Connection-Level Scheduling in Wireless Networks Using Only MAC-Layer Information

Javad Ghaderi, Tianxiong Ji, R. Srikant
Department of ECE and Coordinated Science Lab.
University of Illinois at Urbana-Champaign
{jghaderi, tji2, rsrikant}@illinois.edu

***Abstract*—The paper studies throughput-optimal scheduling in wireless networks when there are file arrivals and departures. In the case of single-hop traffic, the well-studied Max Weight algorithm provides soft priorities to links with larger queue lengths. If packets arrive in bursts during file arrival instants, then large variances in file sizes would imply that some links will have very large queue lengths while others will have small queue lengths. Thus, links with small queue lengths may be starved for long periods of time. An alternative is to use only MAC-layer queue lengths in making scheduling decisions; in fact, typically only this information is available since scheduling is performed at the MAC layer. Therefore the questions we ask in this paper are the following: (i) is scheduling using only MAC-layer queue length information throughput-optimal? and (ii) does it improve delay performance compared to the case where scheduling is performed using the total number of packets waiting at a link? We affirmatively answer both questions in the paper (the first theoretically and the second using simulations), making minimal assumptions on the transport-layer window control mechanism.**

I. Introduction

In order to operate wireless systems efficiently, scheduling algorithms are needed to facilitate simultaneous transmissions of different users. Scheduling algorithms for wireless networks have been widely studied since Tassiulas and Ephremides [1] proposed the *Max Weight* algorithm. The Max Weight Scheduling (MWS) algorithm is throughput optimal in the sense that it can stabilize the queues of the network for the largest set of arrival rates possible without knowing the actual arrival rates. Max Weight works under very general conditions but it does not consider *connection-level dynamics*. It considers *packet-level dynamics* assuming that there is a fixed set of users and packets are generated by each user according to some stochastic process. Moreover, there is no notion of congestion control while most modern communication networks use some congestion control mechanism for fairness purposes or to avoid excessive congestion inside the network [2]. There is a rich body of literature on the packet-level stability of scheduling algorithms. Stability of wireless networks under connection-level dynamics has been studied in, e.g., [3], [4], [5], [6]. The implicit assumption in these works is that algorithm can fully observe the dynamics of queues for different connections while, in practice, the scheduler is implemented as part of the MAC layer and can thus, use only the MAC-layer queue lengths.

Research supported by NSF grant CNS 07-21286, AFOSR MURI FA 9550-10-1-0573, and ARO MURI Subcontracts from Maryland and Berkeley.

In this paper, we are interested in the scenario where files/connections arrive into and depart from an ad hoc wireless network. Upon arrival of a file, a TCP connection is established which regulates the injection of packets to MAC layer. The scheduling algorithm must determine which links can transmit packets at each time instant. When the transmission of a file ends, its corresponding TCP connection is closed and the file departs the system. If the scheduler has access to the total queue length at Transport and MAC layers, then it can use Max Weight algorithm to achieve throughput optimality. However, this would lead to a poor delay performance because files with large sizes might get priority over many small size files for long periods of time. An alternative is to implement a weight-based MAC algorithm as in [1], but use the MAC-layer queue, then large files will not dominate the service because files are stored at the transport layer and only a few packets are released at a time to the MAC layer. However, it is not obvious that such a system will be throughput-optimal.

For the connection-level model, we show that by appropriately choosing weights which are functions of the MAC-layer queue and using the Max-Weight-type scheduling algorithms, throughput optimality is achieved. The only assumption about protocols that we make is that each TCP window size is at least one and that there is a maximum window size, both of which are true for all implementations of TCP. We make no other assumptions on the dynamics of the TCP window flow control mechanism. On the other hand, the fact that our scheduler uses only the MAC-layer information is consistent with the actual implementation, because in reality, the scheduler is part of the MAC layer and might not have access to the transport layer. We will present simulations that verify the fact that our scheduling algorithm improves the delay performance.

Next, we consider the issue of distributed implementation of our scheduling algorithm. CSMA (*Carrier-Sense-Multiple-Access*) scheduling algorithms have attracted attention recently because they can be implemented in a decentralized way. Initially, to prove the throughput optimality of the CSMA scheduling algorithms, most of the papers make the time-scale separation assumption, i.e., convergence of *CSMA Markov Chain* to its stationary distribution happens at a much faster time-scale than queue changes in the network. Recently, it has been shown that CSMA indeed achieve throughput optimality without the time-scale separation assumption [12], [15], [10], [16]. The key element of such efficient CSMA algorithms is

using an appropriate queue-based weight in CSMA parameters. In this paper, we show that CSMA algorithms achieve throughput optimality even when we use only MAC-layer queue length information.

