Approaching Throughput-optimality in a Distributed CSMA Algorithm with Contention Resolution

Libin Jiang
Jean Walrand

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-37
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-37.html

March 6, 2009

Acknowledgement

This research was supported in part by MURI grant BAA 07-036.18.

Approaching Throughput-optimality in a Distributed CSMA Algorithm with Contention Resolution

Libin Jiang and Jean Walrand
EECS Department, University of California at Berkeley
{ljiang,wlr}@eecs.berkeley.edu

***Abstract*—It was shown recently that CSMA (Carrier Sense Multiple Access)-like distributed algorithms can achieve the maximal throughput in wireless networks (and task processing networks) under certain assumptions. One key assumption is that the sensing time is negligible, so that there is no collision. In this paper, we remove this idealized assumption by studying CSMA-based scheduling algorithms with collisions. First, we provide a model and give an explicit throughput formula which takes into account the cost of contention resolution. The formula has a simple form due to the quasi-reversibility structure of the model. Second, we show that the algorithms in [15] can be extended to approach throughput optimality in this case. Finally, sufficient conditions are given to ensure the convergence and stability of the proposed algorithms.**

***Index Terms*—Distributed scheduling, CSMA, Markov chain, convex optimization**

I. Introduction

Efficient resource allocation is essential to achieve high utilization of a class of networks with resource-sharing constraints, such as wireless networks and stochastic processing networks (SPN [3]). In wireless networks, certain links can not transmit at the same time due to the interference constraints among them. In a task processing problem in SPN, two tasks can not be processed simultaneously if they both require monopolizing a common resource. A scheduling algorithm determines which link to activate (or which task to process) at a given time without violating these constraints. Designing efficient distributed scheduling algorithms to achieve high throughput is especially a challenging task [1].

It is well known that maximal-weight scheduling (MWS) [6] is *throughput-optimal*. That is, MWS can stabilize all queues in the network as long as the incoming rates are within the capacity region. In MWS, time is assumed to be slotted. In each slot, a set of non-conflicting links (called an "Independent Set", or "IS") that have the maximal weight are scheduled, where the "weight" of a set of links is the summation of their queue lengths. However, finding such a maximal-weighted IS is NP-complete in general and is hard even for centralized algorithms. So its distributed implementation is not easy in wireless networks.

Acknowledgement: This research was supported in part by MURI grant BAA 07-036.18.

On the other hand, there has been active research on low-complexity but suboptimal scheduling algorithms. For example, reference [2] shows that a distributed greedy protocol similar to IEEE 802.11 (which is related to Maximal Scheduling [4]) can only guarantee a fraction of the network capacity (after ignoring collisions). Longest-Queue-First (LQF) algorithm (see, for example, [7], [8]), which greedily schedules queues in the descending order of the queue lengths, often performs better than Maximal Scheduling, although it is not throughput-optimal in general [8]. Reference [5] proposed random-access-based algorithms that can achieve performance comparable to that of maximal-size scheduling.

Recently, we proposed a distributed adaptive CSMA (Carrier Sensing Multiple Access) algorithm [15] that is throughput-optimal for a general interference model, under certain assumptions (further explained below). The algorithm has a few desirable features. It is distributed (i.e., each node only uses its own backlog information), asynchronous (i.e., nodes do not need to synchronize their transmission to avoid collisions) and requires no control message. (In [20], Rajagopalan and Shah independently proposed a similar randomized algorithm in the context of optical networks.) We have also developed a joint algorithm in [15] that combined the adaptive CSMA scheduling with end-to-end flow control to approach the maximal total utility of competing data flows. In [16], the optimality of these algorithms is proved formally.

However, the algorithms in [15], [16] have assumed perfect CSMA in the sense that the sensing is immediate such that two or more links cannot try to transmit at the same time (i.e., collisions are avoided). In many situations, however, this is an idealized assumption. For example, in CSMA/CA wireless networks, due to the propagation delay, sensing is not immediate. Instead, time can viewed as divided into discrete minislots, and collisions happen if multiple links try to transmit at the same minislot. When a collision occurs, all links that are involved lose their packets, and they will try again later. As another example, in a task processing network, if the requests for resources are initiated in discrete slots, then there is also an issue of collision and contention resolution.

In this paper, we study this important practical issue when designing high-performance CSMA-based scheduling algorithms. First, we present a model and give an explicit throughput formula which takes into account the cost of

contention resolution (in section II). The formula has a simple form due to the quasi-reversible structure of the model. Second, we show that the algorithms in [15], [16] can be extended to approach throughput optimality even with collisions (section III). Finally, sufficient conditions are given to ensure the convergence of the proposed algorithms.

In a related work [21], Ni and Srikant proposed a CSMA-like algorithm to achieve near-optimal throughput with collisions taken into account. The algorithm in [21] uses synchronized and alternate control phase and data phase. In the control phase, the nodes (or links) in the network send control packets in a randomized way, and then the schedule in the following data phase is decided by the result of the control phase and the schedule of the previous data phase. Comparably speaking, the algorithm we will study here is quite different in that it is asynchronous, and has more resemblance to the RTS/CTS mode in IEEE 802.11.

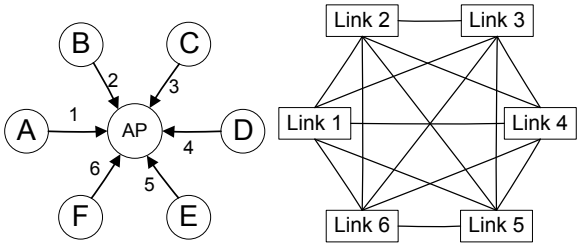
II. Basic model and the throughput formula

A. Basic Model

In this section we present a model for *CSMA/CA-based scheduling with contention resolution*. Note that the goal of the paper is *not* to propose a comprehensive model for IEEE 802.11 networks and predict the performance of such networks (The literature in that area has been very rich. See, for example, [10], [12] and the references therein.) Instead, at a more abstract level, we are interested in a distributed scheduling algorithm that is inspired by CSMA/CA, and designing adaptive algorithms to approach throughput-optimality. Such algorithms have applications not only in wireless networks, but also in a general task processing problem. (Some simplifying assumptions will be made in the next subsection compared to IEEE 802.11 networks.)

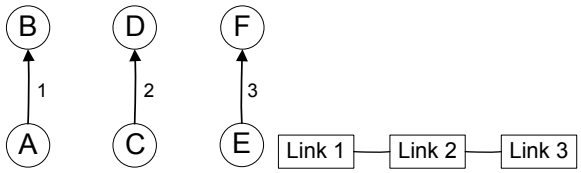
1) A model in the context of CSMA/CA wireless networks: Consider a (single-channel) wireless network. Define a "link" as an (ordered) transmitter-receiver pair. Assume that there are K links, and denote the set of links by $\mathcal{N}$ (then, $K = |\mathcal{N}|$). Without loss of generality, assume that each link has a capacity of 1. We say that two links *conflict* if they cannot transmit (or, "be active") at the same time due to interference. (The conflict relationship is assumed to be symmetric.) Accordingly, define G as the conflict graph. Each vertex in G represents a link, and there is an edge between two vertexes if the corresponding links conflict. Denote $e(i, j) = 1$ if there is an edge between link i and j. (Note that this simple conflict model may not reflect all possible interference that could occur in wireless networks. However, it does provide a useful abstraction and has been used widely in literature. See, for example, [12], [1] and the references therein. Also, it is general enough to be used in other resource sharing problems. An example is provided in the next section.) Further assume that there is no loss of packets due to channel fading.

For example, Fig. 1 (a) shows a wireless LAN with 6 links. Assume that all links can sense the transmissions of each other, then the network's conflict graph is a full graph (Fig. 1 (b)). (We use circles to represent nodes and squares to represent links.) Fig. 2 (a) shows an ad-hoc network with 3 links. Assume that link 1, 2 conflict, and link 2, 3 conflict. Then the network's conflict graph is Fig. 2 (b).



(a) Network topology (b) Conflict graph

Fig. 1: Infrastructure network



(a) Network topology (b) Conflict graph

Fig. 2: Ad-hoc network

Basic Protocol

Next, we describe the basic CSMA/CA protocol with fixed transmission probabilities. (Using fixed transmission probabilities suffices for our later development.) Let σ be the length of each idle slot (or "minislot"). (In 802.11a, for example, $\sigma = 9\mu s$.) In the following we will simply use "slot" to refer to the minislot.

Assume that all links are saturated (i.e., always have packets to transmit). In each slot, if (the transmitter of) link i is not already transmitting and if the medium is idle, the transmitter of link i starts transmission with probability p_i (also denote $q_i := 1 - p_i$). If at a certain slot, link i did not choose to transmit but a conflicting link starts transmitting, then link i keeps silent until that transmission ends. If they transmit at the same time, then a collision happens.

In this paper, we focus on networks without hidden nodes, i.e., a link can sense the transmission from any other link that can collide with its transmission. So if a collision occurs, it must be that multiple conflicting links starting transmitting at the same slot. (In [19], we further discuss the case with hidden nodes.)

