A Node-Based CSMA Algorithm For Improved Delay Performance in Wireless Networks

Sherif ElAzzouni
Dept. of Electrical and Computer Engineering
The Ohio State University, Columbus, OH 43210
elazzouni.1@osu.edu

Eylem Ekici
Dept. of Electrical and Computer Engineering
The Ohio State University, Columbus, OH 43210
ekici@ece.osu.edu

ABSTRACT

Recent studies in wireless scheduling have shown that CSMA can be made throughput optimal by optimizing over activation rates. However, those throughput optimal CSMA algorithms were found to suffer from poor delay performance, especially at high throughputs where the delay can potentially grow exponentially in the size of the network. Motivated by these shortcomings, in this paper we propose a node-based version of the throughput optimal CSMA (NB-CSMA) as opposed to previous link-based CSMA algorithms, where links were treated as separate entities. Our algorithm is fully distributed and corresponds to Glauber dynamics with "Block updates". We show analytically and via simulations that NB-CSMA outperforms conventional link-based CSMA in terms of delay for any fixed-size network. We also characterize the fraction of the capacity region for which the average queue lengths (and the average delay) grow polynomially in the size of the network, for networks with bounded-degree conflict graphs. This fraction is no smaller than the fraction known for link-based CSMA, and is significantly larger for a special class of wireless ad-hoc networks.

CCS Concepts

•**Networks** → **Network protocols; Network algorithms;** *Network performance evaluation; Wireless mesh networks;*

Keywords

CSMA, Wireless Scheduling, Delay Analysis, Distributed Algorithms.

1. INTRODUCTION

Scheduling is an essential task for resource allocation in communication networks. This task is especially challenging in wireless networks due to inherent mutual interference among wireless links, and for various networks, the absence of central control and/or resource management decision making. Typically, a good scheduling algorithm should be able to achieve three goals; i) **High Throughput:** Characterized by the fraction of the network capacity region a scheduling algorithm achieves. Ideally a scheduling algorithm should be able to support any set of arrival rates within the capacity region. ii) **Low Delay:** A good scheduling algorithm should be able to maintain the throughput required by the application without incurring excessive delay at any of the links. Furthermore, the expected delay should scale favorably with the size of the network. iii)**Low Complexity:** Required to ensure easy implementation and to minimize resources required to run the algorithm.
The seminal work of [25] is the first example of a throughput-optimal scheduling algorithm, which can support any arrival rate vector within the network capacity region without any of the link queues growing to infinity. It was shown that if the interference relationships of the network is modeled by a conflict graph, the max-weight algorithm, where the weight of the link is taken to be the queue size, is throughput optimal. However, max-weight based algorithms suffer from high complexity: In general networks, determining the maximum weight independent set is NP-hard.

A recent breakthrough in wireless network scheduling happened when it was shown that CSMA-like algorithms can be made throughput optimal if every link's activation rate is optimized [13] or taken to be an appropriate function of the queue length [22], [21]. This result is attractive because CSMA algorithms are fully distributed. The idea behind such algorithms is to run a Markov chain of collision-free schedules that have a stationary distribution approximating the max-weight solution. When the Markov chain converges to the max-weight solution, throughput optimality is achieved. A major shortcoming of these algorithms is their delay performance which has been shown to be unsatisfactory for many cases [4]. For example, in [19], it has been shown that the delay can grow exponentially with the size of the network in general graphs. Furthermore, it was shown in [23] that there exists worst case topologies, such that, even to attain a fraction of the capacity region, either the delay or the complexity must increase exponentially. The canonical example that illustrates the poor performance of CSMA is the network that has a torus or lattice conflict graph. We can easily see the existence of two optimal schedules, i.e., the "odd" and the "even" schedules. As the network size increases the transitions between the "odd" and "even" schedules become less frequent, causing the average delay to increase. This is known as the starvation problem of CSMA [15]. CSMA gets stuck in a "good" schedule for a very long

MobiHoc'16, July 04-08, 2016, Paderborn, Germany
© 2016 ACM. ISBN 978-1-4503-4184-4/16/07...$15.00
DOI: http://dx.doi.org/10.1145/2942358.2942359

time.

Throughput-optimal CSMA algorithms tend to treat links as separate autonomous entities that do not communicate. This is not true in many instances of wireless networks. In many practical wireless network deployments (e.g. wireless mesh networks and wireless ad-hoc networks) nodes typically control multiple outgoing links. Furthermore, it is a desirable trait for wireless ad-hoc networks to be k-connected [3] to ensure connectivity and fault tolerance. This means that every node will have a minimum of k-outgoing links. We use this fact to motivate our proposed Node-Based CSMA (NB-CSMA), where scheduling decisions are made on a node level rather than a link level. The node-based CSMA implementation was touched upon in [21], however the node based implementation is described as a straight-forward extension to the link based Q-CSMA, that is, it still relies on "single-site updates" in the underlying Glauber dynamics as opposed to our NB-CSMA that relies on updating a number of vertices in the conflict graph jointly. The motivation behind a our NB-CSMA algorithm is our observation that at high throughput regimes, activation rates tend to be high and links tend to be increasingly greedy acquiring and keeping the medium. Thus, a node's ability to switch between two links in one slot without having to go through an idle slot is expected to make switching between dominant schedules more frequent. This causes the starvation period of all links to decrease. Our contributions can be summarized as follows:

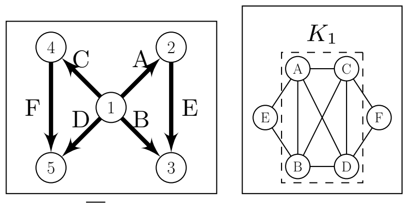
- C1 We propose a new throughput-optimal distributed Node-Based CSMA (NB-CSMA) algorithm, where the scheduling decisions are made on a node level rather than a link level.

- C2 We compare the Node-Based CSMA (NB-CSMA) to the link based CSMA (Q-CSMA) [21] in terms of expected delay for any fixed network. We show analytically and via simulations that NB-CSMA performs no worse than Q-CSMA.

- C3 We use mixing time analysis to characterize the fraction of the capacity region where under the NB-CSMA algorithm, the expected queue lengths and expected delay can be bounded by a polynomial in the size of the network (as opposed to an exponential bound). We show that this fraction is no smaller than the known fraction of capacity region under Q-CSMA.

2. RELATED WORK

Delay performance of throughput optimal CSMA algorithms have been discussed in the literature in several works. In [12], it was shown via a mixing time analysis that for bounded-degree graphs and for a fraction of the capacity region, the delay growth is upper bounded by a polynomial in the size of the network. In [24], it was further shown that for a reduced fraction of the capacity region, the delay is bounded by a constant independent of the network size. In addition to delay analysis, much of the existing research focused on how to alter the CSMA algorithms to improve the network delay performance while maintaining throughput optimality and low complexity. In [19], resetting the algorithm periodically is proposed to prevent starvation in the network. This was shown to be order optimal (with respect to the size of the network) for networks that have a torus or lattice shaped conflict graphs. In [16], the authors propose a new update rule, where the Metropolis algorithm is used as a substitute to the underlying Glauber dynamics-based algorithm to update the schedule. This was shown to have better delay performance for fixed sized networks. In [26], a modified version of the CSMA algorithm is proposed, where only links with queue lengths exceeding a certain threshold where allowed to contend for the medium. This has the effect of reducing the number of contending links every time slot and consequently reducing average delay. Elegant solutions to the delay problem were proposed in [11] and [14] where multiple Markov chains of collision free schedules are run in parallel, and the actual schedule is chosen from them probabalistically [11] or periodically [14]. Intuitively, as the number of parallel Markov Chains increases, the probability that the scheduling algorithm gets "stuck" in one good schedule for a long time decreases, decreasing the expected delay. It was also showed in [17] that the delayed-CSMA proposed in [14], with a suitable number of parallel schedules, achieves order optimal per link steady state delay. However, this vast improvement of steady state delay comes at the cost of fast increase of convergence time as the number of parallel schedules increases. This has a detrimental effect on the transient delay, i.e., these algorithms have favorable steady state delay performance, however the time to get to that steady state can still be exponential in the size of the network. Therefore, the resulting delay performance can still be unsatisfactory.



