Q-CSMA: Queue-Length Based CSMA/CA Algorithms for Achieving Maximum Throughput and Low Delay in Wireless Networks

Jian Ni, Bo (Rambo) Tan, and R. Srikant
University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
Email: {jianni, botan2, rsrikant}@illinois.edu

Abstract—Recently, it has been shown that CSMA-type random access algorithms can achieve the maximum possible throughput in ad hoc wireless networks. However, these algorithms assume an idealized continuous-time CSMA protocol where collisions can never occur. In addition, simulation results indicate that the delay performance of these algorithms can be quite bad. On the other hand, although some simple heuristics (such as distributed approximations of greedy maximal scheduling) can yield much better delay performance for a large set of arrival rates, they may only achieve a fraction of the capacity region in general. In this paper, we propose a discrete-time version of the CSMA algorithm. Central to our results is a discrete-time distributed randomized algorithm which is based on a generalization of the so-called Glauber dynamics from statistical physics, where multiple links are allowed to update their states in a single time slot. The algorithm generates collision-free transmission schedules while explicitly taking collisions into account during the control phase of the protocol, thus relaxing the perfect CSMA assumption. More importantly, the algorithm allows us to incorporate mechanisms which lead to very good delay performance while retaining the throughput-optimality property.

I. INTRODUCTION

For wireless networks with limited resources, efficient resource allocation and optimization play an important role in achieving high performance and providing satisfactory quality-of-service (QoS). In this paper, we study link scheduling (or Media Access Control, MAC) for wireless networks, where the links (node pairs) may not be able to transmit simultaneously due to transceiver constraints and radio interference. A scheduling algorithm (or MAC protocol) decides which links can transmit data at each time instant so that no two active links interfere with each other.

The performance metrics of interest in this paper are throughput and delay. The throughput performance of a scheduling algorithm is often characterized by the largest set of arrival rates under which the algorithm can keep the queues in the network stable. The delay performance of a scheduling algorithm can be characterized by the average delay experienced by the packets transmitted in the network. Since many wireless network applications have stringent bandwidth and delay requirements, designing high-performance scheduling algorithms to achieve maximum possible throughput and low delay is of great importance, which is the main objective of this paper. We also want the scheduling algorithms to be distributed and have low complexity/overhead, since in many wireless networks there is no centralized entity and the resources at the nodes are very limited.

It is well known that the queue-length based Maximum Weight Scheduling (MWS) algorithm is throughput-optimal [19], in the sense that it can stabilize the network queues for all arrival rates in the capacity region of the network (without explicitly knowing the arrival rates). However, for general interference models MWS requires the network to solve a complex combinatorial optimization problem in each time slot and hence, is not implementable in practice.

Maximal scheduling is a low-complexity alternative to MWS but it may only achieve a small fraction of the capacity region [3], [21]. Greedy Maximal Scheduling (GMS), also known as Longest-Queue-First (LQF) Scheduling, is another natural low-complexity alternative to MWS which has been observed to achieve very good throughput and delay performance in a variety of wireless network scenarios. It was shown in [4] that if the network satisfies the so-called local-pooling condition, then GMS is throughput-optimal; but for networks with general topology GMS may only achieve a fraction of the capacity region [8], [9], [22].

Another class of scheduling algorithms are CSMA (Carrier Sense Multiple Access) type random access algorithms. Under CSMA, a node (sender of a link) 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 backoff time. Since CSMA-type algorithms can be easily implemented in a distributed manner, they are widely used in practice (e.g., the IEEE 802.11 MAC protocol). In [1] the authors derived an analytical model to calculate the throughput of a CSMA-type algorithm in multi-hop wireless networks. They showed that the Markov chain describing the evolution of schedules has a product-form stationary distribution under an idealized continuous-time CSMA protocol (which assumes zero propagation/sensing delay and no hidden terminals) where collisions can never occur. This model was used in [20] to study throughput and fairness issues in wireless ad hoc networks. The insensitivity properties of such a CSMA algorithm have been recently studied in [10].
Based on the results in [1], [20], [10], a distributed algorithm was developed in [7] to adaptively choose the CSMA parameters to meet the traffic demand without explicitly knowing the arrival rates. The results in [7] make a time-scale separation assumption, whereby the CSMA Markov chain converges to its steady-state distribution instantaneously compared to the time-scale of adaptation of the CSMA parameters. Preliminary ideas for a related result was reported in [16], where the authors study distributed algorithms for optical networks. But it is clear that their model also applies to wireless networks with CSMA. In [17], a slightly modified version of the algorithm proposed in [16] was shown to be throughput-optimal. The key idea in [17] 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. While the results in our paper are most closely related to the works in [1], [7], [17], we also note important contributions in [2], [5], [12], [15] which make connections between random access algorithms and stochastic loss networks.