Assume that each link transmits a short probe packet with length γ before the data is transmitted. This is similar to the RTS/CTS mode in 802.11. Sending a short probe packet before the actual data can avoid collisions of long data packets. (RTS/CTS in 802.11, however, is originally designed to reduce the hidden-node and exposed-node problems [13].) On the other hand, the probe packet increases the overhead of successful transmissions. When a collision happens, only the probe packets collide, so each collision lasts a length of γ (we also call γ the "collision length"). Assume that a successful transmission of link i lasts τ_i (which includes

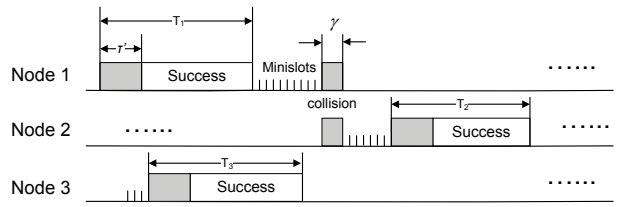


Fig. 3: Timeline in the basic model (In this figure, $\tau_i = T_i, i = 1, 2, 3$ are constants.)

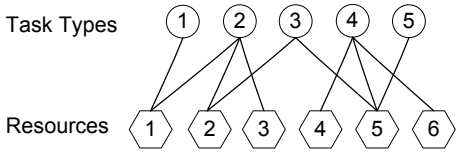


Fig. 4: Task processing

a constant overhead τ' and the data payload τ_i^p which is a random variable). Clearly $\tau_i \geq \tau'$. Let the p.m.f. (probability mass function) of τ_i be $Pr\{\tau_i = b_i\} = P_i(b_i), b_i \in \mathcal{Z}_{++}$, and the mean of τ_i be $T_i = \sum_{b_i} b_i P_i(b_i)$. Fig. 3 illustrates the timeline of the example 3-link network in Fig. 2.[1]

The above model possesses a quasi-reversibility property that leads to a particularly simple throughput formula, as will be shown later. A process is "time-reversible" if the process and its time-reversed process is statistically indistinguishable [14]. Our model, in Fig. 3, reversed in time follows the same protocol as described above, except for the order of the overhead and the payload which are reversed. A key reason for this property is that the collisions start and finish at the same time.

2) A similar model for the task processing problem: The above model can also be applied to a general task processing problem. Assume that there are K different types of tasks and a finite set of resources $\mathcal{B}$. To perform a type-k task, one needs a subset $\mathcal{B}_k \subseteq \mathcal{B}$ of resources and these resources are then monopolized by the task while it is being performed. Note that two tasks cannot be performed simultaneously iff they require some common resource. Clearly, this conflict relationship can be represented by a conflict graph as in the last section. That is, $e(i, j) = 1$ iff $\mathcal{B}_i \cap \mathcal{B}_j \neq \emptyset$.

The relationship between tasks and resources can be illustrated by a bipartite graph (for example, Fig. 4. Here, each circle is a task type, and each hexagon is a resource. There is an edge that connects a task type and each resource it requires. In Fig. 4, for example, task 2 requires resource 1, 2 and 3. So when task 2 is ongoing, task 1, 3 cannot be performed.

Associate a "link" to each type of tasks. Type i tasks arrive at link i with a certain rate, and are queued before being processed. Assume that the links access the resources in the same way as the above CSMA/CA protocol: if link i finds that the resources in the subset $\mathcal{B}_i$ are idle, then it sends a request (or a probe packet) in the current slot with probability p_i. If any other conflicting link sends the request in the same slot, then it is considered as a collision (possibly indicated by NACKs, i.e., negative acknowledgements, sent back by the resources where the conflicts occur) and they will try again later. Otherwise, link i proceeds to perform the task using the resources in subset $\mathcal{B}_i$.

[1]In [22], a similar model for CSMA/CA network is formulated with analogy to a loss network [11]. However, unlike this paper, reference [22] did not give an expression of the stationary distribution, partly because of a complex arrival process assumed there. In this paper, however, considering saturated arrivals suffices for our purpose in approaching the maximal throughput (section III), since we will let a link transmit dummy packets even if its queue is empty.

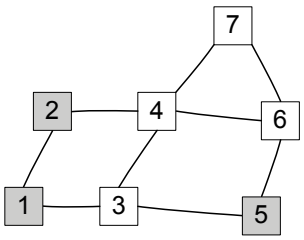


Fig. 5: Example: conflict graph and on-off states. Links 1, 2, 5 are on.

We shall use the language of wireless networks in the rest of the paper. But it is important to note that all results below apply to the more general task processing problem.

B. Notation

Let the "on-off state" be $x \in \{0, 1\}^K$, and x_k be the k'th element of x. Define $x_k = 1$ if the k'th link is active (transmitting) in state x, and $x_k = 0$ otherwise. Then x is a vector indicating which links are active at a given time.

Let $G(x)$ be the subgraph of G after removing all vertexes (each representing a link) with state 0 (i.e., any link j with $x_j = 0$) and their associated edges. In general, $G(x)$ is composed of a number of connected components (or simply called "components") $C_m(x), m = 1, 2, \ldots, M(x)$ where $M(x)$ is the total number of components in $G(x)$. If a component $C_m(x)$ has only one active link (i.e., $|C_m(x)| = 1$), then this link is having a successful transmission; if $|C_m(x)| > 1$, then all the links in the component are experiencing a collision. Let the set of "success" links in state x be $S(x) := \{k | k \in C_m(x)$ with $|C_m(x)| = 1\}$. And define the "collision number" $h(x)$ as the number of components in $G(x)$ with size larger than 1. That is, $h(x) := |\mathcal{N}(x)|$ where the index set $\mathcal{N}(x) := \{m \mid |C_m(x)| > 1\}$. Denote the set of links which are experiencing collisions as $\phi(x) := \cup_{m \in \mathcal{N}(x)} C_m(x)$.

For example, Fig. 5 shows a conflict graph. The set of links $\mathcal{N} = \{1, 2, \ldots, 7\}$. At state x, assume that link 1, 2 and 5 are active. Then there are two connected component in $G(x)$: $C_1(x) = \{1, 2\}, C_2(x) = \{5\}$. Since $|C_1(x)| = 2$, link 1 and 2 are experiencing a collision. Since $|C_2(x)| = 1$, link 5 is having a successful transmission. So $S(x) = \{5\}$, $\phi(x) = \{1, 2\}$ and the collision number $h(x) = 1$.

C. Stationary distribution and throughput computation with random packet sizes

Define the state

$$w := \{x, ((b_k, a_k), \forall k : x_k = 1)\} \tag{1}$$

where b_k is the total length of the current packet link k is transmitting, a_k is the remaining time (including the current slot) before the transmission of link k ends. Note that $a_k \leq b_k$. If $k \in \phi(x)$, then $b_k = \gamma$ and $a_k \in \{1, 2, \ldots, \gamma\}$. An important observation here is that the transmissions in a collision component $C_m(x)$ is "synchronized", i.e., the links in $C_m(x)$ must have started transmitting at the same time, and will end transmitting at the same time, so all links in the component $C_m(x)$ have the same remaining time. That is, $a_k = a^{(m)}$ for any $k \in C_m(x)$ where $|C_m(x)| > 1$, and $a^{(m)}$ denotes the remaining time of the component $C_m(x)$. (To see this, any two links i and j in this component with an edge between them must have started transmitting at the same time. Otherwise, if i starts earlier, j would not transmit since it already hears i's transmission; and vice versa. By induction, all links in the component must have started transmitting at the same time.)

The transitions among the set of states defined in (1) form a discrete-time Markov chain. Its stationary distribution is expressed in the following theorem.

Theorem 1: In the stationary distribution, the probability of the state w as defined by (1) is

$$p(w) = \frac{1}{E} \prod_{i:x_i=0} q_i \prod_{j:x_j=1} [p_j \cdot f(b_j, j, x)] \tag{2}$$

where

$$f(b_j, j, x) = \begin{cases} I(b_j = \gamma) & \text{if } j \in \phi(x) \\ P_j(b_j) & \text{if } j \in S(x) \end{cases}, \tag{3}$$

and E is a normalizing term such that $\sum_w p(w) = 1$, i.e., all probabilities sum up to 1. Note that $p(w)$ does not depend on the remaining time a_k's.

Proof: For a given state $w = \{x, ((b_k, a_k), \forall k : x_k = 1)\}$, define the set of active links whose remaining time is larger than 1 as

$$A_1(w) = \{k | x_k = 1, a_k > 1\}.$$

Links in $A_1(w)$ will continue their transmissions (either with success or a collision) in the next slot.

Define the set of inactive links "blocked" by links in $A_1(w)$ as

$$\partial A_1(w) = \{j | e(j,k) = 1 \text{ for some } k \in A_1(w)\}.$$

Links in $\partial A_1(w)$ will remain inactive in the next slot. Write $\bar{A}_1(w) := A_1(w) \cup \partial A_1(w)$. Define the set of all other links as

$$A_2(w) = \mathcal{N} \backslash \bar{A}_1(w).$$

These links can change their on-off states x_k's in the next slot. On the other hand, links in $\bar{A}_1(w)$ will have the same on-off states x_k's in the next slot.