(a) $\overline{G}(K,V)$ (b) $G(V,E)$

Figure 1: An Example of a simple 5 node network topology and the corresponding conflict graph

3. SYSTEM MODEL

We model the wireless network by the connectivity graph $\overline{G}(K,V)$ where K is the set of nodes in the network and V is the set of the directional links with an arbitrary interference relationship (M-Hop, Geometric, ..etc). The interference relationship between different links in the network can be represented by a conflict graph $G(V,E)$, where V is the set of communication links in the network. An edge $(i,j) \in E$ exists if link i and link j interfere with each other. We define the neighbors of a link v, $N_v = \{w \in V : (v,w) \in E\}$. Neighboring links in the conflict graph are not allowed to transmit simultaneously to avoid collision. We assume that a node can only activate one outgoing link in each slot, which is the case in most wireless networks. Note that under this assumption, outgoing links of node $k \in K$ form a clique (complete subgraph) in the conflict graph. We denote this set of links as K_k. Fig. 1 is an example of this modeling for a simple 5-node network with primary exclusive constraints, i.e., links can be scheduled if and only if they constitute a matching in $\overline{G}$. The clique K_1 is also highlighted as an example.

We define a schedule $\mathbf{s}(t) = (s_v)_{v \in V}(t) \in \{0,1\}^{|V|}$ which represents a set of transmitting links in a given time slot, i.e., link v is active at time t if $s_v(t) = 1$. We use the bold notation $\mathbf{s}(t)$ to denote the schedule of all links in a certain time slot. The mean service rates of all links are $\mathrm{E}(\mathbf{s}(t)) = \mu$. A **feasible** schedule is one which does not violate the conflict relationships of E, therefore a schedule $\mathbf{s}$ is feasible if $s_i + s_j \leq 1 \; \forall (i,j) \in E$. We denote the set of all feasible schedules by $\mathbf{\Omega}$. Note that a feasible schedule is mapped to an independent set of vertices in the conflict graph.

Each link v has a queue q_v to store incoming packets. We assume an independent arrival process $a_v(t)$ at each link with a mean equal to $\mathrm{E}(a_v(t)) = \nu_v$. The queue dynamics for each link are given by

$$q_v(t+1) = (q_v(t) - s_v(t) + a_v(t))^+ \;\; \forall v \in V \tag{1}$$

We assume all traffic is single hop, where a packet exits the network right after a successful transmission.

The capacity region of the network $\mathbf{\Lambda}$ is defined as the set of arrival rates $\{\nu_v\}_{v \in V}$ of which there exists a scheduling algorithm that can stabilize all the queues. From [25], it is known that the capacity region of the network is

$$\Lambda = \{\nu \geq 0 : \exists \mu \in \mathbf{Co}(\Omega), \nu < \mu\} \tag{2}$$

where $\mathbf{Co}(.)$ is the convex hull operator, and vector inequalities are component-wise. A scheduling algorithm is throughput-optimal if it can stabilize all queues in the network for any arrival rate $\nu \in \mathbf{\Lambda}$.

CSMA: Glauber Dynamics Single Site Update

We now briefly explain the discrete time throughput optimal link-based CSMA proposed in [21] (Q-CSMA) to motivate and introduce our NB-CSMA algorithm in the next section. The throughput-optimal CSMA algorithm applies the glauber dynamics from statistical physics with appropriate activation rates. Here, a feasible schedule corresponds to an independent set in the conflict graph G. One application of Glauber dynamics is sampling independents sets on a graph. This is known in the statistical physics literature as the hardcore model (see [18] for details and examples). Q-CSMA is basically an application of glauber dynamics on the conflict graph to sample weighted independent sets. In every time slot t, a link v is randomly chosen to update as follows :

- If $\sum_{w \in N_v} s_w(t) = 0$, then $\begin{cases} s_v(t+1) = 1 \text{ w.p. } \frac{\lambda_v}{1+\lambda_v} \\ s_v(t+1) = 0 \text{ w.p. } \frac{1}{1+\lambda_v} \end{cases}$
- Otherwise, $s_v(t+1) = 0$
- and for all $w \neq v$, let $s_w(t+1) = s_w(t)$

The schedule $\mathbf{s}(t)$ then forms an irreducible aperiodic reversible Markov chain with stationary distribution

$$\pi(\mathbf{s}) = \frac{1}{Z} \prod_{v \in V} \lambda_v^{s_v} \tag{3}$$

where $Z = \sum_{s_v \in \Omega} \prod_{v \in V} \lambda_v^{s_v}$ is a normalizing constant. The throughput optimal Q-CSMA [21] uses a modified version of the Glauber dynamics where multiple parallel updates at non-conflicting links are allowed using the same update rule. The fugacity, λ_v, of any link v changes dynamically over time depending on a local weight function w_v that is taken as a concave nondecreasing function of the queue length q_v. For example, if the fugacity λ_v is taken as

$$\lambda_v = e^{w_v}, \tag{4}$$

then this implies that the stationary distribution of the Q-CSMA is

$$\pi(\mathbf{s}) = \frac{1}{Z} e^{\sum_{v \in \mathbf{s}} w_v(t)} \tag{5}$$

The intuition behind the throughput optimality is that the Q-CSMA approximates the max-weight solution.
The extension to a node-based implementation in [21] describes a protocol to determine which nodes can update their links using an RTS/CTS mechanism. Selected nodes then choose an outgoing link uniformly to update link activity using the Q-CSMA rules. Clearly, the node-based implementation of Q-CSMA does not allow switching between outgoing links. This motivates our Node-Based CSMA (NB-CSMA).

4. NODE BASED CSMA : GLAUBER DYNAMICS WITH BLOCK UPDATES

In this section, we introduce our proposed NB-CSMA algorithm. The reason behind Q-CSMA's poor delay performance is the need for high activation rates that causes all links to be greedy when contending for the medium, which cause the Q-CSMA to converge to one good schedule and remain at that schedule (or fluctuate around it) for a long time. In NB-CSMA, we attempt to solve that problem by examining the possibility of directly switching between links that share a common transmitter. The intuition is that the network will then switch between dominant schedules more often, without having to pay the price of an idle time slot every time a switch happens.

4.1 Step 1: Forming Blocks

We now explain how NB-CSMA works. Recall that the outgoing links of every node $k \in K$ are mapped to a clique in the conflict graph that we call K_k. At the beginning of each slot, a subset of nodes is selected for update. Each one of those selected nodes chooses a subset of outgoing links to update in that slot. We call this subset of links the update clique, C_k, for a node k that is allowed to update this slot. The selection of update cliques C_k is made such that

1. $C_k \subseteq K_k$
2. For any two nodes that will update at the same slot $k, l \in K$, we have $(v, w) \notin E \;\; \forall v \in C_k, w \in C_l$

The first condition simply states that each update clique C_k is a subset of the outgoing links that have a common transmitter node $k \in K$. This ensures that the algorithm is fully distributed with respect to the nodes, i.e., the only information needed to make a scheduling decision comes from the queue lengths at each physical node. The second condition is more subtle: It states that the links (vertices on conflict graph) of update cliques should not have an edge between them, i.e., they do not interfere. This requirement is necessary to ensure that the resulting schedule is feasible since the update clique can turn any of its links on. We denote the collection of update cliques at each time slot as $\overline{\mathbf{C}}$. A simple RTS/CTS scheme can be used, whereby each link waits a random back-off time, and then attempts to add itself to the update clique if no conflict exists, to randomly determine all update cliques $\overline{\mathbf{C}}$ in each time slot such that they satisfy the above two conditions. We denote the probability of choosing a particular update clique collection $\overline{\mathbf{C}}$ at any time slot as $P(\overline{\mathbf{C}})$.