At this point, we comment on the differences between our paper and a related model considered in [7]. In [7], throughput-optimal scheduling algorithms have been derived for a connection-level model of a wireless network assuming that each link has access to the number of files waiting at the link. Here, we make no such assumption and use only MAC-layer queue information which is readily available. Further, [7] assumes a time-scale separation between CSMA and the file arrival-departure process. Such an assumption is not made in this paper.

In a nutshell, the main contributions of this paper are the following:

- Using the total queue length as the weight in MWS algorithms causes short files to experience high latencies. Instead, we use the MAC-layer queues in our algorithm and show that such an algorithm is still throughput optimal and its overall delay performance is better than traditional MWS in simulations.
- In reality, wireless scheduling is performed at the MAC-layer and the scheduler does not have access to the total queue length information. Hence, our algorithm design is based on a more practical assumption.
- A distributed CSMA implementation of the algorithm is proposed such that each user only needs to know its own MAC-layer queue and carrier sensing information. The throughput optimality of the distributed algorithm is established without time-scale separation assumption.

Due to page limits, we have only presented the main ideas behind the proofs and refer the interested reader to [17] for the complete proofs. The rest of the paper is organized as follows. In Section II, we describe our models for the wireless network, file arrivals, and Transport and MAC layers. We propose our scheduling algorithm in Section III and also prove its throughput optimality. Some simulation results are also given to investigate throughput optimality and delay performance of our algorithm. Section IV is devoted to the distributed implementation of the optimal algorithm. It also contains the proof of throughput optimality of the distributed algorithm. Finally, we will end the paper with conclusions.

II. System Model

A. Model of Wireless Network

Consider a wireless network consisting of a set of nodes where each node could be a source and/or a destination for another node. We assume single-hop communications. Time is slotted. In [8] which was a three-page extended abstract, we considered the case where the file sizes are bounded. Here, we consider a more practical case. File arrivals occur according to an i.i.d process with mixture of geometric distributions as described next. Let λ_l denote the file arrival rate at link l. For simplicity, we can assume that files arrive according to an

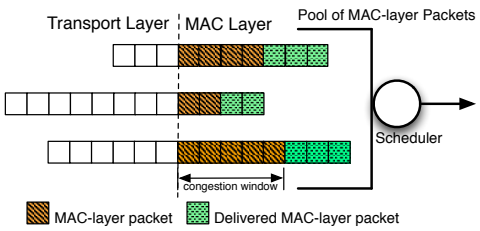

Fig. 1. Transport/MAC layers: the packets at the MAC layer need not be in separate queues as shown above, they can be in a single queue.

i.i.d Bernoulli process with rate λ_l. There are K possible file types where the files of type i are geometrically distributed with mean $1/\eta_i$ packets. The file arrived at link l can belong to type i with probability p_{li}, $i = 1, 2, .., K$. Our motivation for selecting such a model is due to the *heavy-tail distribution* of file sizes in the Internet. It is believed that, see e.g., [9], that most of bytes are generated by long files while most of the flows are short flows. By controlling the probabilities p_{li}, for the same average file size, we can obtain distributions with such properties. See [17] for some examples.

Furthermore, we assume that each link has a unit service rate, i.e., in each time slot, one packet could be successfully transmitted over a link. We use the notion of the *conflict graph* to capture the interference constraints. Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ denote the conflict graph of the wireless network, where each vertex in $\mathcal{V}$ is a communication link in the wireless network. There is an edge $(l_1, l_2) \in \mathcal{E}$ between vertices l_1 and l_2 if simultaneous transmissions over communication links l_1 and l_2 are not successful. Therefore, at each time slot, the active links should form an independent set of $\mathcal{G}$, i.e., no two scheduled vertices can share an edge in $\mathcal{G}$. Let $N = |\mathcal{V}|$ denote the number of communication links in the wireless network.