Although the recent results on CSMA-type random access algorithms show throughput-optimality, simulation results indicate that the delay performance of these algorithms can be quite bad and much worse than MWS and GMS. Thus, one of our goals in this paper is to design distributed scheduling algorithms that have low complexity, are provably throughput-optimal, and have good delay performance. Towards this end, we design a discrete-time version of the CSMA random access algorithm. The algorithm generates collision-free data transmission schedules while allowing for collisions during the control phase of the protocol (as in the 802.11 MAC protocol), thus relaxing the perfect CSMA assumption of the algorithms in [1], [7], [17]. More importantly, our formulation allows us to incorporate delay-reduction mechanisms in the choice of schedules while retaining the algorithm’s throughput-optimality property. It also allows us to resolve the hidden and exposed terminal problems associated with wireless networks.

We organize the paper as follows. In Section II we introduce the network model. In Section III we present the basic scheduling algorithm. In Section IV we develop a distributed implementation of the basic scheduling algorithm called Q-CSMA. In Section V we propose a hybrid Q-CSMA algorithm which combines Q-CSMA with a distributed approximation of GMS to achieve both maximum throughput and low delay. The paper is concluded in Section VI.

II. Network Model

We model a (single-channel) wireless network by a graph $G = (V, E)$, where $V$ is the set of nodes and $E$ is the set of links. Nodes are wireless transmitters/receivers. There exist a directed link $(n, m) \in E$ if node $m$ can hear the transmission of node $n$. We assume that if $(n, m) \in E$, then $(m, n) \in E$.

For any link $i \in E$, we use $\mathcal{C}(i)$ to denote the set of conflicting links (called conflict set) of $i$, i.e., $\mathcal{C}(i)$ is the set of links such that if any one of them is active, then link $i$ cannot be active. The conflict set $\mathcal{C}(i)$ may include links that share a common node with link $i$, and links that will cause interference to link $i$ when transmitting. We assume symmetry in the conflict set so that if $i \in \mathcal{C}(j)$ then $j \in \mathcal{C}(i)$.

We consider a time-slotted system. A feasible (collision-free) schedule of $G = (V, E)$ is a set of links that can be active at the same time according to the conflict set constraint, i.e., no two links in a feasible schedule conflict with each other. We assume that the links have unit capacity, i.e., an active link can transmit one packet in one time slot under a feasible schedule. Note that the results in this paper can be readily extended to networks with arbitrary link capacities.

A schedule is represented by a vector $x \in \{0, 1\}^{\mathcal{E}}$. The $i^{th}$ element of $x$ is equal to 1 (i.e., $x_i = 1$) if link $i$ is included in the schedule; $x_i = 0$ otherwise. With a little bit abuse of notation, we also treat $x$ as a set and write $i \in x$ if $x_i = 1$.

Note that a feasible schedule $x$ satisfies:

$$x_i + x_j \leq 1, \text{ for all } i \in E \text{ and } j \in \mathcal{C}(i). \quad (1)$$

Let $\mathcal{M}$ be the set of all feasible schedules of the network.

A scheduling algorithm is a procedure to decide which schedule to be used in every time slot for data transmission. In this paper we focus on the MAC layer so we only consider single-hop traffic. The capacity region of the network is the set of all arrival rates $\lambda$ for which there exists a scheduling algorithm that can stabilize the queues. It is known (e.g., [19]) that the capacity region is given by

$$\Lambda = \{ \lambda : \lambda \geq 0 \text{ and } \exists \mu \in Co(\mathcal{M}), \lambda < \mu \}. \quad (2)$$

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 $\Lambda$.