4.2 Step 2: Updating Blocks

Now that the update cliques $\overline{\mathbf{C}}$ have been found, we proceed to explain how the schedule shall be updated. We call $\mathbf{x}(t+1)$ the proposed schedule. This schedule will be used to obtain the actual schedule $\mathbf{s}(t+1)$. The NB-CSMA update procedure is given in Algorithm 1.

```
1  for $C_k$ in $\overline{\mathbf{C}}$ do
2      if $\exists v \in C_k$ s.t. $s_v(t) = 1$ then
3          w.p. $\frac{1}{|C_k|}$
4              $s_v(t+1) = 1, s_w(t+1) = 0, \ \forall w \in C_k \setminus v$ w.p. $\frac{\lambda_v}{1+\lambda_v}$
5              $s_v(t+1) = 0, s_w(t+1) = 0, \ \forall w \in C_k \setminus v$ w.p. $\frac{1}{1+\lambda_v}$
6          w.p. $\frac{|C_k|-1}{|C_k|}$
7              $x_w(t+1) = 1, x_z(t+1) = 0, \ \forall z \in C_k \setminus w$ w.p. $\frac{\lambda_w}{\sum_{z\in C_k}(1+\lambda_z)}, \ \forall w \in C_k \setminus v$
8              $x_w(t+1) = s_w(t), \ \forall w \in C_k$ Otherwise
9      else
10         pick link $w \in C_k$ uniformly at random
11         $x_w(t+1) = 1$ w.p. $\frac{\lambda_w}{1+\lambda_w}$
12         $x_w(t+1) = 0$ w.p. $\frac{1}{1+\lambda_w}$
13     end
14     if $x_w(t+1) + s_z(t) \leq 1, \ \forall w \in C_k, \ \forall z \in N_w \setminus C_k$ then
15         $s_w(t+1) = x_w(t+1), \ \forall w \in C_k$
16     else
17         $s_w(t+1) = s_w(t), \ \forall w \in C_k$
18     end
19 end
```

Algorithm 1: NB-CSMA Algorithm

We can see that lines 3-8 in Algorithm 1 describe an operation where an already active link is chosen to be updated according to a procedure similar to the Q-CSMA. Similarly, lines 10-12 describe how an inactive link with no active neighbor is chosen randomly and added to the schedule with some probability. This is again similar to the Q-CSMA procedure, i.e., lines 3-8 and 10-12 just describe the classical Glauber dynamics to generate independent sets with some desired distribution. Lines 6-8 represent the addition provided by making the CSMA algorithm node based. A node can switch from one active link to an inactive one according to the probability defined in line 7. Lines 14-18 state that a link update is only accepted if the new schedule is an independent set to ensure that no collisions happen; otherwise the links' state in the new time slot remains identical to the one in the previous time slot.

Note that the NB-CSMA algorithms corresponds to performing *"Block updates"* as opposed to single site updates in Q-CSMA. The block updates have been used before for glauber dynamics to analyze the mixing time [20], [18].

5. PERFORMANCE OF THE NB-CSMA ALGORITHM

Since the schedule $\mathbf{s}(t+1)$ depends only on the schedule of the previous time slot $\mathbf{s}(t)$, the evolution of schedules over time forms a Discrete Time Markov Chain (DTMC) with state space $\mathbf{\Omega}$. To write the transition probability between two feasible schedules $\mathbf{s}$ and $\mathbf{s}'$, it is useful to look at the conditional transition probabilities given a particular choice of update cliques $\overline{\mathbf{C}}$. Let $\mathbf{s}_{C_k}(t)$ be the schedule of clique C_k at time slot t. We can breakdown update cliques into three types and characterize their different transition probabilities $P(\mathbf{s}_{C_k}, \mathbf{s}'_{C_k})$

1. **Type A:** C_A: $s_w(t) = 0, \ \forall w \in C_A$ and $s'_v(t+1) = 1, \ \exists v \in C_A$ and $P(\mathbf{s}_{C_A}, \mathbf{s}'_{C_A}) = \frac{1}{|C_A|}\frac{\lambda_v}{1+\lambda_v}$
2. **Type B:** C_B: $s_v(t) = 1, \ \exists v \in C_B$ and $s'_w(t+1) = 0, \ \forall w \in C_B$ and $P(\mathbf{s}_{C_B}, \mathbf{s}'_{C_B}) = \frac{1}{|C_B|}\frac{1}{1+\lambda_v}$
3. **Type C:** C_C: $s_v(t) = 1, \ \exists v \in C_C$ and $s'_w(t+1) = 1, \ \exists w \in C_C \backslash v$ and $P(\mathbf{s}_{C_C}, \mathbf{s}'_{C_C}) = \frac{|C_C|-1}{|C_C|}\frac{\lambda_w}{\sum_{z\in C_C}(1+\lambda_z)}$

Since changes over different cliques at each time slot are independent, the conditional transition probability given some update cliques $\overline{\mathbf{C}}$, $P(\mathbf{s}, \mathbf{s}'|\overline{\mathbf{C}})$, is calculated by multiplying the transition probabilities over every clique depending on the transition type as described above. Let $\overline{\mathbf{C}}_A$, $\overline{\mathbf{C}}_B$ and $\overline{\mathbf{C}}_C$ be the cliques that will have a transition of type A, B or C respectively. The total transition probability follows

$$P(\mathbf{s}, \mathbf{s}') = \sum_{\overline{\mathbf{C}}} P(\overline{\mathbf{C}}) P(\mathbf{s}, \mathbf{s}'|\overline{\mathbf{C}}) \tag{6}$$

$$= \sum_{(\overline{\mathbf{C}})} P(\overline{\mathbf{C}}) \Big(\prod_{C_A \in \overline{\mathbf{C}}_A} P(\mathbf{s}_{C_A}, \mathbf{s}'_{C_A}) \prod_{C_B \in \overline{\mathbf{C}}_B} P(\mathbf{s}_{C_B}, \mathbf{s}'_{C_B}) \prod_{C_C \in \overline{\mathbf{C}}_C} P(\mathbf{s}_{C_C}, \mathbf{s}'_{C_C}) \Big) \tag{7}$$

The DTMC of transmission schedules is both irreducible and aperiodic. Irreducibility can be checked by noticing that starting from the state $\{\mathbf{s}, s_v = 0 \ \forall v \in V\}$, any feasible state $\mathbf{s}' \in \Omega$ can be reached in finite number of steps. Aperiodicity is easy to check as well, by noticing that every schedule $\mathbf{s} \in \Omega$ has a self transition if every clique in the update cliques makes no transition, which always happens with a non-zero probability.

THEOREM 1. *The DTMC is reversible with stationary distribution given by*

$$\pi(\mathbf{s}) = \frac{1}{Z} \prod_{v \in V} \lambda_v^{s_v} \tag{8}$$

where $Z = \sum_{s_v \in \Omega} \prod_{v \in V} \lambda_v^{s_v}$ *is a normalizing constant.*

PROOF. Given the update cliques $\overline{\mathbf{C}}$, for two feasible schedules $(\mathbf{s}, \mathbf{s}')$, define the symmetric difference as $\mathbf{s} \triangle \mathbf{s}' = (\mathbf{s} \setminus \mathbf{s}') \cup (\mathbf{s}' \backslash \mathbf{s})$. It is easy to see that for the transition to happen, the update cliques should fulfill the condition that

$$\mathbf{s} \triangle \mathbf{s}' \subseteq \overline{\mathbf{C}} \tag{9}$$

Given any such selection of update cliques, we have for any $C_k \in \overline{\mathbf{C}}$:

1. If $\exists! v \in C_k \cap (\mathbf{s}' \setminus \mathbf{s})$ s.t. $\mathbf{s} \triangle \mathbf{s}' \cap C_k = v$, then $P(\mathbf{s}_{C_k}, \mathbf{s}'_{C_k})$ is a Type A transition.
2. If $\exists! w \in C_k \cap (\mathbf{s} \setminus \mathbf{s}')$ s.t. $\mathbf{s} \triangle \mathbf{s}' \cap C_k = w$, then $P(\mathbf{s}_{C_k}, \mathbf{s}'_{C_k})$ is a Type B transition.
3. If $\exists x, y \in C_k$ s.t. $x \in (\mathbf{s} \backslash \mathbf{s}')$ and $y \in (\mathbf{s}' \backslash \mathbf{s})$, then $P(\mathbf{s}_{C_k}, \mathbf{s}'_{C_k})$ is a Type C transition, where $\exists! a$ means "there exists a unique a".

