotted system. A *feasible schedule* is any schedule in which no two interfering links are active. Thus, a feasible schedule is always an independent set in the graph G. A feasible schedule is transmitted in every time slot t. Let $\mathcal{M}$ be the set of all feasible schedules in a network. A schedule

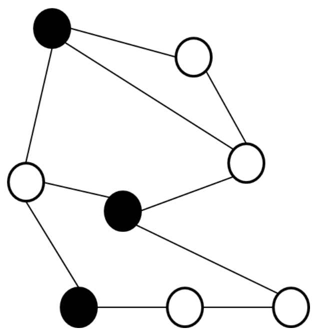

Figure 2.1: Link Contention Graph - Dark nodes represent active links

is represented by a vector $\mathbf{x} \in \{0,1\}^{|V|}$, where the i^{th} element of $\mathbf{x} = 1$ if link i is active in the current schedule; else $x_i = 0$. With a slight abuse of notation, we also treat $\mathbf{x}$ as a set and write $i \in \mathbf{x}$ if $x_i = 1$. Note that, for a feasible schedule,

$$x_i + \sum_{j \in \mathcal{N}(i)} x_j \leq 1 \ \forall i \in V$$

2.2 Throughput Optimality

A scheduling algorithm is an algorithm in which a feasible schedule is determined for transmission in each time slot. In this work, we propose a distributed scheduling algorithm. The capacity region of the network is the set of all arrival rates λ for which there exists a scheduling algorithm that can stabilize the queues, i.e., the queues are bounded in some appropriate stochastic sense depending on the arrival model used. For the purposes of this thesis, we will assume that if the arrival process is stochastic, then the resulting queue length process represents a Markov chain, in which case, stability refers to the positive recurrence of this chain. It is known (e.g., [4],) that the capacity region is given by

$$\mathbf{\Lambda} = \{\lambda \mid \exists \mu \in Co(\mathcal{M}), \mu > \lambda\}$$

where $Co(\mathcal{M})$ is the convex hull of the set of feasible schedules in $\mathcal{M}$. When dealing with vectors, inequalities are interpreted component-wise.

We say that a scheduling algorithm is throughput-optimal, or achieves the maximum throughput, if it can keep the network stable for all arrival rates in $\mathbf{\Lambda}$.

Chapter 3

The Algorithm

We propose the actual algorithm in this chapter and some motivation behind it. The analysis of the algorithm and implementation issues will be address in subsequent chapters.

3.1 Preliminaries

As stated before, we assume a discrete-time slotted system. We divide each time slot into a control slot and a transmission slot. The purpose of the control slot is to determine a collision-free schedule $\mathbf{x}(t)$ to be transmitted in the transmission slot. In order to achieve this, we first select a set of non-interefering links in the control slot, $\mathbf{m}(t)$ in a distributed manner. This set of links $\mathbf{m}(t)$ is called the *decision schedule* in the time slot t. Though the set $\mathbf{m}(t)$ is a feasible schedule, it is not the actual transmission schedule. It is the set of links that are chosen as candidates for update.

Let $\mathcal{M}_0 \subseteq \mathcal{M}$ denote the set of all decision schedules from which $\mathbf{m}(t)$ is chosen and let $\alpha(\mathbf{m}(t))$ be the probability distribution associated with $\mathcal{M}_0$ such that the sum $\sum_{\mathbf{m}(t)\in\mathcal{M}_0} \alpha(\mathbf{m}(t)) = 1$. In [3] the authors propose a distributed algorithm for choosing the transmission schedule in each time slot, using the decision schedule chosen in the control slot. For each link $i \in \mathbf{m}(t)$, if none of the neighbors of i are active in the previous time-slot, then link i becomes active in the current time-slot with probability p_i. With probability $1 - p_i$, the link is made inactive. If at least one of the neighbors of the link i is active in the previous time-slot, then, link i remains inactive with probability 1, in order to avoid interference. Intuitively, we can see that the link-activation probability has to be a function of queue lengths for the queues to be stable. In other words, larger the queue-length of a particular link, higher must be its probability to become active in that time slot. The exact expressions of how the link-activation probability depends on queue-length are specified later.

Though the proposed algorithm in [3] is throughput-optimal, the delays encountered are quite high. Whenever a link becomes active, it remains active for long periods of time and thus results in slower schedule changes and consequently, longer delays. For example, consider the scenario shown in Fig. 3.1. Here, the queue length of link D keeps on increasing until it can become active and once it becomes active, it remains active for a long period of time due to its large queue length.

The algorithm proposed in [3] is an MCMC sampler where the random independent updates converge to the desired product form distribution [8]. The intuition behind the larger delays is the slower schedule changes and hence the slower mixing time of the Markov chain. Thus, a faster mixing MCMC sampler must lead to lower delays.

Fig. 3.2 shows how the introduction of switching between links can

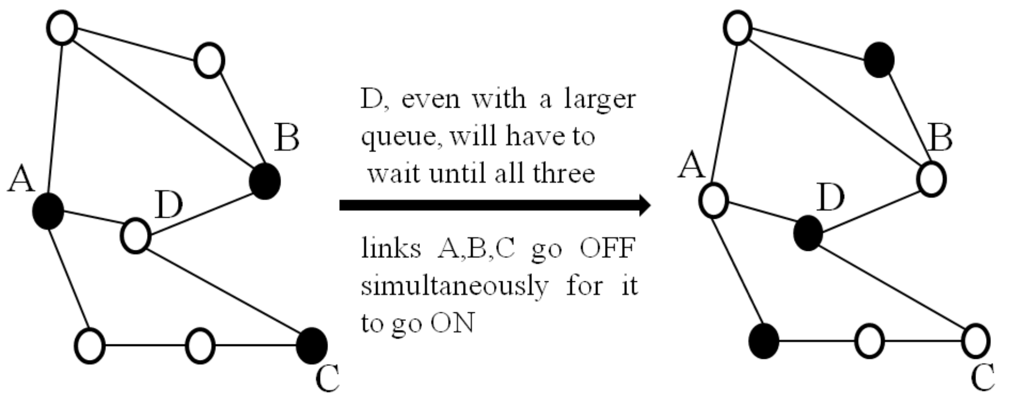
A
B
C
D
D, even with a larger queue, will have to wait until all three
links A,B,C go OFF simultaneously for it to go ON
A
B
C
D