III. The Basic Scheduling Algorithm

We divide each time slot $t$ into a control slot and a data slot. The purpose of the control slot is to generate a collision-free transmission schedule $x(t) \in \mathcal{M}$ used for data transmission in the data slot. To achieve this, the network first selects a set of links that do not conflict with each other, denoted by $\mathfrak{m}(t)$. Note that these links also form a feasible schedule, but it is not the schedule used for data transmission. We call $\mathfrak{m}(t)$ the decision schedule in time slot $t$.

Let $\mathcal{M}_0 \subseteq \mathcal{M}$ be the set of possible decision schedules. The network selects a decision schedule according to a randomized procedure, i.e., it selects $\mathfrak{m}(t) \in \mathcal{M}_0$ with positive probability $\alpha(\mathfrak{m}(t))$, where $\sum_{\mathfrak{m}(t) \in \mathcal{M}_0} \alpha(\mathfrak{m}(t)) = 1$. Then, the transmission schedule is determined as follows. For any link $i$ in $\mathfrak{m}(t)$, if no links in $\mathcal{C}(i)$ were active in the previous data slot, then link $i$ is chosen to be active with an activation probability $p_i$ and inactive with probability $1 - p_i$ in the current data slot. If at least one link in $\mathcal{C}(i)$ was active in the previous data slot, then $i$ will be inactive in the current data slot. Any link not selected by $\mathfrak{m}(t)$ will maintain its state (active or inactive) from the previous data slot. Conditions on the set of decision
schedules $M_0$ and the link activation probabilities $p_i$'s will be specified later.

**Basic Scheduling Algorithm (in Time Slot $t$)**

1. In the control slot, randomly select a decision schedule $m(t) \in M_0$ with probability $\alpha(m(t))$.
   \[ \forall i \in m(t): \]
   - If no links in $C(i)$ were active in the previous data slot, i.e.,
     \[ \sum_{j \in C(i)} x_t(t - 1) = 0 \]
     (a) $x_t(t) = 1$ with probability $p_i$, $0 < p_i < 1$;
     (b) $x_t(t) = 0$ with probability $\bar{p}_i = 1 - p_i$.
   - Else
     (c) $x_t(t) = 0$. \[ \forall i \notin m(t): \]
     (d) $x_t(t) = x_t(t - 1)$.

2. In the data slot, use $x(t)$ as the transmission schedule.

Note that the algorithm is a generalization of the so-called Glauber dynamics from statistical physics [13], where multiple links are allowed to make decisions in a single time slot.

First we show that if the transmission schedule used in the previous data slot and the decision schedule selected in the current control slot both are feasible, then the transmission schedule generated in the current data slot is also feasible. For space limitation, the proofs of the lemmas, propositions, and theorems will be omitted, and can be found in [14].

**Lemma 1:** If $x(t - 1), m(t) \in M$, then $x(t) \in M$.

Because $x(t)$ only depends on the previous state $x(t - 1)$ and some randomly selected decision schedule $m(t)$, $x(t)$ evolves as a discrete-time Markov chain (DTMC). Next we derive the transition probabilities between the states.

**Lemma 2:** A state $x \in M$ can make a transition to a state $x' \in M$ if and only if $x \cup x' \in M$ and there exists a decision schedule $m \in M_0$ such that $x \triangle x' = (x \setminus x') \cup (x' \setminus x) \subseteq m$, and in this case the transition probability from $x$ to $x'$ is:

\[
P(x, x') = \sum_{m \in M_0} \alpha(m) \left( \prod_{i \in \Delta x'} \bar{p}_i \right) \left( \prod_{k \in \Delta x \setminus x} p_k \right) \left( \prod_{i \in m \setminus (x \triangle x')} p_i \right) \left( \prod_{j \in m \setminus (x \triangle x') \cup (x \setminus x')} \bar{p}_j \right).
\]

**Proposition 1:** A necessary and sufficient condition for the DTMC of the transmission schedules to be irreducible and aperiodic is $\cup_{m \in M_0} m = E$, and in this case the DTMC is reversible and has the following product-form stationary distribution:

\[
\pi(x) = \frac{1}{Z} \prod_{i \in x} p_i \frac{1}{\bar{p}_i},
\]

\[
Z = \sum_{x \in M} \prod_{i \in x} p_i \bar{p}_i.
\]

Based on the product-form distribution in Proposition 1, and by choosing the link activation probabilities as appropriate functions of the queue lengths, one can then proceed as in [7] (under a time-scale separation assumption) or as in [17] (without such an assumption) to establish throughput-optimality of the scheduling algorithm. In [14] we provide an alternative simple proof of throughput-optimality under the time-scale separation assumption in [7].