A straight-forward substitution shows that

$$\pi(\mathbf{s}) P(\mathbf{s}, \mathbf{s}'|\overline{\mathbf{C}}) = \pi(\mathbf{s}') P(\mathbf{s}', \mathbf{s}|\overline{\mathbf{C}})) \tag{10}$$

$$\sum_{\overline{C}} P(\overline{\mathbf{C}}) \pi(\mathbf{s}) P(\mathbf{s}, \mathbf{s}'|\overline{\mathbf{C}}) = \sum_{\overline{C}} P(\overline{\mathbf{C}}) \pi(\mathbf{s}') P(\mathbf{s}', \mathbf{s}|\overline{\mathbf{C}})) \tag{11}$$

$$\pi(s) P(\mathbf{s}, \mathbf{s}') = \pi(s') P(\mathbf{s}', \mathbf{s}) \tag{12}$$

Thus the stationary distribution satisfies the detailed balance equations. □

5.1 Throughput Optimality

NB-CSMA is throughput optimal under the time scale separation assumption, i.e., assuming that the DTMC of schedules converges on a fast time-scale with respect to the queue evolution. The schedules can then be assumed to be always at the stationary distribution. This assumption was justified in [8] by choosing the weights to be a logarithmic function of the queue size. Throughput-optimality can be shown by defining a weight function of link v as

$$w_v = f(q_v), \forall v \in V \tag{13}$$

where f is a concave nondecreasing function of the queue length q. Then, if we take the link fugacity to be

$$\lambda_v = e^{w_v}, \tag{14}$$

we obtain the stationary distribution of the DTMC as

$$\pi(\mathbf{s}) = \frac{1}{Z} e^{\sum_{v \in \mathbf{s}} w_v(t)}. \tag{15}$$

We note that the stationary distribution of NB-CSMA is equal to the one of Q-CSMA in [21], that is, in terms of the stationary distribution of schedules, NB-CSMA and Q-CSMA are equal. It is the second-order behavior that is different. This implies that NB-CSMA is throughput optimal under time-scale separation assumption, since it has the "correct" stationary distribution. Intuitively, the NB-CSMA approximates the max-weight solution that is known to be throughput optimal every time slot. A detailed derivation that a distributed algorithm that achieves the stationary distribution in (15) is throughput optimal can be found in [21]. We do not repeat it here for the sake of brevity, since this proof directly applies to NB-CSMA.

5.2 Delay Perfromance

Due to complex interactions between different links, the delay performance is generally hard to assess. However, several tools have been used to analyze the delay of CSMA-like algorithms. We begin our delay performance assessment by comparing the delay performance between Q-CSMA and NB-CSMA for a fixed size network, showing that under NB-CSMA algorithm, each link sees a service process that has less variability, implying better delay performance. We then proceed to test how NB-CSMA scales as the network size increases. In particular, we are interested in characterizing the fraction of the capacity region where the queue sizes can be bounded polynomially in the number of links (as opposed to exponentially). A similar result for Q-CSMA was found in [12]. We show that the fraction of the throughput region where the NB-CSMA is fast mixing is usually larger that the one found in [12], except for some special cases when the two regions coincide. Before we proceed, we make three simplifying assumptions for the sake of tractability:

- A1 All fugacities, λ_v $\forall v$, are fixed and possibly heterogeneous, as opposed to the dynamic fugacities used to prove throughput optimality. Suitable fixed fugacities can be found using the problem formulation/solution in [13] to stabilize any arrival rate vector in the capacity region.
- A2 We assume for our comparison purposes that both NB-CSMA and Q-CSMA perform a single update per time slot, i.e., in Q-CSMA one link (vertex in the conflict graph) is updated, and in NB-CSMA one node (clique in the conflict graph) is updated. This does not change the stationary distribution in (8); it just provides easier grounds for comparison at the cost of slower convergence to the stationary distribution since only one update per slot is allowed.
- A3 We assume that in Q-CSMA, each time slot, a link is chosen uniformly at random to be updated (with probability $\frac{1}{n}$, where n is the number of links). We assume on the other hand in NB-CSMA, each time slot a node k is updated at random with probability proportional to the number of outgoing links (with probability $\frac{|K_k|}{n}$). This makes the probability that a particular link is turned ON or OFF equal in both Q-CSMA and NB-CSMA.

5.2.1 *Comparison to Q-CSMA for fixed size networks*

In a fixed size network setting, we are interested in comparing the steady-state delay performance between Q-CSMA and NB-CSMA. The approach we use is similar to the one used in [16] to compare fixed size networks. This approach depends on looking at the service process of any link v in isolation. First, define the service process of link v under NB-CSMA and Q-CSMA algorithms as $\sigma_v(t)$ and $\tilde{\sigma_v}(t)$, respectively. Also define $P, \tilde{P}, \pi, \tilde{\pi}$ to be the transition matrix and the stationary distribution of the NB-CSMA and the Q-CSMA markov chains, respectively. Throughout this analysis, we assume that both Markov chains P and $\tilde{P}$ have already converged to their steady state distribution.

Let B be the subset of schedules that include link v in them: $B = \{\mathbf{s} \in \Omega : s_v = 1\}$. The service process $\sigma_v(t)$ is a $0-1$ process with

$$P(\sigma_v(t) = 1) = \sum_{\mathbf{s} \in B} \pi(s). \tag{16}$$

We define

$$\begin{aligned} T_1 &= \min\{t \geq 0 : \sigma_v(t) = 1\} && (17)\\ T_{i+1} &= \min\{t > T_i : \sigma_v(t) = 1\}. && (18) \end{aligned}$$

For the states $s \in B$, Let $\tau_i = T_{i+1} - T_i$ $(i \geq 1)$ be the sequence of recurrence times and π_B be their steady state probability. The quantities $\tilde{\pi}_B, \tilde{\sigma}(t), \tilde{\tau}_i$ are defined similarly. We are interested in comparing the quantities $\mathrm{E}(\tau_i), \mathrm{E}(\tau_i^2)$ with the quantities $\mathrm{E}(\tilde{\tau}_i), \mathrm{E}(\tilde{\tau}_i{}^2)$.

THEOREM 2. *Under assumptions A1-A3, for any link v, the following inequalities hold*

$$\begin{aligned} \mathrm{E}(\tau_i) &= \mathrm{E}(\tilde{\tau}_i) && (19)\\ \mathrm{E}(\tau_i^2) &\leq \mathrm{E}(\tilde{\tau}_i^2) && (20) \end{aligned}$$

PROOF. When comparing the reversible Markov Chains $P, \tilde{P}$ we first notice that $\pi(\mathbf{s}) = \tilde{\pi}(\mathbf{s}), \forall \mathbf{s} \in \Omega$ by (3) and (8). It is a well-known fact of Markov Chains that the stationary distribution of a state is the reciprocal of its expected recurrence time, i.e., for any state (or group of states) the following holds by (3) and (8)

$$\frac{1}{\mathrm{E}(\tau_i)} = \pi_B = \tilde{\pi}_B = \frac{1}{\mathrm{E}(\tilde{\tau}_i)} \tag{21}$$

Next we define the Dirichlet form $\varepsilon(f, f)$ for functions $f : \Omega \to \mathbb{R}$ [18] by

$$\varepsilon(f, f) = \frac{1}{2} \sum_{x,y \in \Omega} (f(x) - f(y))^2 \pi(x) P(x, y) \tag{22}$$

The comparison method in [6] provides a method to compare the Dirichlet forms of two reversible Markov Chains defined on the same state space Ω to obtain linear inequalities between them when the Markov Chains do not necessarily have a linear relationship. Define $E = \{(x,y) : P(x,y) > 0\}$. An E-path from x to y is a sequence $\Gamma = (e_1, e_2, ..., e_m)$ of edges in E such that $e_1 = (x, x_1), e_2 = (x_1, x_2), ..., e_m = (x_{m-1}, y)$ for some states $x_1, ..., x_{m-1} \in \Omega$. The length of E-path Γ is denoted by $|\Gamma|$. Suppose that for each $(x,y) \in \tilde{E}$ there is an E-path from x to y. We refer to this path as Γ_{xy}. Now, define the congestion ratio as

$$A = \max_{(z,w)\in E} \frac{1}{\pi(z)P(z,w)} \sum_{\tilde{E}(z,w)} |\Gamma_{xy}|\tilde{\pi}(x)\tilde{P}(x,y) \tag{23}$$