State w can transit in the next slot to another state $w' = \{x', ((b'_k, a'_k), \forall k : x'_k = 1)\}$, i.e., $Q(w, w') > 0$, if and only if w' satisfies that (i) $x'_k = x_k, \forall k \in \bar{A}_1(w)$; (ii) $b'_k = b_k, a'_k = a_k - 1, \forall k \in \bar{A}_1(w)$ such that $x_k = 1$; (iii) $a'_k = b'_k, \forall k \in A_2(w)$ such that $x'_k = 1$, and $b'_k = \gamma, \forall k \in A_2(w) \cap \phi(x')$. (If $A_2(w)$ is an empty set, then condition (iii) is trivially true.) The transition probability is

$$Q(w, w') = \prod_{i \in A_2(w)} [p_i \cdot f(b'_i, i, x')]^{x'_i} q_i^{1-x'_i}.$$

Define

$$\tilde{Q}(w', w) := \prod_{i \in A_2(w)} [p_i \cdot f(b_i, i, x)]^{x_i} q_i^{1-x_i}.$$

(If $A_2(w)$ is an empty set, then $Q(w, w') = 1$ and $\tilde{Q}(w', w) := 1$.) If w and w' does not satisfy conditions (i), (ii), (iii), then $Q(w, w') = 0$, and also define $\tilde{Q}(w', w) = 0$. ($\tilde{Q}(w', w)$ can be viewed as the transition probability of the time-reversed process: notice the similarity between $Q(w, w')$ and $\tilde{Q}(w', w)$.)

Then, if $Q(w, w') > 0$ (and $\tilde{Q}(w', w) > 0$), $p(w)/\tilde{Q}(w', w) = \frac{1}{E} \prod_{i \notin A_2(w)} [p_i \cdot f(b_i, i, x)]^{x_i} q_i^{1-x_i}$. And $p(w')/Q(w, w') = \frac{1}{E} \prod_{i \notin A_2(w)} [p_i \cdot f(b'_i, i, x')]^{x'_i} q_i^{1-x'_i}$. But for any $i \notin A_2(w)$, i.e., $i \in \bar{A}_1(w)$, we have $x'_i = x_i, b'_i = b_i$ by condition (i), (ii) above. Therefore, the two expressions are equal. Thus

$$p(w) Q(w, w') = p(w') \tilde{Q}(w', w), \forall w, w'.$$

Therefore,

$$\sum_w p(w) \cdot Q(w, w') = \sum_w p(w') \cdot \tilde{Q}(w', w) = p(w').$$

That is, the distribution (2) is invariant (or "stationary"). ■

Using Theorem 1, the probability of any on-off state x can be computed by summing up the probabilities of all states w's with the same on-off state x, using (2).

Theorem 2: With the stationary distribution, the probability of $x \in \{0, 1\}^K$ is

$$\begin{aligned} p(x) &= \frac{1}{E} (\gamma^{h(x)} \prod_{k \in S(x)} T_k) \prod_{i:x_i=0} (1 - p_i) \prod_{j:x_j=1} p_j \\ &= \frac{1}{E} (\gamma^{h(x)} \prod_{k \in S(x)} T_k) \prod_{i \in \mathcal{N}} p_i^{x_i} q_i^{1-x_i} \end{aligned} \tag{4}$$

where $T_i = \sum_{b_k} [b_k \cdot P_k(b_k)]$ is the mean transmission time of link k, and E is a normalizing term such that $\sum_x p(x) = 1$.

Proof: Define the set of states $\mathcal{B}(x) := \{w |$ the on-off state is x in the state $w\}$. By Theorem

1, we have

$$\begin{aligned} p(x) &= \sum_{w\in\mathcal{B}(x)} p(w) \\ &= \frac{1}{E}\sum_{w\in\mathcal{B}(x)}\{\prod_{i:x_i=0} q_i \prod_{j:x_j=1}[p_j\cdot f(b_j,j,x)]\} \\ &= \frac{1}{E}(\prod_{i:x_i=0} q_i \prod_{j:x_j=1} p_j)\sum_{w\in\mathcal{B}(x)}\prod_{j:x_j=1} f(b_j,j,x) \\ &= \frac{1}{E}(\prod_{i:x_i=0} q_i \prod_{j:x_j=1} p_j)\cdot \\ &\quad \sum_{w\in\mathcal{B}(x)}[\prod_{j\in S(x)} P_j(b_j)\prod_{j\in\phi(x)} I(b_j=\gamma)] \end{aligned} \quad (5)$$

In a valid state w, if $j\in\phi(x)$, then $b_j=\gamma$. Therefore if w is valid, $\prod_{j\in\phi(x)} I(b_j=\gamma)=1$. Recall the notation $\mathcal{N}(x)$ as the index set of the collision components, and $a^{(m)}$ as the remaining time of the collision component $C_m(x)$. Then

$$\begin{aligned} &\sum_{w\in\mathcal{B}(x)}[\prod_{j\in S(x)} P_j(b_j)\prod_{j\in\phi(x)} I(b_j=\gamma)] \\ =& \sum_{w\in\mathcal{B}(x)}[\prod_{j\in S(x)} P_j(b_j)\prod_{m\in\mathcal{N}(x)} 1] \\ =& \prod_{j\in S(x)}[\sum_{b_j}\sum_{1\le a_j\le b_j} P_j(b_j)]\prod_{m\in\mathcal{N}(x)}(\sum_{1\le a^{(m)}\le\gamma} 1) \\ =& \prod_{j\in S(x)}[\sum_{b_j} b_j P_j(b_j)]\cdot\gamma^{h(x)} \\ =& (\prod_{k\in S(x)} T_k)\gamma^{h(x)} \end{aligned} \quad (6)$$

Combining (5) and (6) completes the proof. ■

After $p(x)$ is computed, the probability that link k is transmitting payload in a given slot is

$$s_k = (1-\frac{\tau'}{T_k})\sum_{x:k\in S(x)} p(x) \quad (7)$$

where τ' is the overhead of a successful transmission (e.g., RTS, CTS, ACK packets). Without loss of generality, assume that the capacity of each link is 1. So $s_k\in[0,1]$ is also the *throughput* of link k (normalized by the link capacity).

D. A useful "detailed state" and its stationary distribution

By Theorem 2,

$$p(x)\propto(\prod_{k\in S(x)} T_k)\gamma^{h(x)}\prod_{i\in\mathcal{N}} p_i^{x_i}q_i^{1-x_i} = g(x)\cdot(\prod_{k\in S(x)} T_k)$$

where $g(x)=\gamma^{h(x)}\prod_{i\in\mathcal{N}} p_i^{x_i}q_i^{1-x_i}$ is a constant which does not depend on the Tx length T_k's.

Denote $T_k := \tau'+T_0\cdot\exp(r_k)$, where τ' is the overhead of a successful transmission, and $T_k^p := T_0\cdot\exp(r_k)$ is the mean time used by the payload. $T_0>0$ is a "reference payload length". Let $\mathbf{r}$ be the vector of r_k's. Then

$$p(x;\mathbf{r}) = \frac{1}{E(\mathbf{r})}g(x)\cdot\prod_{k\in S(x)}(\tau'+T_0\cdot\exp(r_k)) \quad (8)$$

where

$$E(\mathbf{r}) = \sum_{x'}[g(x')\cdot\prod_{k\in S(x')}(\tau'+T_0\cdot\exp(r_k))]. \quad (9)$$

If at a state x, $k\in S(x)$, it is possible that link k is transmitting the overhead or the payload. So we define a more detailed state (x,z), where $z\in\{0,1\}^K$. Let $z_k=1$ if $k\in S(x)$ (i.e., k is in the "success" state) and link k is transmitting its payload (instead of overhead). Let $z_k=0$ otherwise. Then similar to the proof of Theorem 2, and using equation (8), we have the following product-form distribution

$$p((x,z);\mathbf{r}) = \frac{1}{E(\mathbf{r})}g(x,z)\cdot\exp(\sum_k z_k r_k) \quad (10)$$

where

$$g(x,z) = g(x)\cdot(\tau')^{|S(x)|-\mathbf{1}'\mathbf{z}}T_0^{\mathbf{1}'\mathbf{z}}. \quad (11)$$

where $\mathbf{1}'\mathbf{z}$ is the number of links that are transmitting the payload in state (x,z).

III. Distributed algorithms to approach throughput-optimality

In this section we focus on a *scheduling problem* which is described below. Recall that the capacity of each link is assumed to be 1. Assume traffic arrivals at link k with an average arrival rate $\lambda_k\le 1$ (For example, if $\lambda_k=0.5$, then link k needs to be active for at least 50% percent of the time, to serve the arriving traffic). And denote the vector of arrival rates as $\lambda\in R_+^K$.

Our objective is to tune the parameters (p_k's or T_k's) of the above CSMA protocol, in a distributed way, such that (almost) any strictly feasible λ (to be defined later) can be "supported" in some sense. An algorithm is said to be "throughput-optimal" if it can support any strictly feasible λ.

It turns out that tuning p_k's is more difficult: it may increase link k's throughput via more aggressive transmissions, but may also lead to excessive collisions which harm itself and other links. However, by fixing p_k's but tuning T_k's, we show that for any strictly feasible λ, there exist T_k's such that the average service rates are equal to the arrival rates on all links. We will also present distributed algorithms to *approach* the throughput-optimality objective.

An extension of the above *scheduling problem* is a *joint scheduling and congestion control problem*, where in additional to scheduling, the arrival rate λ is adjusted by the sources in order to achieve certain fairness (or "maximal utility") objective among different links or multi-hop data flows. The algorithm to be presented later and its convergence/stability can be extended to the joint problem. But due to the limit of space, more discussion can be found in [19].

A. Using CSMA scheduling to approach throughput-optimality

We first define the feasible and strictly feasible arrival rates.