Figure 3.1: Example of starvation due to slower schedule changes in Q-CSMA. D cannot become active until all its interferers are selected in the decision schedule and are made inactive.

lead to a faster schedule change for the same scenario seen in Fig. 3.1. This gives the idea behind the motivation for our algorithm. In our algorithm, we add an extra *switching* step to the algorithm proposed in [3] in order to obtain a faster mixing. In the original algorithm, whenever at least one of the neighbors of the selected link $i \in \mathbf{m}(t)$ is active in the previous time slot, the link i remains inactive with probability 1. In our algorithm, whenever there is a *unique interferer* to i, say $j \in \mathcal{N}(i)$, we associate a *switching probability* with which i can switch off j and become active. Unique interferer of a link i is the single unique neighbor of i who is currently active. This probability of switching must be chosen such that the Markov chain converges to the steady state distribution proposed in [8]. We select the switching probability to be $p_i\bar{p}_j$ where $i \in \mathbf{m}(t)$ switches off its unique interferer $j \in \mathcal{N}(i)$ and turns itself

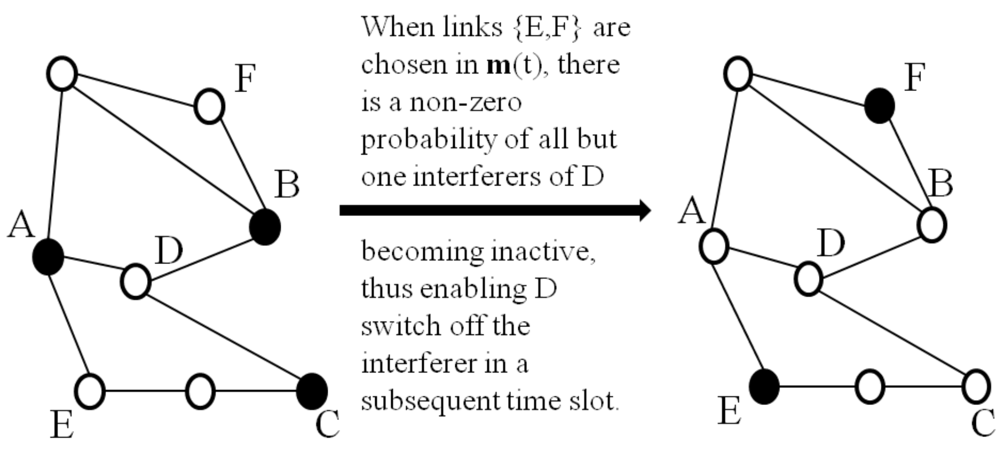
When links {E,F} are chosen in **m**(t), there is a non-zero probability of all but one interferers of D
becoming inactive, thus enabling D switch off the interferer in a subsequent time slot.
A
B
C
D
E
F
A
B
C
D
E
F

Figure 3.2: Links E and F switch off their interferers A and B respectively, leaving C as the only interferer for D, thus enabling D to switch OFF C and become active in a subsequent time slot.

ON.

3.2 SQ-CSMA Algorithm

We provide the basic algorithm here. The actual distributed implementation of the SQ-CSMA protocol is postponed until Chapter 5.

Basic Scheduling Algorithm (in Time Slot t)

1. In the control slot, randomly select a decision schedule $\mathbf{m}(t) \in \mathcal{M}_0$ with probability $\alpha(\mathbf{m}(t))$.

```
    $\forall i \notin \mathbf{m}(t)$:
        (a) $x_i(t) = x_i(t-1)$.
    $\forall i \in \mathbf{m}(t)$:
    If no links in $\mathcal{N}(i)$ were active in the previous data
    slot, i.e., $\sum_{j \in \mathcal{N}(i)} x_j(t-1) = 0$, then
        (b) $x_i(t) = 1$ with probability $p_i, 0 < p_i < 1$;
        (c) $x_i(t) = 0$ with probability $\bar{p}_i = 1 - p_i$.
    If only one link in $\mathcal{N}(i)$ was active in the previous data
    slot, say $j \in \mathcal{N}(i)$ and also, $j$ is not a unique interferer
    for any other link in $\mathbf{m}(t)$, then
        (d) $x_i(t) = 1$ and $x_j(t) = 0$ with probability
        $p_i\bar{p}_j, 0 < p_i, p_j < 1$.
        (e) $x_i(t) = x_i(t-1)$ and $x_j(t) = x_j(t-1)$ with
        probability $1 - p_i\bar{p}_j, 0 < p_i, p_j < 1$.
    Else
        (f) $x_i(t) = 0$.
2. In the transmission slot, use $\mathbf{x}(t)$ as the transmission schedule.
```

Chapter 4

SQCSMA - Analysis

In this chapter, we prove some important lemmas and propositions to show that the proposed algorithm is indeed throughput optimal.

4.1 A Discrete Time Markov Chain

One important condition to be satisfied in order to avoid collisions would be to ensure that the transmission schedule selected in each time slot is a feasible transmission scedule. In this section, we show that if previous transmission schedule and current decision schedule are both feasible, then, current transmission schedule is also feasible.

Lemma 4.1.1. *If* $\mathbf{x}(t-1) \in \mathcal{M}$ *and* $\mathbf{m}(t) \in \mathcal{M}_0 \subseteq \mathcal{M}$*, then,* $\mathbf{x}(t) \in \mathcal{M}$*.*

Proof. We know that, $\mathbf{x}(t) \in \mathcal{M}$ iff $\forall i \in V$ such that $x_i(t) = 1$, we have $x_j(t) = 0 \ \forall j \in \mathcal{N}(i)$.

Now consider any $i \in V$ such that $x_i(t) = 1$.

case(i): If $i \notin \mathbf{m}(t)$, then we know

(a) $x_i(t-1) = x_i(t) = 1$ based on Step (a). Since $\mathbf{x}(t-1) \in \mathcal{M}$, we have $\forall j \in \mathcal{N}(i), x_j(t-1) = 0$.