The comparison theorem then states that

$$\tilde{\varepsilon}(f,f) \le A\varepsilon(f,f). \tag{24}$$

The key to calculating the congestion ratio A is noticing that all the Q-CSMA Markov chain's transitions are contained within the NB-CSMA Markov chain's transitions. In particular, line 4, line 5, line 11 and line 12 of Algorithm 1 are exactly Q-CSMA operations. Furthermore, the first and third assumption ensures that the probability of "refreshing" any link $v \in V$ is equal for both Q-CSMA and NB-CSMA, i.e.,

$$\tilde{P}(x,y) = P(x,y), \forall x,y \in \Omega \text{ s.t. } \tilde{P}(x,y) > 0. \tag{25}$$

We argue that the extra "transitions" in NB-CSMA Markov chain entails better delay performance. To apply the comparison theorem, we simply take the E-path Γ_{xy} given any x and y to be (x,y). Furthermore, by the equation in (25), and the fact that both chains have the same stationary distribution, the computation of the congestion ratio in (23) gives $A = 1$. By the comparison theorem

$$\tilde{\varepsilon}(f,f) \le \varepsilon(f,f). \tag{26}$$

Define the hitting time H_B as ($\tilde{H}_B$ is defined similarly)

$$H_B = \min\{t \ge 0 : \sigma_v(t) = 1\}. \tag{27}$$

The hitting time H_B is the time needed to reach the subset of states B where link v is active. We are interested in the expected hitting time: The time needed to reach subset B starting from a randomly chosen state. By the formula in [16] (Presented originally in [1, Ch.3, Proposition 41]) we have

$$\mathrm{E}(H) = \sup_g\{\frac{1}{\varepsilon(g,g)} : -\infty < g < \infty, g(.) = 1 \text{ on } B \text{ and } \sum_{\mathbf{s}\in\Omega} \pi(\mathbf{s})g(\mathbf{s}) = 0\} \tag{28}$$

By the equality of stationary distributions of the two chains, and substituting with the inequality relating their Dirichlet forms (26) in (28), we get that

$$\mathrm{E}(H) \le \mathrm{E}(\tilde{H}), \tag{29}$$

Again from [1], we obtain an important relationship relating the recurrence time and the hitting time

$$\mathrm{E}(\tau_i^2) = \frac{2\mathrm{E}(H) + 1}{\pi_B} \tag{30}$$

Substituting Inequality (29) in (30)

$$\mathrm{E}(\tau_i^2) = \frac{2\mathrm{E}(H) + 1}{\pi_B} \le \frac{2\mathrm{E}(\tilde{H}) + 1}{\pi_B} = \mathrm{E}(\tilde{\tau}_i^2) \tag{31}$$

□

We conclude that when looking at any link v in isolation, the expected recurrence time of the service process is equal for both the Q-CSMA and NB-CSMA. Thus, on average, q_v sees the same service rate under both algorithms. However, the second order characteristics are different as NB-CSMA results in less variability in the service process. Recall that when comparing M/G/1 queues with the same arrival and mean service rates, using the P-K formula [2], the service process with less variance gives a shorter average queue length and thus a smaller average delay. This suggests that when viewed in isolation, the queue of link v has a lower average length under the NB-CSMA protocol and sees a smaller average delay, as Q-CSMA, on average, starves link v for longer periods of time.
It remains in this section to discuss the validity of our assumptions. Assumptions A2 and A3 ensure that (25) holds. While those two assumptions would hold if we used a continuous-time algorithm like the one in [22], in discrete time algorithms, the transition probability depends on both the access probability, which we take uniformly as $\frac{1}{n}$ and the ON/OFF probabilities of the links. In general, the access probabilities are not uniform (except for special cases such as collocated networks, i.e., networks with a complete interference graph). The network randomly picks an independent set of links to update. This is usually done using a contention window. To get a fair comparison, we should compare between an optimized access scheme for both the Q-CSMA and NB-CSMA. However, the optimized access scheme for Q-CSMA has no closed form as shown in [9] and requires solving a network-wide hard optimization problem. We expect it to be even harder for NB-CSMA. Therefore, we rely on simulations to show that the results of Theorem 2 holds for any practical access scheme with assumptions A2 and A3 removed.

5.2.2 *Fast Mixing Activation Rates*

In this part, we are interested in characterizing the asymptotic delay performance of the NB-CSMA as the size of the network grows for networks with bounded-degree conflict graphs. It was shown in [19], that at high throughput values, the delay of the conventional CSMA algorithm grows exponentially with the number of the links in the network, which makes the delay performance unacceptable in large networks. Our goal is to characterize the throughput region where the average delay is bounded by a polynomial, because this is the region where the network is guaranteed to operate with acceptable delay performance. In [12], this region was shown to be contained in the region where the schedules Markov chain is "fast mixing". Thus, our target is to find the fraction of the capacity region that makes the network fast mixing.
We first define the total variation distance between the Markov chain distribution at time t starting from state x and the stationary distribution π:

$$d_{TV}(P_x^t, \pi) = \max_{A\subset\Omega} |P_x^t(A) - \pi(A)| \tag{32}$$

$$= \tfrac{1}{2}\sum_{y\in\Omega} |P^t(x,y) - \pi(y)| \tag{33}$$

The mixing time of the Markov Chain is defined as

$$T_{\mathrm{mix}}(\epsilon) = \min\{t \mid \max_x d_{TV}(P_x^t, \pi) < \epsilon\}. \tag{34}$$

Our tool for bounding the mixing time will be coupling. The coupling of two Markov chains is a pair process (X^t, Y^t) such that

1. Each of the Markov Chains (X^t, Y^t) when viewed in isolation remains faithful to the original Markov Chain.
2. If $X^t = Y^t$ then $X^{t+1} = Y^{t+1}$.

The mixing time is bounded by the stopping time taken for any two coupled processes to meet, that is

$$T_{mix} \leq \max_{x,y} \min\{t \ : x^t = y^t | X^0 = x, Y^0 = y\}. \tag{35}$$

Equivalently, instead of computing the stopping time explicitly, we can define a distance metric on Ω, and compute the time taken for the distance between the two processes to go to zero. *Path coupling* introduced by Bubley and Dyer [5] is a powerful tool that makes it easier to design and analyze couplings, by showing that to bound the mixing time, it is enough to restrict the couplings to pairs of states that are adjacent in the metric. This is much easier than bounding mixing time using couplings for arbitrary pairs of states. Formally:

THEOREM 3. *[5] Let δ be an integer valued metric on $\Omega \times \Omega$ which takes values $\{0, ..., D\}$. Let S be a subset of $\Omega \times \Omega$ such that for all $(x_t, y_t) \in \Omega \times \Omega$, ther exists a path $x_t = e_0, e_1, ..., e_r = y_t$ between x_t and y_t such that $(e_l, e_{l+1}) \in S$, $\forall\, 0 \leq l < r$ and*

$$\sum_{l=0}^{r-1} \delta(e_l, e_{l+1}) = \delta(x_t, y_t) \tag{36}$$

Define a coupling $(x, y) \to (x', y')$ on the Markov Chain P on all pairs $(x, y) \in S$ and suppose there exists $\beta \leq 1$ s.t. $\mathrm{E}(\delta(x', y')) \leq \beta\delta(x, y)$ for all $x, y \in S$. If $\beta < 1$ the mixing time $T_{mix}(\epsilon)$ satisfies

$$T_{mix}(\epsilon) \leq \frac{\log(D\epsilon^{-1})}{1 - \beta}. \tag{37}$$

Instead of analyzing the NB-CSMA Markov chain, P, we analyze a different Markov Chain Q that is linearly related to the original Markov chain P, and rely on the relationship between P and Q to estimate the mixing time of P. Throughout this section, we shall assume that assumptions A1, A2 and A3 hold. For notational convenience, we refer to the set of outgoing links of node k, that contains link v as K_v. Recall that K_v is also a clique in the conflict graph. Furthermore under A2, since only one node gets to update its schedule every time slot, we assume this node always chooses to update all outgoing links K_v.
Let $\lambda_1 \leq \lambda_2 ... \leq \lambda_{\max}$ be the set of fugacities of all links $v \in V$. Define the transitions of Markov Chain Q as specified in Algorithm 2. It is not difficult to check that the Markov Chain Q has a stationary distribution equal to that of (3) and (8).