**Proposition 2:** Suppose the basic scheduling algorithm satisfies $\cup_{m \in M_0} m = E$ and hence has the product-form stationary distribution. Let $p_i = \frac{e^{x(t)}}{e^{x(t)} + 1}, \forall i \in E$, where $w_i(t)$ is appropriate functions of the queue lengths $q_i(t)$'s, e.g., $w_i(t) = \log(\beta q_i(t))$ for some small constant $\beta > 0$. Then the scheduling algorithm is throughput-optimal.

**IV. DISTRIBUTED IMPLEMENTATION: Q-CSMA**

In this section we present a distributed implementation of the basic scheduling algorithm. The key idea is to develop a distributed randomized procedure to select a (feasible) decision schedule in the control slot. To achieve this, we further divide the control slot into control mini-slots. Note that once a link knows whether it is included in the decision schedule, it can determine its state in the data slot based on its carrier sensing information (i.e., whether its conflicting links were active in the previous data slot) and link activation probability. We call this implementation Q-CSMA (Queue-length based CSMA/CA), since the activation probability of a link is determined by its queue length to achieve maximum throughput, and collisions of data packets are avoided via carrier sensing and the exchange of control messages.

At the beginning of each time slot, every link $i$ will select a random backoff time. Link $i$ will send a message announcing its INTENT to make a decision at the expiry of this backoff time subject to the constraints described below.

**Q-CSMA Algorithm (at Link $i$ in Time Slot $t$)**

1. Link $i$ selects a random (integer) backoff time $T_i$ uniformly in $[0, W - 1]$ and waits for $T_i$ control mini-slots.
2. If link $i$ hears an INTENT message from a link in $C(i)$ before the $(T_i + 1)$-th control mini-slot, $i$ will not be included in $m(t)$ and will not transmit an INTENT message anymore. Link $i$ will set $x_i(t) = x_i(t - 1)$.
3. If link $i$ does not hear an INTENT message from any link in $C(i)$ before the $(T_i + 1)$-th control mini-slot, it will send (broadcast) an INTENT message to all links in $C(i)$ at the beginning of the $(T_i + 1)$-th control mini-slot.
   - If there is a collision (i.e., if there is another link in $C(i)$ transmitting an INTENT message in the same mini-slot), link $i$ will not be included in $m(t)$ and will set $x_i(t) = x_i(t - 1)$.
   - If there is no collision, link $i$ will be included in $m(t)$ and decide its state as follows:
     - If there are no links in $C(i)$ were active in the previous data slot
       \[
x_i(t) = 1 \text{ with probability } p_i, \quad 0 < p_i < 1;
       \]
     \[
x_i(t) = 0 \text{ with probability } \bar{p}_i = 1 - p_i.
     \]
     - Else
       \[
x_i(t) = 0.
     \]
4. If $x_i(t) = 1$, link $i$ will transmit a packet in the data slot.
Lemma 3: $\mathbf{w}(t)$ produced by Q-CSMA is a feasible schedule. Let $\mathcal{M}_0$ be the set of all decision schedules produced by Q-CSMA. If the window size $W \geq 2$, then $\cup_{\mathbf{w} \in \mathcal{M}_0} \mathbf{w} = E$.

Combining Lemma 3 and Propositions 1, 2 we have the following result.

Proposition 3: Q-CSMA has the product-form distribution given in Proposition 1 if $W \geq 2$. Further, it is throughput-optimal if we let $p_i = \frac{e^{w_i(t)}}{e^{w_i(t)} + 1}, \forall i \in E$, where $w_i(t)$s are appropriate functions of the queue lengths.