- If $j \notin \mathbf{m}(t)$, then $x_j(t) = x_j(t-1) = 0$ based on Step (a) of the scheduling algorithm above;
- If $j \in \mathbf{m}(t)$, then since $i \in \mathcal{N}(j)$ and $x_i(t-1) = 1, x_j(t) = 0$ based on Step (f). Or,

(b) $j \in \mathcal{N}(i) \cap \mathbf{m}(t)$ and $\sum_{k \in \mathcal{N}(j)} x_k(t-1) = 1$ based on step (d) of the scheduling algorithm. i.e., $k = i$ is the only active neighbor of a link j selected in $\mathbf{m}(t)$, and by step (e) of the scheduling algorithm, $x_i(t) = 1; x_j(t) = 0$ and hence, $\forall j \in \mathcal{N}(i)$, $x_j(t) = 0$.

case(ii): If $i \in \mathbf{m}(t)$, from the scheduling algorithm we have $x_i(t) = 1$ if

(a) $x_j(t-1) = 0 \ \forall j \in \mathcal{N}(i)$ based on step (b) of the scheduling algorithm. Since $i \in \mathbf{m}(t)$ and $\mathbf{m}(t)$ is feasible, we know $\mathcal{N}(i) \cap m = \emptyset$. i.e., $j \notin \mathbf{m}(t)$. Therefore, for any $j \in \mathcal{N}(i)$, $x_j(t) = x_j(t-1) = 0$ based on Step (a). Or,

(b) $\sum_{j \in \mathcal{N}(i)} x_j(t-1) = 1$ based on step (d) of the scheduling algorithm. In this case, only one of the neighbors of link i is active in the previous time slot, say $\{j \in \mathcal{N}(i) \mid x_j(t-1) = 1\}$ and by step (d) of the scheduling algorithm, $x_j(t) = 0$ and hence, $\forall j \in \mathcal{N}(i)$, $x_j(t) = 0$.

$\square$

Thus, we see that $\mathbf{x}(t)$ depends only on the previous state $\mathbf{x}(t-1)$ and some randomly selected decision schedule $\mathbf{m}(t)$. Hence, we can say that $\mathbf{x}(t)$ evolves as a discrete-time Markov chain (DTMC) on state space $\mathcal{M}$.

4.2 State Transition Probability

In this section, we will derive the transition probabilities between the states (transmission schedules) in the Markov chain desxribed in the previous section.

Proposition 4.2.1. *For a randomly selected decision schedule* $\mathbf{m}(t) \in \mathcal{M}_0$ *such that* $\cup_{m \in \mathcal{M}_0} m = V$*, the transition probability from* x *to* x' *is given by:*

$$P(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{m} \in \mathcal{M}_0} \alpha(\mathbf{m}) \left(\prod_{l \in \mathbf{x} \setminus \mathbf{x}'} \bar{p}_l \right) \left(\prod_{k \in \mathbf{x}' \setminus \mathbf{x}} p_k \right) \left(\prod_{i \in \mathbf{m} \cap (\mathbf{x} \cap \mathbf{x}')} p_i \right) \left(\prod_{j \in \mathbf{m} \setminus (\mathbf{x} \cup \mathbf{x}') \setminus \mathcal{N}(\mathbf{x} \cup \mathbf{x}')} \bar{p}_j \right) \left(\prod_{(a,b) \in (A,B)} (1 - p_a \bar{p}_b) \right) \tag{4.1}$$

where, the set (A, B) *is defined by the following equations:*

$$(A, B) = \{(a, b) \mid a \in \mathbf{m} \setminus (\mathbf{x} \cup \mathbf{x}'); \; b = \mathcal{N}(a) \cap (\mathbf{x} \cap \mathbf{x}'); |b| = 1\} \tag{2}$$

$$| A | = | B | \tag{3}$$

Proof. Given that $\mathbf{m} \in \mathcal{M}_0$ is the selected decision schedule in the current time slot, we can calculate the probability that the schedule makes a transition from $\mathbf{x}$ to $\mathbf{x}'$ by dividing into the following cases:

1. $k \in \mathbf{x}' \setminus \mathbf{x} \subseteq \mathbf{m}$: link k is seleceted in the decision schedule and it decides to change its state from 0 to 1. This occurs with probability p_k as given in Step (b) of the scheduling algorithm;

2. $l \in \mathbf{x} \setminus \mathbf{x}' \subseteq \mathbf{m}$: link l is selected in the decision schedule and it decides to change its state from 1 to 0. This occurs with probability $\bar{p}_l$ as given in Step (c) of the scheduling algorithm;

3. $i \in \mathbf{m} \cap (\mathbf{x} \cap \mathbf{x}')$: link i is selected in the decision schedule and it decides to keep its state 1, this occurs with probability p_i as given in Step (b) of the scheduling algorithm;

4. $j \in \mathbf{m} \setminus (\mathbf{x} \cup \mathbf{x}') \setminus \mathcal{N}(\mathbf{x})$: link j decides to keep its state 0, this occurs with probability $\bar{p}_j$ as given in Step (c) of the scheduling algorithm;

5. $e \in \mathbf{m} \cap \mathcal{N}(\mathbf{x})$ where $\mathcal{N}(\mathbf{x}) = \cup_{l \in \mathbf{x}} \mathcal{N}(l)$: link e has one or more interfering neighbors. If it has a unique interferer, it either switches states with the neighbor based on $case(6)$ or retains its state based on $case(7)$ given below. If it has more than one interfering neighbors, it has to keep its state 0, this occurs with probability 1 as given in Step (f) of the scheduling algorithm;

6. $k \in \mathbf{x}' \setminus \mathbf{x} \subseteq \mathbf{m}; l \in \mathcal{N}(k) \cap (\mathbf{x} \setminus \mathbf{x}' \setminus \mathbf{m})$: link k is seleceted in the decision schedule and it has a single interfering neighbor l. Link k switches off link l and turns itself ON. This happens with probability $p_k \bar{p}_l$ as given in Step (d) of the scheduling algorithm.

7. $(a,b) \in (A,B)$, where (A,B) is defined by (2), (3): link a is seleceted in the decision schedule and it has a single interfering neighbor b. Link a and link b retain their state. This happens with probability $(1 - p_a\bar{p}_b)$ as given in Step (d) of the scheduling algorithm.