Definition 1: 1) A vector of arrival rate $\lambda \in \mathcal{R}_+^K$ (where K is the number of links) is *feasible* if there exists a probability distribution, $\bar{p}((x,z))$, over the detailed state (x,z) (i.e., $\sum_{(x,z)} \bar{p}((x,z)) = 1$ and $\bar{p}((x,z)) \geq 0$)

$$\lambda_k = \sum_{(x,z)} \bar{p}((x,z)) \cdot z_k. \tag{12}$$

This is because if λ can be scheduled by the network, the fraction of time that the network spent in the detailed states must be non-negative and sum up to 1. (Note that (12) is the probability that link k is sending its payload given the distribution of the detailed states.)

For example, in the Ad-hoc network in Fig. 2, $\lambda = (0.5, 0.5, 0.5)$ is feasible, because (12) holds if we let the probability of the detailed state $(x = (1,0,1), z = (1,0,1))$ be 0.5, the probability of the detailed state $(x = (0,1,0), z = (0,1,0))$ be 0.5, and all other detailed states have probability 0.

2) A vector of arrival rate $\lambda \in \mathcal{R}_+^K$ is *strictly feasible* if it can be written as (12) where $\sum_{(x,z)} \bar{p}((x,z)) = 1$ and $\bar{p}((x,z)) > 0$ (which is a slightly stronger condition). In other words, $\lambda \in \mathcal{R}^K$ is strictly feasible if it is in the *interior* of the region of feasible arrival rates. In the previous example, $\lambda = (0.5, 0.5, 0.5)$ is not strictly feasible since it cannot be written as (12) where all $\bar{p}((x,z)) > 0$. But $\lambda' = (0.49, 0.49, 0.49)$ is strictly feasible.

The following theorem shows that any strictly feasible λ can be supported by properly choosing the mean payload lengths $T_k^p = T_0 \exp(r_k), \forall k$.

Theorem 3: Assume that the parameters including $\gamma, \tau' > 0$, and transmission probabilities $p_k \in (0,1), \forall k$ are fixed. Given any strictly feasible arrival rate vector $\lambda \in \mathcal{R}^K$, there exists $\mathbf{r}^* \in \mathcal{R}^K$ such that the throughput (or "service rate") of link k is equal to the arrival rate for all k:

$$s_k(\mathbf{r}^*) = \frac{T_0 \cdot \exp(r_k^*)}{\tau' + T_0 \cdot \exp(r_k^*)} \sum_{x: k \in S(x)} p(x; \mathbf{r}^*) = \lambda_k, \forall k. \tag{13}$$

Remark 1: The result seems surprising since it says that even with collisions, and overhead τ' in each successful packet, the achievable throughput can still support any strictly feasible arrival rates. An intuitive explanation is as follows: if the payload size is large enough in each packet, and the duration of each collision does not increase with the transmission length, then the overhead and collisions are negligible compared to the time the system spends on successful payload transmissions. Roughly speaking, longer packets lead to higher throughput. (But there is a tradeoff since longer packets also lead to larger delays for the conflicting links.)

Remark 2: The fact that proper choice of Tx lengths can approach the maximal throughput in a single-cell network is easy to recognize, but is not trivial for general topology.

Proof: We start with the following function (the "log likelihood function" [9] if we estimate the parameter $\mathbf{r}$ from $\bar{p}((x,z))$'s). We will show that $\mathbf{r}^*$ that maximizes $F(\mathbf{r}; \lambda)$ over $\mathbf{r}$ satisfies (13).

$$\begin{aligned} & F(\mathbf{r}; \lambda) \\ = & \sum_{(x,z)} \bar{p}((x,z)) \log(p((x,z); \mathbf{r})) \\ = & \sum_{(x,z)} \bar{p}((x,z)) [\sum_k z_k r_k + \log(g(x,z)) - \log(E(\mathbf{r}))] \\ = & \sum_k \lambda_k r_k + \sum_{(x,z)} [\bar{p}((x,z)) \log(g(x,z))] - \log(E(\mathbf{r})) \end{aligned}$$

where $\lambda_k = \sum_{(x,z)} \bar{p}((x,z)) \cdot z_k$ is the arrival rate at link k. Note that $F(\mathbf{r}; \lambda)$ is concave in $\mathbf{r}$. This is because (a) the first term is linear in $\mathbf{r}$; (b) the second term does not involve $\mathbf{r}$ and (c) $E(r)$ as defined in (9) can be expanded to a sum of exponential terms of $\mathbf{r}$. So $\log(E(\mathbf{r}))$ is a log-sum-exp function which is convex [17]. Therefore $F(\mathbf{r}; \lambda)$ is concave in $\mathbf{r}$.

Consider the following optimization problem

$$\sup_{\mathbf{r}} F(\mathbf{r}; \lambda) \ . \tag{14}$$

Since $\log(p((x,z); \mathbf{r})) \leq 0$, we have $F(\mathbf{r}; \lambda) \leq 0$. Therefore $\sup_{\mathbf{r}} F(\mathbf{r}; \lambda)$ exists. Since λ is strictly feasible, $\sup_{\mathbf{r}} F(\mathbf{r}; \lambda)$ can be attained (The proof of this subtle point is given in Appendix VI-A.) So the problem is the same as $\max_{\mathbf{r}} F(\mathbf{r}; \lambda)$. Hence, the solution of (14), $\mathbf{r}^*$, satisfies

$$\begin{aligned} & \frac{\partial F(\mathbf{r}^*; \lambda)}{\partial r_k} \\ = & \lambda_k - \frac{1}{E(\mathbf{r}^*)} \sum_{x: k \in S(x)} [g(x) \cdot T_0 \cdot \exp(r_k^*) \cdot \\ & \prod_{j \in S(x), j \neq k} (\tau' + T_0 \cdot \exp(r_j^*))] \\ = & \lambda_k - \frac{1}{E(\mathbf{r}^*)} \frac{T_0 \cdot \exp(r_k^*)}{\tau' + T_0 \cdot \exp(r_k^*)} \sum_{x: k \in S(x)} [g(x) \cdot \\ & \prod_{j \in S(x)} (\tau' + T_0 \cdot \exp(r_j^*))] \\ = & \lambda_k - \frac{T_0 \cdot \exp(r_k^*)}{\tau' + T_0 \cdot \exp(r_k^*)} \sum_{x: k \in S(x)} p(x; \mathbf{r}^*) \\ = & \lambda_k - s_k(\mathbf{r}^*) = 0. \end{aligned}$$

■

In Appendix VI-B, it is further shown that $\mathbf{r}^*$ is unique (for a given strictly feasible λ).

B. Distributed Algorithm and its convergence/stability

By Theorem 3, we need to design an algorithm to solve problem (14) in a distributed way. For this purpose, we want r_k to change in the direction of the gradient $\partial F(\mathbf{r}) / \partial r_k = \lambda_k - s_k(\mathbf{r})$. However, due to the randomness of the system, λ_k and $s_k(\mathbf{r})$ cannot be obtained directly.

ALGORITHM 1. TX-LENGTH CONTROL ALGORITHM

The vectors $\mathbf{r}$ is updated at time t_i, $i = 1, 2, \ldots$. Let $t_0 = 0$ and $t_i - t_{i-1} = M$ (millisecond), $i = 1, 2, \ldots$. Let "period i" be the time between t_{i-1} and t_i, and $\mathbf{r}(i)$ be the value of $\mathbf{r}$ at the end of period i, i.e., at time t_i. Initially, link k sets $r_k(0) \in [r_{min}, r_{max}]$ where r_{min}, r_{max} are two parameters (to be further discussed). Then at time t_i, $i = 1, 2, \ldots$, update

$$r_k(i) = r_k(i-1) + \alpha(i)[\lambda_k'(i) - s_k'(i) + h(r_k(i-1))] \tag{15}$$

where $\alpha(i) > 0$ is the step size in period i, $\lambda'_k(i), s'_k(i)$ are the empirical average arrival rate and service rate[2] in period i (i.e., the actual amount of arrived traffic and served traffic in period i divided by M. Note that $\lambda'_k(i), s'_k(i)$ are random variables which are generally not equal to λ_k and $s_k(\mathbf{r}(i))$ in period i. Assume that the maximal instantaneous arrival rate is $\bar{\lambda}$, so $\lambda'_k(i) \leq \bar{\lambda}, \forall k, i$. And $h(\cdot)$ is a "penalty function", defined below, to keep $\mathbf{r}$ in a bounded region. (This is a "softer" approach than directly projecting $r_k(i)$ to the set $[r_{min}, r_{max}]$. The purpose is only to simplify the proof of Theorem 4 later.)

$$h(y) = \begin{cases} r_{min} - y & \text{if } y < r_{min} \\ 0 & \text{if } y \in [r_{min}, r_{max}] \\ r_{max} - y & \text{if } y > r_{max} \end{cases} \quad (16)$$

Intuitively speaking, Algorithm 1 says that when $r_k \in [r_{min}, r_{max}]$, if the empirical arrival rate of link k is larger than the service rate, then link k should transmit more aggressively by using a larger Tx length, and vice versa.

Algorithm 1 is parametrized by r_{min}, r_{max} which are fixed during the execution of the algorithm. Note that the choice of r_{max} affects the maximal possible payload length. Also, as discussed below, the choices of r_{max} and r_{min} also determine the "capacity region" of Algorithm 1.