Formally, a schedule can be represented by a vector $X = [x_l : l = 1, ..., N]$ such that $x_l \in \{0, 1\}$ and $x_i + x_j \leq 1$ for all $(i, j) \in \mathcal{E}$. A schedule can also be represented by a set s of links such that $l \in s$ if $x_l = 1$ (and $l \notin s$ if $x_l = 0$). Let $\mathcal{M}$ denote the set of all feasible schedules. At each time slot, a feasible schedule is chosen by the scheduling algorithm based on the current network information. Let $m_l = \sum_{i=1}^{K} p_{li}/\eta_i$ denote the mean file size at link l, and define the work load at link l by $\rho_l = \lambda_l m_l$. Then, the capacity region of the network is the set of all load vectors $\boldsymbol{\rho} = (\rho_1, \cdots, \rho_N)$ that make the network queues stable. It is well known, e.g., see [1], that, under our model, the capacity region is given by

$$\mathcal{C} = \left\{ \boldsymbol{\rho} : \exists \boldsymbol{\mu} \in Co(\mathcal{M}) \text{ s.t. } \boldsymbol{\rho} < \boldsymbol{\mu} \right\}$$

where $Co(\cdot)$ is the convex hull operator. When dealing with vectors, inequalities should be interpreted component-wise. A scheduling algorithm is called throughput-optimal if it stabilizes the queues in the network for any load vector inside the capacity region $\mathcal{C}$.

B. Models of Transport and MAC Layers

Upon arrival of a file at a source, a TCP-connection is established that regulates the injection of packets into the MAC layer. The transmission of MAC-layer packets is itself controlled by the scheduling algorithm. Once transmission of a file ends, file departs and the corresponding TCP-connection will be closed.

At each link, we index the files according to their arriving order such that the index 1 is given to the earliest file. This means that once transmission of a file ends, the indices of the remaining files are updated such that indices again start from 1 and are consecutive. Note that the indexing rule is not part of the algorithm implementation and it is used here only for the purpose of analysis.

Let $W_{li}[t]$ denote the congestion window size of file i at link l at time t. W_{li} is a time-varying sequence which changes as a result of TCP congestion control. If the congestion window of file i is not full, TCP will continue injecting packets from the remainder of file i to the congestion window until file i has no packets remaining at the Transport layer (See Figure 1). It is important to note that the MAC layer does not know the number of remaining packets at the Transport layer, so any scheduling decision has to be made based on the MAC-layer information only. It is reasonable to assume that $1 \leq W_{li}[t] \leq W_{max}$, i.e., all congestion window sizes are upper-bounded by a constant W_{max}.

Let $q_{li}[t]$ and $q_{li}^{mac}[t]$ denote the number of remaining packets and the number of MAC-layer packets of file i at link l at time t respectively. Let $n_l[t]$ denote the number of files at link l at time t. Therefore, the total number of packets at link l is $q_l[t] = \sum_{i=1}^{n_l[t]} q_{li}[t]$, and the total number of MAC-layer packets at link l is $q_l^{mac}[t] = \sum_{i=1}^{n_l[t]} q_{li}^{mac}[t]$. Based on our model so far, we have $n_l[t] \leq q_l^{mac}[t] \leq q_l[t]$, where the lower bound follows from the fact that each window size is at least one.

C. The State of the Network

We aim to find a scheduling algorithm which only uses the MAC-layer information to stabilize the system when the traffic loads are within the capacity region. We not only allow the scheduling algorithm to choose the set of active links but also allow the algorithm to determine which MAC-layer packets to serve at each time instant. For example, our analysis is general enough to allow even service disciplines that are not FIFO (*First-in, First-Out*) at each MAC-layer queue. However, we assume that the scheduling algorithm is such that the network state can be described by a Markov chain as we will specify now. Let $\mathcal{I}_l$ denote additional MAC-layer information, other than MAC-layer queues and congestion window sizes, that might be needed for a scheduling algorithm to update the state of the network. Obviously, such additional MAC-layer information depends on specific service discipline utilized by the scheduling algorithm to serve the MAC-layer packets of the active links. For example, if a scheduling algorithm serves MAC-layer packets on a FIFO basis at each link, then $\mathcal{I}_l$ includes the ordering of MAC-layer packets and the rule according which the congestion window sizes change. On the other hand, if a scheduling algorithm servers MAC-layer packets in a totally random manner, then $\mathcal{I}_l$ does not need to maintain the ordering information of MAC-layer packets.