Note that $\mathbf{m} \cap \mathcal{N}(\mathbf{x} \setminus \mathbf{x}') = \emptyset$ because $\mathbf{x}' \setminus \mathbf{x} \subseteq \mathbf{m}$, we have $\mathbf{m} \setminus (\mathbf{x} \cup \mathbf{x}') \setminus \mathcal{N}(\mathbf{x}) = \mathbf{m} \setminus (\mathbf{x} \cup \mathbf{x}') \setminus \mathcal{N}(\mathbf{x} \cup \mathbf{x}')$. Since each link in $\mathbf{m}$ makes its decision independently of each other, we can multiply these probabilities together. Summing over all possible decision schedules, we get the total transition probability from $\mathbf{x}$ to $\mathbf{x}'$ given in (4.1). $\square$

Remark 4.2.1. In (2), $a \in \mathbf{m} \setminus (\mathbf{x} \cup \mathbf{x}')$ ensures that link a which is inactive in previous slot is selected for switching. $b \in \mathcal{N}(a) \cap (\mathbf{x} \cap \mathbf{x}')$ ensures that link b is an active neighbor of a. $b \setminus \mathcal{N}(a) \cap (\mathbf{x} \cap \mathbf{x}') = \emptyset$ ensures that b is the only active neighbor of link a. Putting these two conditions together, we write it as $b \in \mathcal{N}(a) \cap (\mathbf{x} \cap \mathbf{x}')$ and $|b| = 1$. (3) ensures that link b is the unique interferer of a alone and not a unique interferer for any of its other neighbors. Note that the definition of the set (A,B) is symmetric in $\mathbf{x}$ and $\mathbf{x}'$. This will be useful in the analysis later.

4.3 Achieving the Product-form Distribution

As we explained earlier, the intuition behind the proposed protocol is to obtain a faster mixing MCMC but we must also make sure that we attain the product-form distribution stated in [8]. We have the following proposition.

Proposition 4.3.1. *The DTMC is irreversible and aperiodic if and only if* $\cup_{\mathbf{m}\in\mathcal{M}_0}\mathbf{m} = V$ *and in this case, the DTMC is reversible and has the following product-form stationary distribution:*

$$\pi(\mathbf{x}) = \frac{1}{Z}\prod_{i\in\mathbf{x}}\frac{p_i}{\bar{p}_i} \tag{3}$$
$$Z = \sum_{\mathbf{x}\in\mathcal{M}}\prod_{i\in\mathbf{x}}\frac{p_i}{\bar{p}_i}$$

Proof. The proof for the necessary and sufficient condition for the DTMC to be irreducible and aperiodic is similar to that of Proposition 1 from [3]. We will prove that for our protocol, the DTMC is also reversible. When $\mathbf{x}$ makes a transition to $\mathbf{x}'$, we want to show that distribution in (3) satisfies the following detailed balance equation:

$$\pi(\mathbf{x})P(\mathbf{x},\mathbf{x}') = \pi(\mathbf{x}')P(\mathbf{x}',\mathbf{x}) \tag{5}$$

From (3),

$$\begin{aligned}\frac{\pi(\mathbf{x})}{\pi(\mathbf{x}')} &= \frac{\prod_{i\in\mathbf{x}}\frac{p_i}{\bar{p}_i}}{\prod_{j\in\mathbf{x}'}\frac{p_j}{\bar{p}_j}} \\ &= \frac{\left(\prod_{i\in\mathbf{x}\setminus\mathbf{x}'}p_i\right)\left(\prod_{j\in\mathbf{x}'\setminus\mathbf{x}}\bar{p}_j\right)}{\left(\prod_{j\in\mathbf{x}'\setminus\mathbf{x}}p_j\right)\left(\prod_{i\in\mathbf{x}\setminus\mathbf{x}'}\bar{p}_i\right)}\end{aligned} \tag{6}$$

Referring back to the transition probability defined in (4.1), we can see that the first two product terms are independent of $\mathbf{m}$ and can be pulled out of the summation. Also, all the other terms except the first two product terms are symmetric in $(\mathbf{x},\mathbf{x}')$ and hence cancel with each other in $\frac{P(\mathbf{x}',\mathbf{x})}{P(\mathbf{x},\mathbf{x}')}$. Thus,

$$\frac{P(\mathbf{x}', \mathbf{x})}{P(\mathbf{x}, \mathbf{x}')} = \frac{\left(\prod_{k \in \mathbf{x}' \setminus \mathbf{x}} \bar{p}_k\right)\left(\prod_{l \in \mathbf{x} \setminus \mathbf{x}'} p_l\right)}{\left(\prod_{l \in \mathbf{x} \setminus \mathbf{x}'} \bar{p}_l\right)\left(\prod_{k \in \mathbf{x}' \setminus \mathbf{x}} p_k\right)} \tag{7}$$

From (6), (7), the DTMC is reversible and (3) is indeed its stationary distribution. $\square$

4.4 Throughput Optimality

Using the results obtained above, we finally are in a position to prove that the proposed algorithm is indeed throughput optimal. We impose some constraints on the link weight function and end this chapter with two important propositions.

Each link i has a non-negative *link weight* $w_i(t)$ associated with it, and the link activation probability p_i is a function of these link weights. These link weights, in turn, depend upon the queue length of the link. This was generalized in [10] as follows. For all $i \in V$, let link weight $w_i(t) = f_i(q_i(t))$, where $f_i : [0, \infty] \to [0, \infty]$ are functions that satisfy the following conditions:

1. $f_i(q_i)$ is a nondecreasing and continuous function with $\lim_{q_i \to \infty} f_i(q_i) = \infty$.

2. Given any $M_1 > 0, M_2 > 0$ and $0 < \epsilon < 1$, there exists a $Q < \infty$, such that for all $q_i > Q$ and $\forall\ i$, we have

$$(1 - \epsilon) f_i(q_i) \le f_i(q_i - M_1) \le f_i(q_i + M_2) \le (1 + \epsilon) f_i(q_i).$$

The very idea behind striving to achieve the product form distribution in (3) is, for the choice of $p_i = \frac{\exp w_i}{\exp w_i + 1}$, it was proved in [3] that the system is throughput optimal under the time scale separation assumption. We quote the following 2 propositions from [10] and [3] respectively, to assert that our scheduling algorithm is throughput optimal.