We define the region of arrival rates

$$\mathcal{C}(r_{min}, r_{max}) := \{\lambda | \mathbf{r}^* := \arg\max_{\mathbf{r}} F(\mathbf{r}; \lambda) \in (r_{min}, r_{max})^K\}$$

(recall that $\mathbf{r}^*$ is unique for a given strictly feasible λ). Later we will show that the algorithm can "support" any $\lambda \in \mathcal{C}(r_{min}, r_{max})$ in some sense under certain conditions on the step sizes.

Clearly, $\mathcal{C}([r_{min}, r_{max}]) \to \mathcal{C}$ as $r_{min} \to -\infty$ and $r_{max} \to \infty$, where $\mathcal{C}$ is the set of all strictly feasible λ (by Theorem 3). Therefore, although given r_{min}, r_{max}, the region $\mathcal{C}(r_{min}, r_{max})$ is generally smaller than $\mathcal{C}$, one can choose r_{min}, r_{max} to arbitrarily approach the maximal capacity region $\mathcal{C}$. Also, there is a tradeoff between the capacity region and the maximal packet length.

Algorithm 1 involves adjusting $\mathbf{r}$, and therefore the mean payload lengths. To implement the adjustment, a link can randomize the number of packets to transmit (or the number of tasks to process) before it releases the medium (or resources). For example, assume that the length of each packet of payload is i.i.d with mean 100, and the link needs to achieve a mean payload length of 120. Then, the link can transmit one packet with probability 0.8, and transmit two packets with probability 0.2. For convenience, in this paper we randomize the payload length (in "slots") over the two integers closest to the mean, assuming that the packets can be fragmented and reassembled.

Theorem 4: Assume that the vector of arrival rates $\lambda \in \mathcal{C}(r_{min}, r_{max})$. Then

(i) If $\alpha(i) > 0$ is non-increasing and satisfies $\sum_i \alpha(i) = \infty$, $\sum_i \alpha(i)^2 < \infty$ and $\alpha(1) \leq 1$, then the algorithm converges, i.e., $\mathbf{r}(i) \to \mathbf{r}^*$ as $i \to \infty$ with probability 1, where $\mathbf{r}^*$ satisfies $s_k(\mathbf{r}^*) = \lambda_k, \forall k$.

(ii) If $\alpha(i) = \alpha$ (i.e., constant step size), then for any $\delta > 0$, there exists $\alpha > 0$ such that $\liminf_{N\to\infty} \sum_{i=1}^N s'_k(i)/N] \geq \lambda_k - \delta, \forall k$ with probability 1. In other words, one can achieve average service rates arbitrarily close to the arrival rates by choosing small enough α.

The complete proof is given in Appendix VI-C. But the result can be intuitively understood as follows. If the step size is small, r_k is "quasi-static" such that roughly, the service rate is averaged (over multiple periods) to $s_k(\mathbf{r})$, and the arrival rate is averaged to λ_k. Thus the algorithm solves the optimization problem (14) by a stochastic approximation [18] argument, such that $\mathbf{r}(i)$ converges to $\mathbf{r}^*$ in part (i), and $r(i)$ is near $\mathbf{r}^*$ with high probability in part (ii).

Corollary 1: Consider a variant of Algorithm 1:

$$r_k(i) = r_k(i-1) + \alpha(i)[\lambda'_k(i) + \epsilon - s'_k(i) + h(r_k(i-1))] \quad (17)$$

where $\epsilon > 0$. That is, the algorithm "pretends" to serve the arrival rate $\lambda + \epsilon \cdot \mathbf{1}$ (where $\epsilon > 0$) which is slightly larger than the actual λ. Assume that

$$\begin{aligned} \lambda &\in \mathcal{C}'(r_{min}, r_{max}, \epsilon) \\ &:= \{\lambda | \lambda + \epsilon \cdot \mathbf{1} \in \mathcal{C}(r_{min}, r_{max})\}. \end{aligned}$$

(i) If $\alpha(i) > 0$ is non-increasing and satisfies $\sum_i \alpha(i) = \infty$ and $\sum_i \alpha(i)^2 < \infty$ and $\alpha(1) \leq 1$, $\mathbf{r}(i) \to \mathbf{r}^*$ as $i \to \infty$ with probability 1, where $\mathbf{r}^*$ satisfies $s_k(\mathbf{r}^*) = \lambda_k + \epsilon > \lambda_k, \forall k$

(ii) If $\alpha(i) = \alpha$ (i.e., constant step size) where α is small enough, the all queues are positive recurrent.

Algorithm (17) is parametrized by r_{min}, r_{max} and ϵ. Clearly, as $r_{min} \to -\infty$, $r_{max} \to \infty$ and $\epsilon \to 0$, $\mathcal{C}'(r_{min}, r_{max}, \epsilon) \to \mathcal{C}$, the maximal capacity region.

The proof is in Appendix VI-E. A sketch: Part (i) is similar to (i) in Theorem 4. Part (ii) holds because if we choose $\delta = \epsilon/2$, then by Theorem 4, $\liminf_{N\to\infty} \sum_{i=1}^N s'_k(i)/N] \geq \lambda_k + \epsilon - \delta > \lambda_k, \forall k$ almost surely if α is small enough. Then the result follows by showing that the queues have negative drift.

IV. Numerical examples

Consider the conflict graph in Fig. 6. Let the vector of arrival rates be $\lambda = \rho \cdot \hat{\lambda}$, where $\rho \in (0, 1)$ is the "load", and $\hat{\lambda}$ is a convex combination of several maximal IS: $\hat{\lambda} = 0.2 * [1,0,1,0,1,0,0] + 0.2 * [0,1,0,0,1,0,1] + 0.2 * [0,0,0,1,0,1,0] + 0.2 * [0,1,0,0,0,1,0] + 0.2 * [1,0,1,0,0,1,0] = [0.4, 0.4, 0.4, 0.2, 0.4, 0.6, 0.2]$. Since $\rho \in (0, 1)$, λ is strictly feasible. Fix the Tx probabilities at $p_k = 1/16, \forall k$. The "reference payload length" $T_0 = 15$. The collision length (e.g., RTS length) is $\gamma = \eta \cdot 10$, and the overhead of successful transmission is $\tau' = \eta \cdot 20$, where η is a "relative size" of the overhead for simulation purpose. Later we will let $\eta \in \{1, 0.5, 0.2\}$ to illustrate the effects of overhead size.

[2]We let link k send dummy packets when the queue is empty (so each link is saturated). This ensures that the CSMA Markov chain has the desired stationary distribution in section II. Note that the transmitted dummy packets are also included when $s'_k(i)$ is computed.

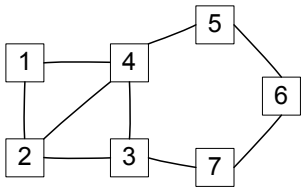
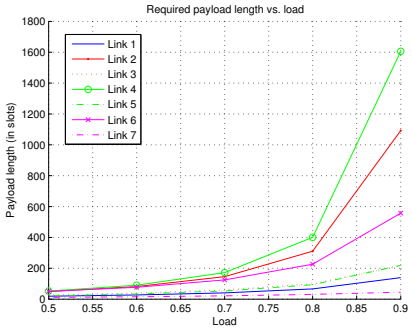
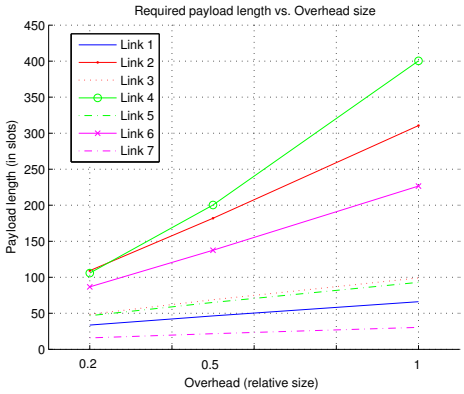


Fig. 6: An example conflict graph

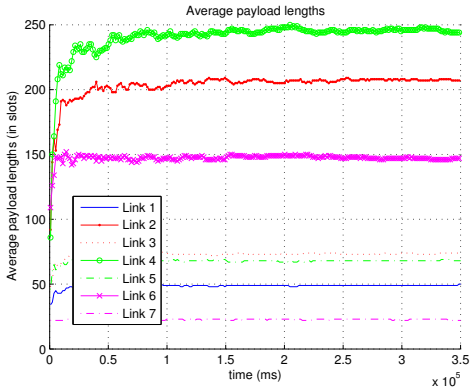


(a) Required mean payload lengths vs. load (with $\eta = 1$)

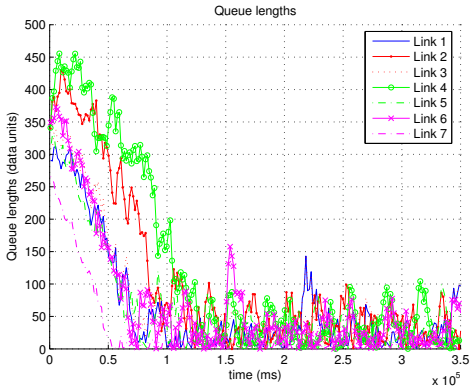


(b) Required mean payload lengths vs. the size of overhead (with $\rho = 0.8$)

Fig. 7: Required mean payload lengths



(a) Convergence of the mean payload lengths



(b) Stability of the queues

Fig. 8: Simulation of Algorithm (17) (with the conflict graph in Fig. 6)

Now we vary ρ and η. And in each case we solve problem (14) to obtain the required mean payload length $T_k^p := T_0 \cdot \exp(r_k^*), k = 1, 2, \ldots, 7$. Fig. 7 (a) shows how T_k^p's change as the load ρ changes, where $\eta = 1$. Clearly, as ρ increases, T_k^p's tend to increase. Also, the rate of increase becomes faster as ρ approaches 1. Therefore, there is a tradeoff between the throughput and transmission lengths (long transmission lengths introduce larger delays for conflicting links). Fig. 7 (b) shows how T_k^p's depends on the relative size η of overhead (with fixed $\rho = 0.8$ and $\eta \in \{1, 0.5, 0.2\}$). As expected, the smaller the overhead, the smaller T_k^p's are required.