Remark 1: A control slot of Q-CSMA consists of $W$ mini-slots and each link needs to send at most one INTENT message. Hence Q-CSMA has constant (and low) signalling/time overhead, independent of the size of the network. Suppose the duration of a data slot is $D$ mini-slots. Taking control overhead into account, Q-CSMA can achieve $\frac{D}{D+W}$ of the capacity region, which approaches the full capacity when $W \ll D$.

When describing the Q-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 the nodes in the network. In [14] we provide details to implement Q-CSMA based on the nodes. Such an implementation also allows us to resolve the hidden and exposed terminal problems associated with wireless networks [14].

V. A LOW-DELAY HYBRID Q-CSMA ALGORITHM

By Little’s law, the long-term average queuing delay experienced by the packets is proportional to the long-term average queue length in the network. In our simulations (see [14]) we find that the delay performance of Q-CSMA can be quite bad and much worse than greedy maximal scheduling GMS (this is also true in simulations of the continuous-time CSMA algorithm). However, GMS is a centralized algorithm and is not throughput-optimal in general. We are therefore motivated to design a distributed scheduling algorithm that can combine the advantages of both Q-CSMA (for achieving maximum throughput) and GMS (for achieving low delay). We first develop a distributed algorithm to approximate GMS, which we call D-GMS.

The basic idea of D-GMS is to assign smaller backoff times to links with larger queue lengths. However, to handle cases where two or more links in a neighborhood have the same queue length, some collision resolution mechanism is incorporated in D-GMS. Further, we have conducted extensive simulations to understand how to reduce the control overhead required to implement D-GMS while maintaining the ability to control the network when the queue lengths become large. Based on these simulations, we conclude that it is better to use the log of the queue lengths (rather than the queue lengths themselves) to determine the channel access priority of the links. The resulting D-GMS algorithm is described below.

### D-GMS Algorithm (at Link $i$ in Time Slot $t$)

1. Link $i$ selects a random backoff time $T_i = W \times \lfloor B - \log_b(q_i(t) + 1) \rfloor + \text{Uniform}[0, W - 1]$ and waits for $T_i$ control mini-slots.
2. If link $i$ hears an RESV message (e.g., an RTS/CTS pair) from a link in $C(i)$ before the $(T_i+1)$-th control mini-slot, it will not be included in $x(t)$ and will not transmit an RESV message. Link $i$ will set $x_i(t) = 0$.
3. If link $i$ does not hear an RESV message from any link in $C(i)$ before the $(T_i+1)$-th control mini-slot, it will send an RESV message to all links in $C(i)$ at the beginning of the $(T_i+1)$-th control mini-slot.
   - If there is a collision, link $i$ will set $x_i(t) = 0$.
   - If there is no collision, link $i$ will set $x_i(t) = 1$.
4. IF $x_i(t) = 1$, link $i$ will transmit a packet in the data slot.

Remark 2: In the above algorithm, each control slot can be thought as $B$ frames, with each frame consisting of $W$ mini-slots. Links are assigned a frame based on the log of their queue lengths and the $W$ mini-slots within a frame are used to resolve contentions among links in the same frame. Hence a control slot of D-GMS consists of $W \times B$ mini-slots, and links with empty queues will not compete for the channel in this time slot.

Now we are ready to present a hybrid Q-CSMA algorithm which is both provably throughput-optimal and has very good delay performance in simulations. The basic idea behind the algorithm is as follows. For links with weight greater than a threshold $w_0$, the Q-CSMA procedure is applied first to determine their states; for other links, the D-GMS procedure is applied next to determine their states. To achieve this, a control slot is divided into $W_0$ mini-slots which are used to perform Q-CSMA for links whose weight is greater than $w_0$ and $W_1 \times B$ mini-slots which are used to implement D-GMS among the other links. Each link $i$ uses a 1-bit memory $NA_i$ to record whether any of its conflicting links becomes active due to the Q-CSMA procedure in a time slot. This information is used in constructing a schedule in the next time slot.