Proposition 4.4.1. *For a scheduling algorithm, given any ϵ and δ, $0 < \epsilon, \delta < 1$, there exists a $B > 0$ such that: in any time slot t, with probability greater than $1 - \delta$, the scheduling algorithm chooses a schedule $\mathbf{x}(t) \in \mathcal{M}$ that satisfies*

$$\sum_{i \in \mathbf{x}(t)} w_i(t) \geq (1 - \epsilon) \max_{\mathbf{x} \in \mathcal{M}} \sum_{i \in \mathbf{x}(t)} w_i(t)$$

whenever $||\mathbf{q}(t)|| > B$, where $\mathbf{q}(t) = (q_i(t) : i \in V)$. Then the scheduling algorithm is throughput-optimal.

Proposition 4.4.2. *Suppose the basic scheduling algorithm satisfies $\cup_{\mathbf{m} \in \mathcal{M}_0} \mathbf{m} = V$ and hence has the product-form stationary distribution. Let $p_i = \frac{\exp w_i(t)}{\exp w_i(t) + 1} \forall\, i \in V$. Then the scheduling algorithm is throughput-optimal.*

Chapter 5

Distributed Implementation

In this chapter, we present a distributed implementation of the basic scheduling algorithm proposed in chapter 3. The important thing to keep in mind is that we need to come up with a distributed algorithm to select a feasible schedule $m(t)$ and also to perform switching with as little signalling as possible. To achieve this, each time slot is divided into a control slot and a transmission slot. The control slot in turn is divided into a reserve slot and a switching slot (Fig. 5.1). The decision schedule $m(t)$ is determined in the reserve slot and switches between neighbors, if any, are handled in the switching slot. Interference information (i.e., whether or not a neighbor was active in the previous time slot) is available at each link via carrier sensing. We also avoid collisions in the transmission schedule by exchanging control messages. We call this algorithm **SQ-CSMA (Switch-enabled Queue-based CSMA)**.

5.1 Protocol for SQ-CSMA

We now describe the actual implementation of the SQ-CSMA protocol. The time slot structure in a SQ-CSMA protocol is shown in Fig. 5.1. At this

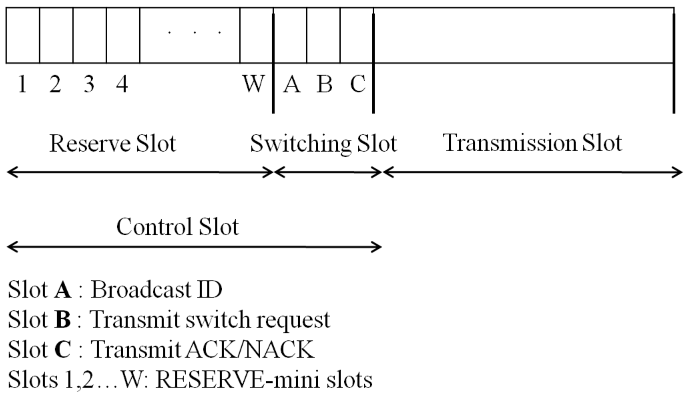

Figure 5.1: Time Slot structure in SQ-CSMA

point of time, we would advice the reader to refer back to the basic scheduling algorithm that was proposed in chapter 3 to correlate it with Fig. 5.1 and the SQ-CSMA protocol described below.

At the beginning of each control slot, every link i starts a clock with a random back-off time, T_i.

SQ-CSMA Algorithm (in Time Slot t at link i)

Reserve-Phase:

1. Link i selects a random (integer) back-off time T_i uniformly from the interval $[0, W-1]$ and waits for T_i reserve mini-slots. W is the number of reserve mini-slots.

2. If link i hears a RESERVE message from any of its neighbors $\mathcal{N}(i)$ before the $(T_i + 1)^{th}$ mini-slot, it deactivates its back-off timer and waits until the end of the reserve-phase (W mini-slots). It is not included in $\mathbf{m}(t)$.

3. If link i does not hear a RESERVE message from any of its neighbors $\mathcal{N}(i)$ before the T_i^{th} mini-slot, it sends out a RESERVE message to all its neighbors at the beginning of the $(T_i + 1)^{th}$ mini-slot.

   (a) If there is a collision (i.e., if there are one or more links in the neighbor set $\mathcal{N}(i)$ with the same back-off time as T_i which transmit a RESERVE message in the same reserve mini-slot as i), set $x_i(t) = x_i(t-1)$.

   (b) If there is no collision, link i is selected in the decision schedule and it becomes a candidate for update and it waits till the end of the reserve phase.

Switching-Phase:

At the end of the reserve phase, link i enters the *switching phase.* From Fig. 5.1,

Slot **A**: Links active in previous transmission slot broadcast their id to all the neighbors from which RESERVE message was received.
Slot **B**: Links selected in decision schedule which receive a single ID during time slot A, transmit a SWITCH REQUEST message to the sender.

Slot **C**: Links which receive a single switch request transmit back an ACK. Links which see a collision due to multiple switch requests transmit back a NACK to all their neighbors.

(a) $i \in m(t)$

In mini-slot A (Fig. 5.1), link i will receive the ID's of all its neighbors who were active in the previous transmission slot.

1. If no links in $\mathcal{N}(i)$ were active in the previous transmission slot,(i.e, if no ID is received in mini-slot A) the following scheduling update is made:
    - $x_i(t) = 1$ with probability p_i;
    - $x_i(t) = 0$ with probability $\bar{p}_i = 1 - p_i$.
2. If only one ID is received, say from $j \in \mathcal{N}(i)$, in mini-slot B (Fig. 5.1), link i sends a switch request to link j with probability $p_i\bar{p}_j, 0 < p_i, p_j < 1$.
    - If link i gets back an ACK message from link j in mini-slot C (Fig. 5.1), it becomes active with probability 1. i.e., $x_i(t) = 1$.
    - If link i gets back a NACK message from link j in mini-slot C, it remains inactive with probability 1. i.e., $x_i(t) = 0 = x_i(t-1)$.
3. If more than one link in $\mathcal{N}(i)$ was active in the previous data slot (i.e, if link i sees a collision in mini-slot A), $x_i(t) = 0$.

(b) $i \notin m(t)$

1. If link i was inactive in the previous transmission slot, it just retains its state. i.e., $x_i(t) = x_i(t-1) = 0$.

2. If link i was active in the previous time slot, it broadcasts its ID to all its neighbors who had sent a RESERVE message during the reserve slot. The ID is broadcasted in the mini-slot A (Fig. 5.1).

3. If link i a *unique interferer* for one of its neighbors $\mathcal{N}(i)$, it might hear a *switch request* from this neighbor during the slot B (Fig. 5.1).