Next, we evaluate algorithm (17) (a variant of Algorithm 1) in our C++ simulator. The update in (17) is performed every $M = 5ms$. Let the step size $\alpha(i) = 0.23/(2 + i/100)$. The upper bound $r_{max} = 5$, lower bound $r_{min} = 0$, and the "gap" $\epsilon = 0.005$. Assume the initial values of r_k's are 0.

Let the "load" of arrival rates be $\rho = 0.8$ (i.e., $\lambda = 0.8 \cdot \hat{\lambda}$), and the relative size of overhead $\eta = 0.5$ (i.e., $\gamma = 5, \tau' = 10$). To show the negative drift of the queue lengths, assume that initially all queue lengths are 300 (data units). As expected, Fig. 8 (a) shows the convergence of the mean payload lengths, and Fig. 8 shows that all queues are stable.

V. Conclusion

In this paper, we have studied CSMA-based scheduling algorithms with collisions and contention resolution. We have provided a model and given an explicit throughput formula which takes into account the cost of collisions and overhead. The formula has a simple product form due to the time-reversible structure of the model. Next, we showed that for any strictly feasible vector of arrival rates, there exists a proper setting of the mean transmission lengths for different links to support the arrival rates. We then designed distributed algorithms where each link adaptively updates its mean transmission time to approach the throughput-optimality. Sufficient conditions have been given to ensure the convergence of the proposed algorithms. Finally, simulations results were presented to illustrate and verify the main results.

In the algorithm, the transmission probabilities of the links are chosen to be fixed at a reasonable level, since we have shown that adjusting the transmission lengths alone is sufficient to approach throughput-optimality (the main goal of this

paper). However, the choices of the transmission probabilities p_k's has an effect on the probability of collisions among the probe packets. In the future, we would like to further study whether the adjustment of transmission probabilities can be combined with the algorithm.

REFERENCES

[1] X. Lin, N.B. Shroff, R. Srikant, "A Tutorial on Cross-Layer Optimization in Wireless Networks," IEEE Journal on Selected Areas in Communications, 2006.

[2] X. Wu, R. Srikant, "Bounds on the Capacity Region of Multi-Hop Wireless Networks under Distributed Greedy Scheduling," IEEE INFOCOM, 2006.

[3] J. M. Harrison and R. J. Williams, "Workload interpretation for Brownian models of stochastic processing networks," Mathematics of Operations Research, 32 (2007), 808–820.

[4] P. Chaporkar, K. Kar, and S. Sarkar, "Throughput guarantees through maximal scheduling in wireless networks," in Proceedings of 43rd Annual Allerton Conference on Communication, Control and Computing, 2005.

[5] A. Proutiere, Y. Yi, M. Chiang, "Throughput of Random Access without Message Passing," Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, March 2008.

[6] 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, 36:1936-1948, December 1992.

[7] 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.

[8] C. Joo, X. Lin, and N. B. Shroff. Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks. In Proceedings of IEEE INFOCOM, April 2008.

[9] M. Jordan, "Graphical Models", lecture notes, UC Berkeley.

[10] A. Kumar, E. Altman, D. Miorandi, and M. Goyal, "New insights from a fixed point analysis of single cell IEEE 802.11 WLANs," In Proc. IEEE INFOCOM, USA, March 2005.

[11] F. P. Kelly, "Loss networks," Ann. Appl. Prob., vol. 1, no. 3, 1991.

[12] X. Wang and K. Kar, "Throughput Modelling and Fairness Issues in CSMA/CA Based Ad-Hoc Networks," Proceedings of IEEE Infocom 2005, Miami, March 2005.

[13] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A Media Access Protocol for Wireless LANs. In Proc. ACM SIGCOMM, London, UK, 1994.

[14] F. P. Kelly, "Reversibility and Stochastic Networks," Wiley, 1979.

[15] L. Jiang and J. Walrand, "A Distributed CSMA Algorithm for Throughput and Utility Maximization in Wireless Networks," the Forty-Sixth Annual Allerton Conference on Communication, Control, and Computing, Sep. 23-26, 2008.

[16] L. Jiang and J. Walrand, "Convergence Analysis of a Distributed CSMA Algorithm for Maximal Throughput in a General Class of Networks," Technical Report, UC Berkeley, Dec 2008. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-185.html

[17] S. Boyd and L. Vandenberghe, "Convex Optimization", Cambridge University Press, 2004.

[18] V. Borkar, "Stochastic Approximation: A Dynamical Systems Viewpoint," 2008, draft copy.

[19] L. Jiang and J. Walrand, "A Novel Approach to Model and Control the Throughput of CSMA/CA Wireless Networks", Technical Report, UC Berkeley, Jan 2009. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-8.html

[20] S. Rajagopalan and D. Shah, "Distributed Algorithm and Reversible Network", CISS'08.

[21] J. Ni and R. Srikant, "Distributed CSMA/CA algorithms for achieving maximum throughput in wireless networks," in Proc. of ITA, Feb 2009.

[22] C. Bordenave, D. McDonald, A. Proutiere, "Performance of random medium access control, an asymptotic approach," in Proceedings of the ACM Sigmetrics 2008, pp. 1-12.

[23] J. Liu, Y. Yi, A. Proutiere, M. Chiang, and H. V. Poor, "Maximizing utility via random access without message passing," Microsoft Research Technical Report, September 2008.

[24] V. Borkar, Stochastic Approximation: A Dynamical Systems Viewpoint, 2008.

VI. APPENDICES

A. Proof of the attainability of $\sup_{\mathbf{r}} F(\mathbf{r};\lambda)$

Lemma 1: Assume that λ is strictly feasible. Problem (14) is the dual problem of the following convex optimization problem, where the vector $\mathbf{u}$ can be viewed as a probability distribution over the detailed state (x,z):

$$\begin{aligned} \max_{\mathbf{u},\mathbf{f}} \quad & \{\sum_{(x,z)}[-u_{(x,z)}\log(u_{(x,z)})] + \\ & \sum_{(x,z)}[u_{(x,z)}\cdot\log(g(x,z))]\} \\ \text{s.t.} \quad & \sum_{(x,z):z_k=1} u_{(x,z)} \geq \lambda_k, \forall k \\ & u_{(x,z)} \geq 0, \sum_{(x,z)} u_{(x,z)} = 1. \end{aligned} \tag{18}$$

Proof: Since λ is strictly feasible, problem (18) is strictly feasible and satisfies the Slater condition [17].

Let $y_k \geq 0$ be the dual variable associated with the constraint $\sum_{(x,z):z_k=1} u_{(x,z)} \geq \lambda_k$, then a partial Lagrangian is

$$\begin{aligned} & \mathcal{L}(\mathbf{u};\mathbf{y}) \\ = & \sum_{(x,z)}[-u_{(x,z)}\log(u_{(x,z)})] + \sum_{(x,z)}[u_{(x,z)}\cdot\log(g(x,z))] \\ & + \sum_k y_k[\sum_{(x,z):z_k=1} u_{(x,z)} - \lambda_k] \\ = & \sum_{(x,z)}\{u_{(x,z)}[-\log(u_{(x,z)}) + \log(g(x,z)) + \sum_{k:z_k=1} y_k]\} \\ & - \sum_k (y_k\lambda_k). \end{aligned}$$

So

$$\frac{\partial L(\mathbf{u},\mathbf{f};\mathbf{y})}{\partial u_{(x,z)}} = -\log(u_{(x,z)}) - 1 + \log(g(x,z)) + \sum_{k:z_k=1} y_k.$$

If $u_{(x,z)} = p((x,z);\mathbf{y})$ (cf. equation (10)), then the partial derivative

$$\frac{\partial L(\mathbf{u},\mathbf{f};\mathbf{y})}{\partial u_{(x,z)}} = \log(E(\mathbf{y})) - 1$$

which is the same for all state (x,z) (Given the dual variables $\mathbf{y}$, $\log(E(\mathbf{y}))$ is a constant). This means that $u_{(x,z)} = p((x,z);\mathbf{y}) > 0$ maximizes $\mathcal{L}(\mathbf{u};\mathbf{y})$ over $\mathbf{u}$ subject to $u_{(x,z)} \geq 0, \sum_{(x,z)} u_{(x,z)} = 1$ (since it is impossible to increase $\mathcal{L}(\mathbf{u};\mathbf{y})$ over $\mathbf{u}$ by slightly perturbing $\mathbf{u}$).. Denote $l(\mathbf{y}) = \max_{\mathbf{u}} \mathcal{L}(\mathbf{u};\mathbf{y})$, and $u_{(x,z)}(y) = p((x,z);\mathbf{y})$ as the maximizer. Then the dual problem of (18) is $\min_{\mathbf{y}\geq 0} l(\mathbf{y})$. Plugging the expression of $u_{(x,z)}(y)$ into $l(\mathbf{y})$, it is not difficult to find that $\min_{\mathbf{y}\geq 0} l(\mathbf{y})$ is equivalent to problem (14), with $\mathbf{r}$ and $\mathbf{y}$ interchangeable. ■

Since problem (18) satisfies the Slater condition, there exist (finite) optimal dual variable $\mathbf{r}^*$, which is also the solution of (14). Therefore, $\sup_{\mathbf{r}} F(\mathbf{r};\lambda)$ is attainable.