### Hybrid Q-CSMA Algorithm (at Link $i$ in Time Slot $t$)

**IF** $w_i(t) > w_0$ (Q-CSMA Procedure)

1. **Link $i$ selects a random backoff time** $T_i = \text{Uniform}[0, W_0 - 1]$.
2. **If link $i$ hears an INTENT message from a link in $C(i)$ before the $(T_i+1)$-th control mini-slot, then it will set $x_i(t) = x_i(t-1)$ and go to Step 1.4.**
3. **If link $i$ does not hear an INTENT message from any link in $C(i)$ before the $(T_i+1)$-th control mini-slot, it will send an INTENT message to all links in $C(i)$ at the beginning of the $(T_i+1)$-th control mini-slot.**
   - If there is a collision, link $i$ will set $x_i(t) = x_i(t-1)$.
   - If there is no collision, link $i$ will decide its state as follows:
     - **If no links in $C(i)$ were active due to the Q-CSMA procedure in the previous data slot, i.e., $NA_i = 0$**
       - $x_i(t) = 1$ with probability $p_i$, $0 < p_i < 1$;
       - $x_i(t) = 0$ with probability $p_i = 1 - p_i$.**
w_i(t) = 0. If link \( i \) hears an RESV message from any link in \( C(i) \) in the \((W_0+1)\)-th control mini-slot, it will set \( N_A_i = 0 \) and keep silent in this time slot. Otherwise, link \( i \) will set \( N_A_i = 0 \) and select a random backoff time \( T_i = (W_0 + 1) + W_1 \times [B - \log_2(q_i(t) + 1)]^+ + \text{Uniform}[0, W_1 - 1] \) and wait for \( T_i \) control mini-slots.

2.2 If link \( i \) hears an RESV message from a link in \( C(i) \) before the \((T_i+1)\)-th control mini-slot, it will set \( x_i(t) = 0 \) and keep silent in this time slot.

2.3 If link \( i \) does not hear an RESV message from any link in \( C(i) \) before the \((T_i+1)\)-th control mini-slot, it will send an RESV message to all links in \( C(i) \) at the beginning of the \((T_i+1)\)-th control mini-slot.

- If there is a collision, link \( i \) will set \( x_i(t) = 0 \).
- If there is no collision, link \( i \) will set \( x_i(t) = 1 \).

2.4 If \( x_i(t) = 1 \), link \( i \) will transmit a packet in the data slot.

**Remark 3:** The \((W_0+1)\)-th control mini-slot (which occurs between the first \( W_0 \) mini-slots and the last \( W_1 \times B \) mini-slots) is reserved for all the links which have not been scheduled so far to conduct carrier sensing. In this mini-slot those links which have already been scheduled (due to the Q-CSMA procedure) will send an RESV message so their neighbors can sense and record this information in their \( N_A \) bit.

**Remark 4:** The control overhead of the hybrid Q-CSMA algorithm is \( W_0 + 1 + W_1 \times B \) per time slot. As in the pure D-GMS algorithm, links with empty queues will keep silent throughout the time slot.

**Proposition 4:** For each link \( i \) with weight \( w_i(t) > w_0 \), if we choose its activation probability \( p_i = \frac{w_i(t)}{\sum_{j=1}^{n} w_j(t)} \), where \( w_i(t) \)s are appropriate functions of the queue lengths, then the hybrid Q-CSMA algorithm is throughput-optimal.

In the above algorithm, one can replace D-GMS by any other heuristic and still maintain throughput-optimality. We simply use D-GMS because it is an approximation to GMS which is known to perform well in previous simulation studies.

A simulation study of different scheduling algorithms including MWS, GMS, D-GMS, Q-CSMA, and the hybrid Q-CSMA algorithm can be found in [14].

VI. CONCLUSION

In this paper, we proposed a discrete-time distributed queue-length based CSMA/CA protocol that leads to collision-free data transmission schedules. The protocol is provably throughput-optimal. The discrete-time formulation allows us to incorporate mechanisms to dramatically reduce the delay without affecting the theoretical throughput-optimality property.

In particular, combining CSMA with distributed GMS leads to very good delay performance. We believe that it should be straightforward to extend our algorithms to be applicable to networks with multi-hop traffic and congestion-controlled sources (see [11], [6], [18] for related surveys).

**REFERENCES**