4. If a single switch request is received in mini slot B, link i sends back an ACK message in the mini slot C (Fig. 5.1) to the neighbor who had sent a switch request and becomes inactive with probability 1. i.e., $x_i(t) = 0$.

5. If link i sees a collision due to multiple switch requests, it sends a NACK message to all its neighbors in the mini slot C and retains its previous state. i.e, $x_i(t) = x_i(t-1) = 1$.

6. If link i does not receive a switch request from any of its neighbors during the switching phase, it just retains its previous state. i.e., $x_i(t) = x_i(t-1)$.

At the end of the control slot, if $x_i(t) = 1$, link i will transmit a packet in the transmission slot.

Proposition 4.3.1 states that the DTMC is reversible and aperiodic if and only if $\cup_{\mathbf{m} \in \mathcal{M}_0} \mathbf{m} = V$. From [3], in order to ensure this condition, reserve-window length has to be at least 2, i.e., $W \geq 2$. Thus, from proposition 4.4.2, the SQ-CSMA protocol is throughput optimal for $W \geq 2$.

Remark 5.1.1. When describing the SQ-CSMA algorithm, we treat every link as an entity, while in reality each link consists of a sender node and a receiver node. Both carrier sensing and transmission of data/control packets are actually conducted by those nodes.

5.2 Other distributed scheduling protocols

Comparing our SQ-CSMA protocol with the Q-CSMA protocol proposed in [3], we can see that the only extra overhead involved in the implementation of SQ-CSMA is the signalling information sent during the 3 switching mini-slots. But, as we will be seeing from the simulations in chapter 6, this small extra signalling gives a considerable improvement in the delay performance of SQ-CSMA over Q-CSMA.

D-GMS[3] is a distributed implementation of the GMS protocol. Here, the control slot is divided into B frames and each frame, in turn, is divided into W mini-slots. The back-off time T_i is a random integer in the range $[0, BW - 1]$, where, the frame at which the timer expires is deterministically chosen, depending upon the link's queue length and the mini-slot within this frame, at which the timer expires is chosen uniformly at random. The links whose clock expire first are included in the schedule and all their neighbors are switched off. GMS, as we explained earlier, achieves lower delay in the low traffic region but performs badly in the moderate to high traffic region, for a general network topology.

We have also implemented a distributed algorithm to approximate maximal scheduling (called D-MS [3]), which can be viewed as a synchronized slotted version of the IEEE 802.11 DCF with the RTS/CTS mechanism. DMS is a special case of D-GMS where the number of frames $B = 1$. In other words, the transmission schedule in D-MS is just the decision schedule that we choose in the SQ-CSMA protocol.

Chapter 6

Simulations

In this chapter, we compare the performance of SQ-CSMA algorithm with the other distributed scheduling schemes mentioned in the previous chapter.

6.1 SQ-CSMA vs. Q-CSMA

Consider the link contention graph shown in Fig. 6.1.

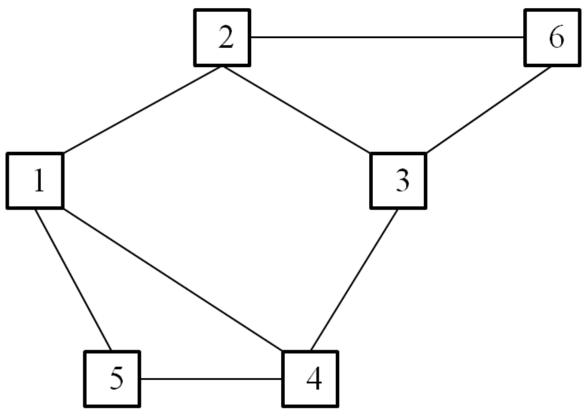

Figure 6.1: Link Contention Graph for a 6-link network

Each link is represented by a square and interference between links are represented by edges. Each link maintains its own queue and it only needs to know the set of links that conflict with itself. Consider the following set of non-interfering links:

$$L_1 = \{1, 6\}; L_2 = \{3, 5\}; L_3 = \{2, 4\}$$

Each set represents a maximal independent set of the network, $\mathbf{M_i} = \mathbf{e_{L_i}}$, where $\mathbf{e_{L_i}}$ is a vector in which the components with indices in L_i are 1 and others are 0. We define $0 < \rho < 1$ as the *traffic load factor*. Let the arrival rate vector be a convex combination of the maximal schedules scaled by ρ : $\lambda = \rho \times [0.2\mathbf{M_1} + 0.3\mathbf{M_2} + 0.2\mathbf{M_3} + 0.3\mathbf{M_4}]$. $\rho < 1$ ensures that the arrival rate vector is inside the capacity region with $\rho \to 1$ representing arrival rates approaching the boundary of the capacity region.

The packets arrive at each link i according to a Bernoulli process with parameter λ_i independent of arrival at the other links. The queue lengths are all initialized to zero at the beginning of each simulation. For each of the scheduling algorithm and for a fixed $rho < 1$, the simulation is run for 10^5 time slots. The system parameters for the different algorithms are given below:

- SQ-CSMA: $W = 48$; link weights $w_i(t) = log(1 + \beta q_i(t))$; $\beta = 1$; link activation probability $p_i(t) = \frac{exp(w_i(t))}{1+exp(w_i(t))}$; probability of link i switching off link $j = p_i(1 - p_j)$

- D-GMS: $B = 3$; $W = 16$; D-MS: $W = 48$

- Q-CSMA: $W = 48$; link weights $w_i(t) = log(1 + \alpha q_i(t))$; $\alpha = 0.1$; link activation probability $p_i(t) = \frac{exp(w_i(t))}{1+exp(w_i(t))}$