B. Proof of the uniqueness of $\mathbf{r}^*$

Proof: Note that the objective function of (18) is strictly concave. Therefore $\mathbf{u}^*$, the optimal solution of (18) is unique. Consider two extended state $(\mathbf{e}_k, \mathbf{e}_k)$ and $(\mathbf{e}_k, \mathbf{0})$, where $\mathbf{e}_k$ is the K-dimensional vector whose k'th element is 1 and all other elements are 0's. We have $u^*_{(\mathbf{e}_k,\mathbf{e}_k)} = p((\mathbf{e}_k, \mathbf{e}_k); \mathbf{r}^*)$ and $u^*_{(\mathbf{e}_k,\mathbf{0})} = p((\mathbf{e}_k, \mathbf{0}); \mathbf{r}^*)$. Then by (10),

$$u_{(\mathbf{e}_k,\mathbf{e}_k)}(\mathbf{r}^*)/u_{(\mathbf{e}_k,\mathbf{0})}(\mathbf{r}^*) = \exp(r^*_k) \cdot (T_0/\tau'). \qquad (19)$$

Suppose that r^* is not unique, that is, there exist $\mathbf{r}^*_I \neq \mathbf{r}^*_{II}$ but both are optimal $\mathbf{r}$. Then, $r^*_{I,k} \neq r^*_{II,k}$ for some k. This contradict to (19) and the uniqueness of $\mathbf{u}^*$. Therefore $\mathbf{r}^*$ is unique. ■

C. Proof of Theorem 4

We will use results in [24] to prove Theorem 4. Similar techniques have been used in [23] to analyze the convergence of an algorithm in [15].

1) Part (i): Decreasing step size: Define the concave function

$$H(y) := \begin{cases} -(r_{min} - y)^2/2 & \text{if } y < r_{min} \\ 0 & \text{if } y \in [r_{min}, r_{max}] \\ -(r_{max} - y)^2/2 & \text{if } y > r_{max} \end{cases} \qquad (20)$$

Note that $dH(y)/dy = h(y)$ where $h(y)$ is defined in (16). Let $G(\mathbf{r}; \lambda) = F(\mathbf{r}; \lambda) + \sum_k H(r_k)$. Since λ is strictly feasible, $\max_{\mathbf{r}} F(\mathbf{r}; \lambda)$ has a unique solution $\mathbf{r}^*$ (Appendix VI-B). That is, $F(\mathbf{r}^*; \lambda) > F(\mathbf{r}; \lambda), \forall \mathbf{r} \neq \mathbf{r}^*$. Since $\mathbf{r}^* \in (r_{min}, r_{max})^K$ by assumption, then $\forall \mathbf{r}$, $\sum_k H(r^*_k) = 0 \geq \sum_k H(r_k)$. Therefore, $G(\mathbf{r}^*; \lambda) > G(\mathbf{r}; \lambda), \forall \mathbf{r} \neq \mathbf{r}^*$. So $\mathbf{r}^*$ is the unique solution of $\max_{\mathbf{r}} G(\mathbf{r}; \lambda)$. Because $\partial G(\mathbf{r}; \lambda)/\partial r_k = \lambda_k - s_k(\mathbf{r}) + h(r_k)$, Algorithm 1 tries to solve $\max_{\mathbf{r}} G(\mathbf{r}; \lambda)$ with inaccurate gradients.

Let $\mathbf{v}^s(t)$ be the solution of the following differential equation (for $t \geq s$)

$$dv_k(t)/dt = \lambda_k - s_k(\mathbf{v}(t)) + h(v_k(t)), \forall k \qquad (21)$$

with the initial condition that $\mathbf{v}^s(s) = \bar{\mathbf{r}}(s)$. So, $\mathbf{v}^s(t)$ can be viewed as the "ideal" trajectory of Algorithm 1 with the smoothed arrival rate and service rate. And (21) can be viewed as a continuous-time gradient algorithm to solve $\max_{\mathbf{r}} G(\mathbf{r}; \lambda)$. We have shown above that $\mathbf{r}^*$ is the unique solution of $\max_{\mathbf{r}} G(\mathbf{r}; \lambda)$. Therefore $\mathbf{v}^s(t)$ converges to the unique $\mathbf{r}^*$ for any initial condition.

For simplicity, assume that M milliseconds is a multiple of minislots. (However, the result does not change otherwise.) Define $Y(i-1) := (s'_k(i), w_0(i))$ where $w_0(i)$ is the state w at time t_i. Then $\{Y(i)\}$ is a Markov process. The update in Algorithm 1 can be written as

$$r_k(i) = r_k(i-1) + \alpha(i) \cdot [f(r_k(i-1), Y(i-1)) + M(i)]$$

where $f(r_k(i-1), Y(i-1)) := \lambda_k - s'_k(i) + h(r_k(i-1))$, and $M(i) = \lambda'_k(i) - \lambda_k$ can be viewed as Martingale noise.

To use Corollary 8 in page 74 of [24] to show Algorithm 1's almost-sure convergence to $\mathbf{r}^*$, the following conditions are sufficient:

(i) $f(\cdot, \cdot)$ is Lipschitz in the first argument, and uniformly in the second argument. This holds by the construction of $h(\cdot)$;

(ii) The transition kernel of $Y(i)$ is continuous in $\mathbf{r}(i)$;

(iii) (21) has a unique convergent point $\mathbf{r}^*$, which has been shown above;

(iv) Tightness condition ((†) in [24], page 71): This is satisfied since $Y(i)$ has a bounded state-space (cf. conditions (6.4.1) and (6.4.2) in [24], page 76).

(v) With Algorithm 1, $r_k(i)$ is bounded $\forall k, i$ almost surely. This is prove in the following lemma.

Lemma 2: With Algorithm 1, $\mathbf{r}(i)$ is always bounded. Specifically, $r_k(i) \in [r_{min} - 2, r_{max} + 2\bar{\lambda}], \forall k, i$.

Proof: We first prove the upper bound $r_{max} + 2\bar{\lambda}$ by induction: (a) $r_k(0) \leq r_{max} \leq r_{max} + 2\bar{\lambda}$; (b) For $i \geq 1$, if $r_k(i-1) \in [r_{max} + \bar{\lambda}, r_{max} + 2\bar{\lambda}]$, then $h(r_k(i-1)) \leq -\bar{\lambda}$. Since $\lambda'_k(i) - s'_k(i) \leq \bar{\lambda}$, we have $r_k(i) \leq r_k(i-1) \leq r_{max} + 2\bar{\lambda}$. If $r_k(i-1) \in (r_{min}, r_{max} + \bar{\lambda})$, then $h(r_k(i-1)) \leq 0$. Also since $\lambda'_k(i) - s'_k(i) \leq \bar{\lambda}$ and $\alpha(i) \leq 1, \forall i$, we have $r_k(i) \leq r_k(i-1) + \bar{\lambda} \cdot \alpha(i) \leq r_{max} + 2\bar{\lambda}$. If $r_k(i-1) \leq r_{min}$, then

$$\begin{aligned} r_k(i) &= r_k(i-1) + \alpha(i)[\lambda'_k(i) - s'_k(i) + h(r_k(i-1))] \\ &\leq r_k(i-1) + \alpha(i)\{\bar{\lambda} + [r_{min} - r_k(i-1)]\} \\ &\leq \bar{\lambda} + r_{min} \leq r_{max} + 2\bar{\lambda}. \end{aligned}$$

The lower bound $r_{min} - 2$ can be proved similarly. ■

So, by [24], $\mathbf{r}(i)$ converges to $\mathbf{r}^*$ almost surely.

D. Part (ii): Constant step size

Let $r[i], i = 0, 1, \ldots$ be the sequence of $\mathbf{r}$ generated by Algorithm 1 with constant step size α. Let $\bar{\mathbf{r}}(t)$ be the interpolated trajectory of $\mathbf{r}[i]$, defined as follows.

$$\bar{\mathbf{r}}(\alpha \cdot i) = \mathbf{r}[i], \forall i = 0, 1, \ldots$$

with $\bar{\mathbf{r}}(t)$ defined on $[\alpha \cdot i, \alpha \cdot (i+1)]$ by linear interpolation of $\bar{\mathbf{r}}(\alpha \cdot i)$ and $\bar{\mathbf{r}}(\alpha \cdot (i+1))$ for all i.

Define $\mathbf{v}^s(t)$ as the solution of (21), with the initial condition that $\mathbf{v}^s(s) = \bar{\mathbf{r}}(s)$. It has been shown above that $\mathbf{v}^s(t)$ converges to a unique $\mathbf{r}^*$ for any initial condition.