THEOREM 4. *Given the Markov Chain Q,*
if $\lambda_{\max} < \frac{1}{\max_v(d_v - |K_v|)}$, then the mixing time of Q satisfies

$$T_{mix}(\epsilon) \leq \frac{2n(1 + \lambda_{\max}) \log(n\epsilon^{-1})}{1 - \max_v(d_v - |K_v|)\lambda_{\max}} = \mathcal{O}(n \log n). \tag{38}$$

PROOF. We choose our distance metric function $\delta(x, y)$ to be the Hamming distance between the two schedules x and y, $\delta(x, y) = \sum_{v \in V} \mathbb{1}(x_v \neq y_v)$. We take the subset $S \subseteq \Omega$ to be the states that are different at only 1 link , i.e., $\delta(x, y) = 1$. It is straightforward to check that the subset S satisfies the condition (36). Line 2 in Algorithm 2 has the effect of making the Markov chain *lazy*: A Markov chain is

```
1  Pick a Clique $K_v$ to update randomly w.p. $\frac{|K_v|}{n}$
2  w.p. $\frac{1}{2}$
3      $\mathbf{s}(t+1) = \mathbf{s}(t)$
4  w.p. $\frac{1}{2}$
5      if $\exists v \in K_v$ s.t. $s_v(t) = 1$ then
6          w.p. $\frac{1}{|K_v|}$
7              $s_v(t+1) = 1, s_w(t+1) = 0, \ \forall w \in K_v \setminus v$
               w.p. $\frac{\lambda_v}{1+\lambda_{\max}}$
8              $s_v(t+1) = 0, s_w(t+1) = 0, \ \forall w \in K_v \setminus v$
               w.p. $\frac{1}{1+\lambda_{\max}}$
9          w.p. $\frac{|K_v|-1}{|K_v|}$
10             $x_w(t+1) = 1, x_z(t+1) = 0, \ \forall z \in K_v \setminus w$
               w.p. $\frac{\lambda_w}{2(|K_v|-1)(1+\lambda_{\max})}, \ \forall w \in K_v \setminus v$
11             $x_w(t+1) = s_w(t), \ \forall w \in K_v$ otherwise
12     else
13         pick link $w \in K_v$ uniformly at random
14         $x_w(t+1) = 1$ w.p. $\frac{\lambda_w}{1+\lambda_{\max}}$
15         $x_w(t+1) = 0$ w.p. $\frac{1}{1+\lambda_{\max}}$
16     end
17     if
       $x_w(t+1) + s_z(t) \leq 1, \ \forall w \in K_v, \ \forall z \in N_w \setminus K_v$
       then
18         $s_w(t+1) = x_w(t+1), \ \forall w \in K_v$
19     else
20         $s_w(t+1) = s_w(t), \ \forall w \in K_v$
21     end
```

Algorithm 2: Evolution of Markov Chain Q

lazy if the probability of staying in any state is at least $\frac{1}{2}$. Making the Markov chain lazy makes the task of relating the mixing time of Q to the mixing time of P easier in Theorem 7. Also, making Q lazy has the effect of doubling the mixing time. Therefore we neglect this self transition (Line 2 and Line 3 in Algorithm 2) in the analysis and compensate for it by a factor of 2 at the end of the analysis. We will use the prime symbol to donate states after one slot has elapsed, for example $P(X', Y'|x, y)$ is the distribution of the schedule at time slot $t + 1$ given that the Markov chain was at state (x, y) at time t. The next step is to calculate $\mathrm{E}(\delta(x', y'))$, that is, the expected distance between the states after one time slot has elapsed.

Let x and y be two feasible schedules on Ω that agree everywhere except at link v. Suppose WLOG that $x_v = 1$ and $y_v = 0$. Note that this directly implies that $x_w = y_w = 0 \ \forall w \in N_v$. We run the Markov chain Q for one slot and estimate the expected distance metric after one time slot $\delta(x', y')$. There are 5 different cases that result in different $\mathrm{E}(\delta(x', y'))$. We define the coupling for each of these cases:

1. K_v is chosen to be updated w.p. $\frac{|K_v|}{n}$ for both (X, Y). Furthermore both (X, Y) choose link v to update w.p. $\frac{1}{|K_v|}$ (where x is performing lines 7, 8 of algorithm 2 and y is performing lines 13, 14 of algorithm 2), and $P(X' = X, Y' = X) = \frac{\lambda_v}{n(1+\lambda_{\max})}, P(X' = Y, Y' = Y) = \frac{1}{n(1+\lambda_{\max})}$. Thus, in this case $\delta(x', y') = 0$ w.p. 1.
2. K_v is chosen to be updated w.p. $\frac{|K_v|}{n}$ for both (X, Y), also both X and Y attempt to activate a new link $w \in K_v$ that has $x_z(t) = y_z(t) = 0, \ \forall z \in N_w \setminus v$.

Define the coupling as follows

$$P(X' = Y \cup w, Y' = Y \cup w) \quad = \tfrac{1}{2n} \tfrac{\lambda_w}{(1+\lambda_{\max})} \qquad (39)$$

$$P(X' = X, Y' = Y \cup w) \quad = \tfrac{1}{2n} \tfrac{\lambda_w}{(1+\lambda_{\max})} \qquad (40)$$

where x is performing lines 10, 11 and y is performing lines 14, 15 of Algorithm 2. Since Both contributions are equal we have $\mathrm{E}(\delta(x', y')) = 1$.

3. A link $w \in N_v \setminus K_v$ where $w \in C_w$ and $\sum_{w\in C_w} x_w = \sum_{w\in C_w} y_w = 0$. Now both x and y are performing lines 13, 14 of Algorithm 2. We have the following coupling

$$P(X' = X, Y' = Y \cup w) = \frac{\lambda_w}{n(1+\lambda_{\max})} \qquad (41)$$

$$P(X' = X, Y' = Y) = \frac{1}{n(1+\lambda_{\max})} \qquad (42)$$

Thus, in the first equation $\delta(x', y') = 2$, and in the second equation $\delta(x', y') = 1$.

4. A link $w \in N_v \setminus K_v$ where $w \in C_w$ and $\sum_{w\in C_w} x_w = \sum_{w\in C_w} y_w = 1$. Now both x and y are performing lines 10, 11 of Algorithm 2. We have the following coupling

$$P(X' = X, Y' = Y \cup w) = \frac{\lambda_w}{2n(1+\lambda_{\max})} \qquad (43)$$

$$P(X' = X, Y' = Y) = \frac{1}{2n(1+\lambda_{\max})} \qquad (44)$$

Thus, in the first equation $\delta(x', y') = 3$ and the second equation $\delta(x', y') = 1$.

5. A link w is chosen to updated where w does not fall in any of the previous four categories. In that case, the coupling is defined to make both X and Y perform the same update. In this case, we have $\delta(x', y') = 1$.

Let d_v be the interference degree of link v (number of neighbors in conflict graph or number of interferers). It is straightforward to see that there are at most $|K_v|-1$ links satisfying case 2, and at most $d_v - |K_v| + 1$ satisfying case 3 and case 4. By collecting the individual contributions of all cases we obtain the following result

$$\mathrm{E}(\delta(x', y') - 1) \le \frac{1}{n}\Big(\sum_{w\in N_v\setminus K_v} \frac{\lambda_w}{1+\lambda_{\max}} - 1\Big) \qquad (45)$$

$$\le \frac{1}{n}\Big((d_v - |K_v| + 1)\frac{\lambda_{\max}}{1+\lambda_{\max}} - 1\Big) \qquad (46)$$

Now by taking $\lambda_{\max} < \min_v \frac{1}{d_v - |K_v|} = \frac{1}{\max_v(d_v - |K_v|)}$, we get the β term in Theorem 3 as

$$\beta = 1 - \frac{1}{n}\frac{(1 - \max_v(d_v - |K_v|)\lambda_{\max} + 1)}{1+\lambda_{\max}} < 1 \qquad (47)$$

Directly applying Theorem 3 proves the theorem. □

Now that we have bounded the mixing time of Q, we need to relate it to the mixing time of P. We make use of the following Lemma stated as Theorem 5.3 in [7].