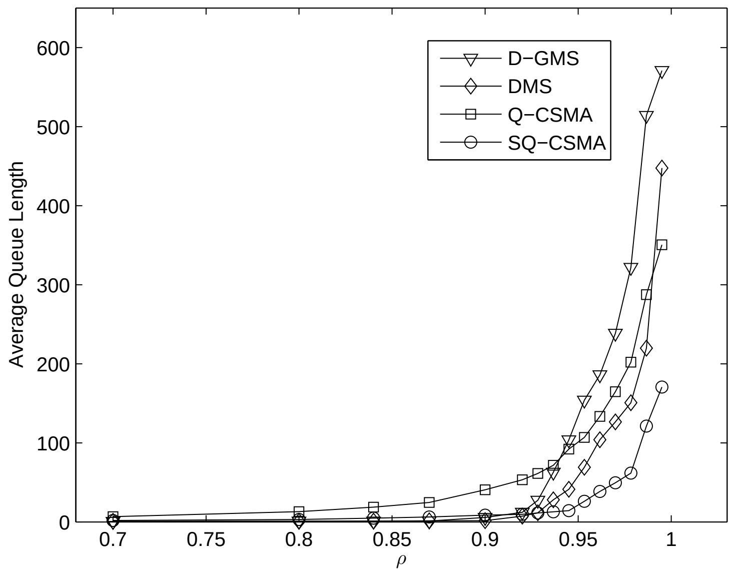

Figure 6.2: Long term average queue length per link for the network in 6.1

The rationality to choose $w_i(t) = log(1 + \beta q_i(t))$ as the link weight function is to make the link weights change much slower than the dynamics of the CSMA Markov chain (to satisfy the time-scale separation assumption) [3]. A couple of other link weight functions such as $w_i(t) = \beta q_i(t)$ [1] and $w_i(t) = log(log(\beta q_i(t)))$ [2] have been suggested in the literature but $w_i(t) = log(1+\beta q_i(t))$ with $\beta = 1$ seems to give the best performance for the SQ-CSMA protocol.

The simulation results for the link contention graph in Fig. 6.1 are shown in Fig. 6.2. By Little's law, the long-term average queueing delay experienced by the packets is proportional to the long-term average queue length in the network. Using this notion, we can conclude that:

- The long term average queue length of SQ-CSMA is better than that of Q-CSMA by almost a factor of 2, for all values of ρ. This fact is also emphasized from Fig. 6.3 which shows the evolution of average queue length per link as the time increases.
- Under low traffic intensity, D-GMS and D-MS have a lower long-term average delay when compared to both Q-CSMA and SQ-CSMA. But when the traffic intensity increases (which is the region of interest in many real world applications), there is a steep rise in the long term average queue lengths of both D-GMS and D-MS schemes delay performance become worse than Q-CSMA and SQ-CSMA.
- Though D-GMS and D-MS have good delay performance in the low-traffic region, they are not generally throughput optimal, as discussed in the next example.

Remark 6.1.1. Srikant and Ni came up with a modified version of Q-CSMA called the Hybrid Q-CSMA([3])which makes use of the fact that D-GMS has a very good delay performance in the low-traffic region. In Hybrid Q-CSMA, each link has a predefined threshold value for the queue length, below which D-GMS is employed. Once the queue length crosses this threshold, Q-CSMA is implemented. Thus, Hybrid Q-CSMA has a delay performance similar to D-GMS in the low-traffic region and Q-CSMA in the heavy-traffic region [3].

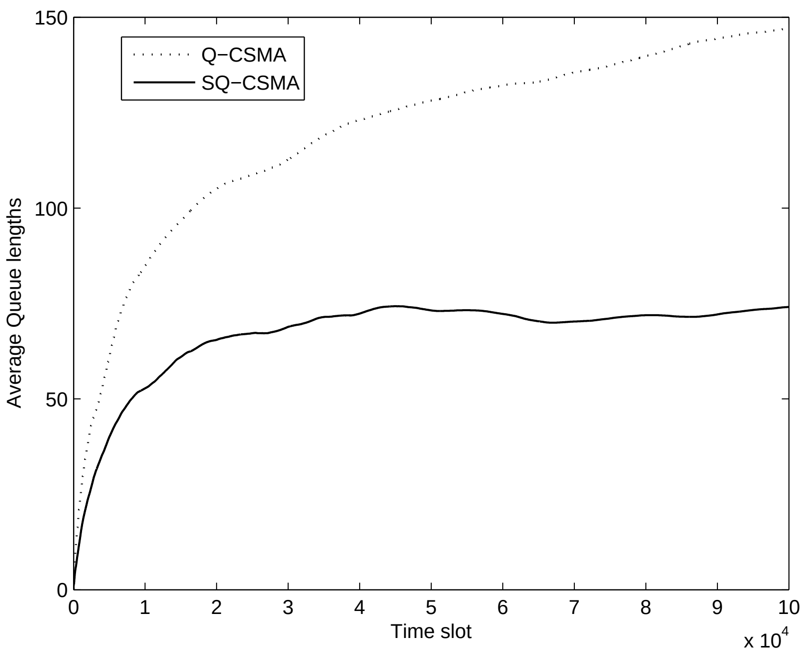

Figure 6.3: Evolution of average queue length per link

6.2 A 9-link ring network

We consider a 9-link ring network ([3]) under 2-hop interference model, as shown in Fig. 6.4. This graph is different from a link contention graph. Edges of this graph actually form the nodes of a link contention graph.

In [7], [3] the authors show that GMS for this network achieves only 2/3 of the capacity region for the traffic pattern defined as follows. Define $L_i = \{i, (i+4) \bmod 9\}, 1 \leq i \leq 9$. Starting with empty queues, in time slot $9k + i (k \in \mathbb{Z})$, one packet arrives at each of the 2 links in L_i, and, with probability ϵ, an additional packet arrives at each of the 9 links. The average

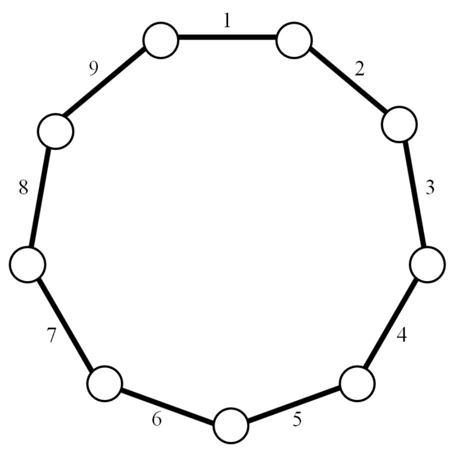

Figure 6.4: A 9-Link Ring Topology

arrival rate is then $\lambda = (\frac{2}{9} + \epsilon)$, for each link. When $0 < \epsilon < 1/9$, λ lies in the interior of the capacity region, but GMS cannot keep the network stable. This is emphasized in Fig. 6.5 where we evaluate the performance of GMS,D-MS, QCSMA and SQ-CSMA for the 9-link ring network with the traffic pattern defined as above. For $\epsilon = 0.09$, the long term average queue length increases linearly with time for both GMS and D-MS but it stabilizes for Q-CSMA and SQ-CSMA at a much lower value. Another fact to note is that the SQ-CSMA again has a better delay performance than Q-CSMA by a factor of 2.