Then, by Theorem 7 in [24] (page 114), for a given initial value $\bar{\mathbf{r}}(s)$, we have for any $T > 0$,

$$\sup_{t \in [s, s+T]} ||\bar{\mathbf{r}}(t) - \mathbf{v}^s(t)|| \to 0 \text{ in probability as } \alpha \to 0 \qquad (22)$$

uniformly in $s \geq 0$ (and in the initial value). In other words, when α is very small, then in the time window $[s, s+T]$, the trajectory $\bar{\mathbf{r}}(t)$ closely approximates $\mathbf{v}(t)$ with high probability.

Let $\Delta := d(\mathbf{r}^*)$ where $d(\mathbf{y})$ is defined as the minimum distance from $\mathbf{y}$ to the boundary of the region $[r_{min}, r_{max}]^K$ (where $\mathbf{y}$ is a dummy variable). $\Delta > 0$ since $\mathbf{r}^* \in (r_{min}, r_{max})^K$ by assumption.. Let $\epsilon_1 = \Delta/4$.

Since the differential equation defines a gradient algorithm, the distance between $\mathbf{v}(t)$ and $\mathbf{r}^*$ decreases monotonically with t.

Denote $\mathcal{D} := [r_{min} - 2, r_{max} + 2\bar{\lambda}]^K$. For any $\eta \in (0,1)$, one can choose $T > 3\epsilon_1$ such that for any initial $\mathbf{v}(s) \in \mathcal{D}$,

$$||\mathbf{v}(t) - \mathbf{r}^*|| \leq \epsilon_1, \forall t \geq s + \eta \cdot T. \tag{23}$$

That is, from any initial point in $\mathcal{D}$, we can choose T large enough such that $\mathbf{v}(t)$ is close enough to $\mathbf{r}^*$ for at least a fraction $1-\eta$ of time in a time window of T.

By (22), given the T, ϵ_1 chosen above, for any $\delta_1 > 0, \alpha_1 > 0$, there exists $\alpha \leq \alpha_1$ such that $P(A) \geq 1 - \delta_1$ where the "event" $A := \{\sup_{t\in[s,s+T]} ||\bar{\mathbf{r}}(t) - \mathbf{v}(t)|| \leq \epsilon_1\}$. If A happens, then for all $t \in [s + \eta \cdot T, s + T]$, we have (i) $||\bar{\mathbf{r}}(t) - \mathbf{r}^*|| \leq 2\epsilon_1$; (ii) $\bar{\mathbf{r}}(t) \in [r_{min}, r_{max}]^K$; and (iii) $||\bar{\mathbf{r}}(t_1) - \bar{\mathbf{r}}(t_2)|| \leq 3\epsilon_1$ for any $t_1, t_2 \geq [s + \eta \cdot T, s + T]$.

In Algorithm 1 with constant step size,

$$r_k(i) = r_k(i-1) + \alpha \cdot [\lambda'_k(i) - s'_k(i) + h(r_k(i-1))].$$

Note that $h(r_k(i-1)) = 0$ if $r_k(i-1) \in [r_{min}, r_{max}]^K$. Summing over the period i's where , and denote $\underline{i}, \bar{i}$ as the minimum and maximum of i's such that $\alpha \cdot i \in [s+\eta\cdot T, s+T]$. If A happens, we sum over $\underline{i} + 1 \leq i \leq \bar{i}$ (and using (iii) above),

$$\begin{aligned} & 3\epsilon_1 \geq r_k(\bar{i}) - r_k(\underline{i}) \\ = \; & \alpha[\sum_{\underline{i}+1\leq i\leq \bar{i}} \lambda'_k(i) - \sum_{\underline{i}+1\leq i\leq \bar{i}} s'_k(i)] \end{aligned}$$

Let i_0 be the initial i, i.e., $\alpha \cdot i_0 = s$. Then $\alpha \sum_{i_0+1\leq i\leq \bar{i}} s'_k(i) \geq \alpha \sum_{\underline{i}+1\leq i\leq \bar{i}} s'_k(i) \geq \alpha \sum_{\underline{i}+1\leq i\leq \bar{i}} \lambda'_k(i) - 3\epsilon_1$.

If A^c happens, $\alpha \sum_{i_0+1\leq i\leq \bar{i}} s'_k(i) \geq 0 \geq \alpha \sum_{\underline{i}+1\leq i\leq \bar{i}} \lambda'_k(i) - \bar{\lambda} \cdot T$. The last inequality is because $\sum_{\underline{i}+1\leq i\leq \bar{i}} \lambda'_k(i) \leq (\bar{i} - \underline{i}) \cdot \bar{\lambda} \leq \bar{\lambda} \cdot T/\alpha$. (Also assume $\bar{\lambda} \geq 1$ WLOG.) Denote $\bar{s}_k := \sum_{i_0+1\leq i\leq \bar{i}} s'_k(i)/(\bar{i} - i_0)$. We then have

$$\begin{aligned} & T \cdot E(\bar{s}_k) \geq \\ \geq \; & \alpha E[\sum_{i_0+1\leq i\leq \bar{i}} s'_k(i)] \text{ (since } T \geq \alpha \cdot (\bar{i} - i_0)) \\ = \; & \alpha E[I(A) \sum_{i_0+1\leq i\leq \bar{i}} s'_k(i)] + \alpha E[I(A^c) \sum_{i_0+1\leq i\leq \bar{i}} s'_k(i)]\} \\ \geq \; & E[I(A)(\alpha \sum_{\underline{i}+1\leq i\leq \bar{i}} \lambda'_k(i) - 3\epsilon_1)] + \\ & E[I(A^c)(\alpha \sum_{\underline{i}+1\leq i\leq \bar{i}} \lambda'_k(i) - \bar{\lambda} \cdot T)] \\ \geq \; & \alpha \cdot (\bar{i} - \underline{i})\lambda_k - (1-\eta)3\epsilon_1 - \eta\bar{\lambda}T \text{ (since } \bar{\lambda}T > 3\epsilon_1) \\ \geq \; & [T \cdot (1-\eta) - 2\alpha_1]\lambda_k - (1-\eta)3\epsilon_1 - \eta\bar{\lambda}T \\ & \text{(since } T \cdot (1-\eta) \leq (\bar{i} - \underline{i})\alpha + 2\alpha_1) \end{aligned}$$

That is, $E(\bar{s}_k) \geq [(1-\eta) - 2\alpha_1/T]\lambda_k - (1-\eta)3\epsilon_1/T - \eta\bar{\lambda}$.

Since $\eta, \epsilon_1, \alpha_1$ can be made arbitrarily close to 0 and T can be made arbitrarily large, therefore $E(\bar{s}_k)$ can be arbitrarily close to λ_k. Finally, let $N' = \lfloor T/\alpha \rfloor$, and denote $\bar{s}_k^{(n)} = \sum_{n\cdot N'+1\leq i\leq (n+1)\cdot N} s'_k(i)/N'$ for $n = 0, 1, 2, \ldots$. We can choose α, T to make $E_n(\bar{s}_k^{(n)}) \geq \lambda_k - \delta$ for arbitrarily small δ, where $E_n(\cdot)$ means the expectation conditioned on anything that happened up to period $n \cdot N'$. In that case, $\liminf_{N\to\infty} \sum_{i=1}^N s'_k(i)/N] \geq \lambda_k - \delta$ almost surely (since the martingale noise $\bar{s}_k^{(n)} - E_n(\bar{s}_k^{(n)}), n = 0, 1, 2, \ldots$ averaged to 0 almost surely).

E. Proof of Corollary 1

The proof of part (i) is similar to part (i) of Theorem 4.

Part (ii): Similar to the proof of part (ii) in Theorem 4, for any $\delta > 0$, there exists α, T, and $N' = \lfloor T/\alpha \rfloor$, such that $E_n(\bar{s}_k^{(n)}) \geq \lambda_k + \epsilon - \delta$, where $\bar{s}_k^{(n)} = \sum_{n\cdot N'+1\leq i\leq (n+1)\cdot N} s'_k(i)/N'$ for $n = 0, 1, 2, \ldots$. (Note that we have replaced λ_k by $\lambda_k + \epsilon$ in the above inequality, since here the algorithm pretends to serve the arrival rates $\lambda + \mathbf{1} \cdot \epsilon$.). Let $\delta = \epsilon/2$. Then there exists α, T, and $N' = \lfloor T/\alpha \rfloor$, such that $E_n(\bar{s}_k^{(n)}) \geq \lambda_k + \epsilon - \delta = \lambda_k + \epsilon/2$. Denote by $Q_k^{(n)}$ the queue size of link k at time $t_{n\cdot N'}$. If $Q_k^{(n)} \geq (N'+1)M$, then from time $t_{n\cdot N'}$ to $t_{(n+1)\cdot N'}$, the queue length is positive (since it at most decreases by 1 per time unit, i.e., millisecond here), so the amount of departure is equal to the amount of service. Therefore, if $Q_k^{(n)} \geq (N'+1)M$, then $E_n(Q_k^{(n+1)}|Q_k^{(n)}) - Q_k^{(n)} = [\lambda_k - E_n(\bar{s}_k^{(n)})] \cdot M \cdot N' \leq -(\epsilon/2)M \cdot N'$, i.e., the queue has negative drift. By the Foster-Lyapunov stability criterion, the queue is positive recurrent. The same is true for other queues.