Lemma 5. *[7] Given two Markov chains P and Q, let their mixing times be T^P_{mix} and T^Q_{mix} respectively. If*

$$P \ge \alpha Q, \qquad (48)$$

then $T^P_{mix} \le 2\alpha^{-1} T^Q_{mix} \log(\pi^*(2\epsilon)^{-1})$ where $\pi^* = \max_{x\in\Omega}\sqrt{\frac{1-\pi(x)}{\pi(x)}}$.

Before we apply this Lemma to bound the mixing time of P, we state another Lemma

Lemma 6. *Let P^* be the lazy version of the Markov Chain P (by having a self transition with probability $\frac{1}{2}$ every time slot). Then $P^* \ge \alpha Q$, where $\alpha = \frac{1}{2}$.*

Proof. Denote the transition from $\mathbf{s}$ to $\mathbf{s}'$ as $P^*(\mathbf{s}, \mathbf{s}')$ and $Q(\mathbf{s}, \mathbf{s}')$ in the P^* and Q chains respectively. We verify that $\alpha = \frac{1}{2}$ for all three types of transitions (and we add type 4 for self-transitions)

T1 $P^*(\mathbf{s}, \mathbf{s}') = \frac{\lambda_v}{2n(1+\lambda_v)} \ge \frac{\lambda_v}{2n(1+\lambda_{\max})} \ge Q(\mathbf{s}, \mathbf{s}') \ge \alpha Q(\mathbf{s}, \mathbf{s}')$

T2 $P^*(\mathbf{s}, \mathbf{s}') = \frac{1}{2n(1+\lambda_v)} \ge \frac{1}{2n(1+\lambda_{\max})} \ge Q(\mathbf{s}, \mathbf{s}') \ge \alpha Q(\mathbf{s}, \mathbf{s}')$

T3 $P^*(\mathbf{s}, \mathbf{s}') = \frac{|K_v|-1}{2n} \frac{\lambda_w}{\sum_{z\in K_v}(1+\lambda_z)} \ge \frac{|K_v|-1}{2n} \frac{\lambda_w}{|K_v|(1+\lambda_{\max})} \ge \alpha Q(\mathbf{s}, \mathbf{s}')$

T4 $P^*(\mathbf{s}, \mathbf{s}') \ge \frac{1}{2} \ge \alpha \ge \alpha Q(\mathbf{s}, \mathbf{s}')$

□

We can now directly bound the mixing time of P using the following theorem:

Theorem 7. *Given the NB-CSMA Markov Chain P, if $\lambda_{\max} < \frac{1}{\max_v(d_v - |K_v|)}$, then the mixing time of P can be bounded as*

$$T_{mix}(\epsilon) = \mathcal{O}(n^2 \log n). \qquad (49)$$

Proof. The theorem is proved by applying Lemma 5 for P^* with $\alpha = \frac{1}{2}$ from Lemma 6. Also, for any graph, $\log(\pi^*) = \mathcal{O}(n)$ [7]. Applying Lemma 5 with these quantities, and noticing that P^* has double the mixing time of P concludes the theorem. □

It remains to find the fraction of the capacity region that causes the Markov chain P to be fast mixing, and consequently causes the queue lengths of links to be polynomially bounded in the number of links. Before we state our main theorem we state a related result from [13] as a Lemma

Lemma 8. *[13] Given any $\nu \in \Lambda$, there exists suitable activation rates λ such that for every link v, the mean service rate $\mathrm{E}(s_v)$ is equal to the mean arrival rate ν_v.*

The last Lemma ensures the existence of fixed activation rates (fugacities) that stabilize the queues whenever the arrival rate vector falls within the capacity region. Now we are ready to present our final theorem

Theorem 9. *Given an arrival rate vector ν that satisfies $\nu \in \gamma\Lambda$, where*

$$\gamma = \frac{1}{\max_v(d_v - |K_v| + 1)}, \qquad (50)$$

the Markov chain P is fast mixing, and the expected queue lengths of any link v can be bounded by

$$\mathrm{E}(q_v(t)) = \mathcal{O}(T_{mix}) = \mathcal{O}(n^2 log(n)) \qquad (51)$$

Proof sketch. We follow the approach of [12] to prove the theorem. It suffices to show that when $\nu \in \gamma\Lambda$, the activation rates that cause $\mathrm{E}(s_v) = \nu_v\ \forall v \in V$ satisfy $\lambda_v < \frac{1}{d_v - |K_v|}$ which implies fast-mixing by Theorem 7. Let p_{v0} be the probability that the medium as seen by link v is not blocked. It was shown in [12] that for the fugacities that make $\nu_v = \mathrm{E}(s_v)$ for every v, p_{v0} satisfies $\nu_v = \frac{\lambda_v}{1+\lambda_v} p_{v0}$. By the union bound we have

$$1 - p_{v0} \leq \sum_{j \in K_v} s_j + \sum_{k \in N_v \setminus K_v} s_k \leq \sum_{j \in K_v} \nu_j + \sum_{k \in N_v \setminus K_v} \nu_k \quad (52)$$

Also, note that $\nu' = \frac{1}{\gamma}\nu \in \Lambda$. Hence, there exists a scheduling algorithm which can serve ν'. Under this algorithm, we have $1 - \nu'_v$ as the fraction of time where link v is idle. During these idle slots, at most 1 link from K_v and $d_v - |K_v| + 1$ links from $N_v \setminus K_v$ are active. Thus

$$\sum_{j \in K_v} \nu'_j + \sum_{k \in N_v \setminus K_v} \nu'_k \leq (d_v - |K_v| + 2)(1 - \nu'_v). \quad (53)$$

Combining (52) and (53)

$$1 - p_{v0} \leq \sum_{j \in K_v} \nu_j + \sum_{k \in N_v \setminus K_v} \nu_k \leq \gamma(d_v - |K_v| + 2)(1 - \frac{\nu_v}{\gamma}) \leq (1 - \frac{\nu_v}{\gamma}) \quad (54)$$

Hence, $\nu_v \leq \gamma p_{v0}$ which implies $\frac{\lambda_v}{1+\lambda_v} \leq \gamma$. A direct substitution gives $\lambda_v \leq \frac{1}{d_v - |K_v|}$. $\square$

5.3 Discussion

In the last theorem, we characterized the fraction of the capacity region that makes the Markov chain P fast-mixing. This is the fraction of the capacity region for which the average queue lengths grow polynomially in the number of links n. The region in Theorem 6 is no smaller than the regions found in [12] and [24]. In fact the result in [12] showed that the network was fast-mixing for $\nu \in \frac{1}{\triangle}\Lambda$, where $\triangle = \max_v d_v$ is the conflict graph degree. The difference between this result and ours is the following: In [12] we have to decrease the throughput as the number of interferers increase to get acceptable delay performance. In Theorem 9, we have to decrease the throughput only when the number of external interferers increase (interferers that have a different transmitter). We expect the difference to be significant in ad-hoc networks when the average node degree increases as the network becomes dense.

Another difference is the following: The mixing time bound obtained in [12] was $\mathcal{O}(\log(n))$ under parallel-updates assumption (with assumption A2 removed). However the mixing time of Q-CSMA for single-site updates is lower bounded by $\Omega(n\log(n))$ [10]. Furthermore, in Theorem 9, an extra $\mathcal{O}(n)$ factor comes from the $\log(\pi^*)$ term in the comparison theorem in Lemma 5. Thus the mixing time upper bound of P is $\mathcal{O}(n)$ times larger than that of Q. However , as asserted in [7], this extra $\mathcal{O}(n)$ factor is almost certainly an artifact of the analysis. Thus, although the bound of Theorem 9 is $\mathcal{O}(n^2)$ times that of [12], we do not expect the mixing time of Q-CSMA to be less than that of NB-CSMA for any arrival rate vector. This will be further validated by simulations.