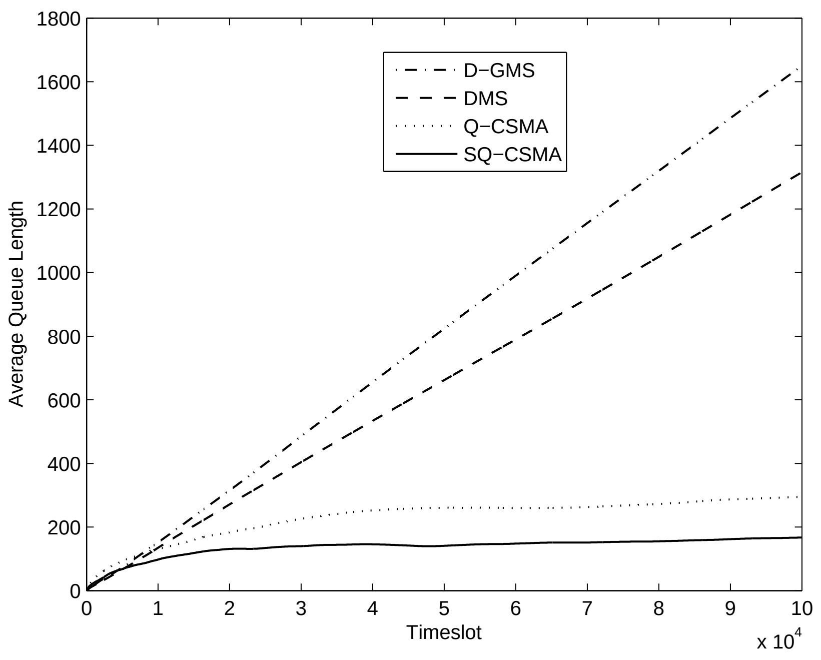

Figure 6.5: Evolution of average queue length per link for 9-link topology

Chapter 7

Conclusions

In this work, we presented a distributed scheduling algorithm which is both throughput optimal and has a lower delay when compared to the other existing distributed protocols. We achieved this by introducing a small amount of signaling between neighboring nodes and we saw that this simple functionality just be included in any distributed scheduling protocol that uses queue lengths to determine the schedules. The reduction in delay can be attributed to the fact that the queue-lengths converge faster in the SQ-CSMA due to the inclusion of switching. The simulations presented in Chapter 7 also emphasize this claim. The motivation behind this protocol was from the work on Markov chain on independent sets by Vigoda, Dyer, Greenhill et al. ([11],[12]). In [11], the authors show that a Markov chain defined on unweighted independent sets mixes faster when switching is allowed between neighboring nodes. Our model is an extension of this system with weights being associated with each independent set in terms of the queue lengths of the links. An interesting extension to our problem would be to reiterate the claim made using simulations by performing a similar mixing time analysis for our system.

Adding (even the most basic) messaging and coordination lowers the delay of queue based CSMA. This thesis just takes a first step at lowering the delay. In principle, we can have a sequence of protocols that trade-off complexity and delay while remaining throughput optimal. A more interesting question to address would be to include non-reversible steps, for e.g., a link with large queue switching off all its active neighbors while turning itself ON. However, non-reversible throughput optimal algorithms are not known.

Bibliography

[1] L. Jiang and J. Walrand. A distributed CSMA algorithm for Throughput and Utility Maximization in Wireless Networks. In *Proceedings 46th Annual Allerton Conference on Communication, Control and Computing*, 2008.

[2] S. Rajagopalan, D. Shah, and J. Shin. Aloha that works. November 2008.

[3] J. Ni, B. Tan, and R. Srikant. Q-CSMA: Queue length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks. In *Proc. IEEE INFOCOM Mini-Conference*, 2010.

[4] L. Tassiulas and A. Ephremides. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximal Throughput in Multihop Radio Networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, December 1992.

[5] A. Dimakis and J. Walrand. Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits. *Advances in Applied Probabilities*, 38(2):505–521, 2006.

[6] C. Joo, X. Lin, and N. B. Shroff. Understanding the capacity region of the greedy maximal scheduling algorithm in multihop wireless networks.

In *In Proceedings of IEEE INFOCOM*, April 2008.

[7] M. Leconte, J. Ni, and R. Srikant. Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks. In *In Proceedings of ACM MOBIHOC*, May 2009.

[8] R. R. Boorstyn, A. Kershenbaum, B. Maglaris, , and V. Sahin. Throughput Analysis in Multihop CSMA Packet Radio Networks. *IEEE Transactions on Communications*, 35(3):267–274, March 1987.

[9] S. Rajagopalan and D. Shah. Distributed algorithm and reversible network. In *In Proceedings of CISS*, March 2008.

[10] A. Eryilmaz, R. Srikant, and J. R. Perkins. Stable scheduling policies for fading wireless channels. *IEEE/ACM Transactions on Networking*, 13(2):411–424, April 2005.

[11] M. Dyer and C. Greenhill. On Markov Chains for Independent Sets. *Journal of Algorithms*, pages 17–49, 2000.

[12] E. Vigoda. A Note on the Glauber Dynamics for Sampling Independent Sets. January 2001.

[13] L. Jiang and J. Walrand. Approaching Throughput-Optimality in a Distributed CSMA Algorithm: Collisions and Stability. *(invited), ACM Mobihoc'09 S3 Workshop*, May 2009.

Vita

Rajaganesh Ganesh was born in Tanjore, Tamilnadu, India on 11 July 1987, the son of Ganesh Krishnamoorthy and Susila Ganesh. He received the Bachelor of Engineering degree from the College of Engineering, Guindy in Electronics and Communications Engineering. He applied to the Electrical Engineering department at the University of Texas at Austin for enrollment in their Masters program. He was accepted and started graduate studies in August, 2009.

Permanent address: 3703 Harmon Avenue Apt 202
Austin, Texas 78705

This thesis was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.