6. NUMERICAL RESULTS

We simulate a random topology where 20 wireless nodes are placed uniformly at random in a 600x600m square. A wireless link is established with probability 1, if the transmitter node is within 150m from the receiver node, and with probability 0.5 if the transmitter is within 250m from the receiver. All links in this simulation are unidirectional. We have a geometric interference relationship between links

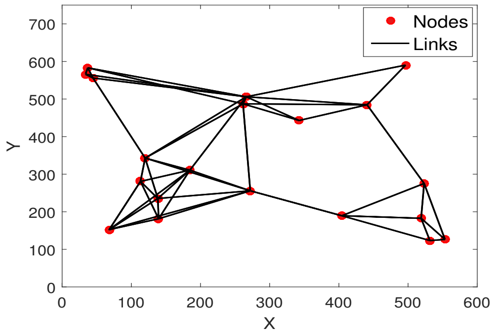


Figure 2: 600x600m Random Network Topology

whereby two links interfere with each other if a) they share a transmitter node, b) they share a receiver node, c) the transmitter node of one link is within 250 m of the receiver node of the other one. The resulting instance of the geometric random network has 48 links. To determine the decision schedule of Q-CSMA and the update blocks of the NB-CSMA, we use a contention window of size 8 and a random back-off scheme, where every link waits a random time, then attempts to include itself in the update blocks (decision schedule). If no conflict happens, the inclusion is successful. We also use the dynamic fugacities $\lambda_v = \frac{\log(1+q_v)}{\log(e+\log(1+q_v))}$ found to have the best delay performance [8]. Thus effectively in the simulation we have dropped assumptions (A1-A3) of Section 5. The arrivals at different links are independent Bernoulli processes. We determine the arrival rates of all links using the following three steps: a) We compute all the maximal independent sets of the conflict graph G of the network $\overline{G}$ in Fig. 2, we call this set of independent sets $\mathcal{A}_M$. b) For each of these sets, $\mathcal{A}_{m\in M}$, we obtain an arrival rate vector ν_m on the boundary of the capacity region by setting the arrival rates of all links v included in the set to $\{\nu_{vm} = 1, \ \forall v \in m\}$ and otherwise $\{\nu_{vm} = 0, \ \forall v \notin m\}$. c) By taking the average of ν_m that is, taking the average of the binary vector $[\nu_{vm}]_{m\in M}$ at each link v, we get an arrival rate vector ν^* at the boundary of the capacity region that has a strictly positive arrival rate for every link $v \in V$. We multiply ν^* by a factor $\rho \in (0,1)$ that we call **"Traffic Intensity"** to simulate the network at different levels of throughput. In Fig. 3, we plot the time-average queue lengths per link against the traffic intensity ρ. We calculate the average queue lengths for 2×10^5 time slots. In all simulations, we neglect the evolution of queue lengths for the first half of the simulation time when calculating time-averages, to make sure that we have minimized the influence of the transient behavior at the beginning of the simulation. We can see as expected that NB-CSMA outperforms Q-CSMA for all values of ρ. In this example, the average queue lengths of NB-CSMA is on average half of that of Q-CSMA for all values of ρ. Thus, for this example, NB-CSMA results in 50% decrease in average delay for any arrival rate vector within the capacity region.

7. CONCLUSION

In this paper, we have proposed a node-based CSMA algorithm that is throughput optimal and outperforms link-based CSMA. The improvement margin depends on network topology, but for practical ad-hoc networks, we expect the improvement to be significant. NB-CSMA is also fully dis-

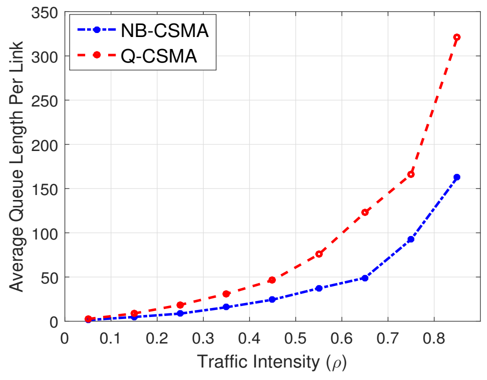


Figure 3: Average Queue Length per link vs. ρ

tributed, in the sense that the nodes make their decisions based solely on local information. Our mixing time analysis gives an insight on how topology affects the low delay fraction of the capacity region. In particular, we have shown that interferers that originate from the same transmitter do not contribute to the shrinkage of the fraction of the capacity region with low delays under NB-CSMA. Our simulation results show a significant improvement over link-based CSMA. In our future work, we will investigate the delay performance of NB-CSMA in multihop ad-hoc networks and characterize the effect of network density on delay.

8. ACKNOWLEDGMENTS

This work was supported in part by the NSF under Grants CNS-1421576 and IIA-1445542

9. REFERENCES

[1] D. Aldous and J. Fill. Reversible markov chains and random walks on graphs, 2002.

[2] D. P. Bertsekas, R. G. Gallager, and P. Humblet. *Data networks*, volume 2. Prentice-Hall International New Jersey, 1992.

[3] C. Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *Proceedings of the 3rd ACM MobiHoc*. ACM, 2002.

[4] N. Bouman, S. C. Borst, and J. van Leeuwaarden. Delay performance of backlog based random access. *ACM SIGMETRICS Performance Evaluation Review*, 39(2), 2011.

[5] R. Bubley and M. Dyer. Path coupling: A technique for proving rapid mixing in markov chains. In *Proceedings., 38th Annual Symposium on Foundations of Computer Science*. IEEE, 1997.

[6] P. Diaconis and L. Saloff-Coste. Comparison theorems for reversible markov chains. *The Annals of Applied Probability*, 1993.

[7] M. Dyer and C. Greenhill. On markov chains for independent sets. *Journal of Algorithms*, 35(1), 2000.

[8] J. Ghaderi and R. Srikant. On the design of efficient csma algorithms for wireless networks. In *49th IEEE CDC, 2010*. IEEE, 2010.

[9] J. Ghaderi and R. Srikant. Effect of access probabilities on the delay performance of q-csma algorithms. In *INFOCOM*. IEEE, 2012.

[10] T. P. Hayes and A. Sinclair. A general lower bound for mixing of single-site dynamics on graphs. In *46th FOCS*. IEEE, 2005.

[11] P.-K. Huang and X. Lin. Improving the delay performance of csma algorithms: A virtual multi-channel approach. In *INFOCOM*. IEEE, 2013.

[12] L. Jiang, M. Leconte, J. Ni, R. Srikant, and J. Walrand. Fast mixing of parallel glauber dynamics and low-delay csma scheduling. *IEEE Transactions on Information Theory*, 58(10), 2012.

[13] L. Jiang and J. Walrand. A distributed csma algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 18(3), 2010.

[14] J. Kwak, C.-H. Lee, et al. A high-order markov chain based scheduling algorithm for low delay in csma networks. In *INFOCOM*. IEEE, 2014.

[15] K.-K. Lam, C.-K. Chau, M. Chen, and S.-C. Liew. Mixing time and temporal starvation of general csma networks with multiple frequency agility. In *ISIT*. IEEE, 2012.

[16] C.-H. Lee, D. Y. Eun, S.-Y. Yun, and Y. Yi. From glauber dynamics to metropolis algorithm: Smaller delay in optimal csma. In *IEEE ISIT, 2012*. IEEE, 2012.

[17] D. Lee, D. Yun, J. Shin, Y. Yi, and S.-Y. Yun. Provable per-link delay-optimal csma for general wireless network topology. In *INFOCOM*. IEEE, 2014.

[18] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Soc., 2009.

[19] M. Lotfinezhad and P. Marbach. Throughput-optimal random access with order-optimal delay. In *INFOCOM*. IEEE, 2011.

[20] F. Martinelli. *Lectures on Glauber dynamics for discrete spin models*. Springer, 1999.

[21] 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. *IEEE/ACM Transactions onNetworking*, 20(3), 2012.

[22] S. Rajagopalan, D. Shah, and J. Shin. Network adiabatic theorem: an efficient randomized protocol for contention resolution. In *ACM SIGMETRICS Performance Evaluation Review*. ACM, 2009.

[23] D. Shah, D. N. Tse, and J. N. Tsitsiklis. Hardness of low delay network scheduling. *IEEE Transactions on Information Theory*, 2011.

[24] V. G. Subramanian and M. Alanyali. Delay performance of csma in networks with bounded degree conflict graphs. In *ISIT*, 2011.

[25] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12), 1992.

[26] D. Xue and E. Ekici. On reducing delay and temporal starvation of queue-length-based csma algorithms. In *Allerton*. IEEE, 2012.