Let $\xi_{li}[t]$ be the indicator that whether there are packets of file i at the Transport layer or not, i.e., if $\xi_{li}[t] = 1$, the last packet of file i has not been injected to the MAC layer; if $\xi_{li}[t] = 0$, then there is no remaining packets of file i at the Transport layer. Let $\tau_i[t] \in \{1, ..., K\}$ denote the type of file with index i at time t. It is worth mentioning that the number of remaining packets of file i at the Transport layer follows a geometric distribution with mean $\sigma_{li}[t] = 1/\eta_{\tau_i[t]}$, as long as $\xi_{li}[t] = 1$, due to the memoryless property of the geometric distribution. Note that $\sigma_{li}[t]$ is a function of time only because of re-indexing since a file might change its index from slot to slot. We define the state of link l to be

$$\mathcal{S}_l[t] = \{\boldsymbol{\xi}_l[t], \boldsymbol{q}_l^{mac}[t], \boldsymbol{W}_l[t], \boldsymbol{\sigma}_l[t], \mathcal{I}_l[t]\},$$

where $\boldsymbol{\xi}_l[t] = \{\xi_{li}[t]\}_i$, $\boldsymbol{q}_l^{mac}[t] = \{q_{li}^{mac}[t]\}_i$, $\boldsymbol{W}_l[t] = \{W_{li}[t]\}_i$, $\boldsymbol{\sigma}_l[t] = \{\sigma_{li}[t]\}_i$. The network state $\mathcal{S}[t]$ is the set of all link states, i.e., $\mathcal{S}[t] = \{\mathcal{S}_l[t]\}_l$. Suppose that the change of congestion window sizes is determined by $\mathcal{S}[t]$. Therefore, under a specific scheduling algorithm which only uses the MAC-layer information, the network Markov chain is well defined.

III. A Max Weight-Type Scheduling Algorithm

A. Algorithm Description

Define a function $f(x)$ as

$$f(x) := \frac{\log(1+x)}{g(x)}, \tag{1}$$

where $g(x)$ is an arbitrary increasing function which makes $f(x)$ an increasing concave function. Assume that $g(0) > 0$ and $f(x)$ is a continuously differentiable function on $[0, \infty)$. Our *scheduling algorithm* is as follows:

- We assign a weight of $f(q_l^{mac}[t])$ to each link l. At each time instant t, the algorithm picks a schedule $\tilde{s}[t]$ such that

$$\tilde{s}[t] \in \arg\max_{s \in \mathcal{M}} \sum_{l \in s} f(q_l^{mac}[t]). \tag{2}$$

- If link l is scheduled, we choose a MAC-layer packet at link l to transmit. The scheduling decision within link l can be based on some arbitrary service discipline, for example, FIFO or random selection.

Recall that $q_l^{mac}[t]$ is the total number of packets at the MAC-layer of link l at time t. Therefore, the scheduling decision is only based on MAC-layer information.

Theorem 1. *For any $\varepsilon > 0$, the Max Weight-type algorithm can stabilize the network for all $\boldsymbol{\rho} \in \mathcal{C}/(1+\varepsilon)$, independent of transport layer algorithm (as long as the minimum window size is one and the window sizes are bounded) and the (non-idling) service discipline used to transmit packets from active links.*

In particular, one can implement service disciplines that give priority to packets from short files if such information can be made available to the MAC layer. Such algorithms are often used in practice to reduce file transfer delays of short files. Notice that the purpose of choosing weight functions as in (1) is to achieve the throughput optimality by only using the MAC-layer information. Furthermore, such weight functions enable us to implement a fully-distributed version of the algorithm using CSMA as we will see later.

B. Proof of Throughput-Optimality

Since we use a discrete-time model, we have to specify the order in which files/packets arrive and depart, which we do below:

1) At the beginning of each time slot, a scheduling decision is made by the scheduling algorithm. Packets depart from the MAC layer of scheduled links.
2) File arrivals occur next. Once a file arrives, a new TCP connection is set up for that file with an initial pre-determined congestion window size.
3) For each TCP connection, if the congestion window is not full, packets are injected into the MAC layer from the Transport layer until the window size is fully used or there is no more packets at the Transport layer.

We re-index the files at the beginning of each time slot because some files might have been departed during the last time slot.

Define $\bar{q}_l[t] := \mathbb{E}[q_l[t]|\mathcal{S}_l[t]]$ to be the expected queue length at link l given the state $\mathcal{S}_l[t]$. Then,

$$\bar{q}_l[t] = \sum_{i=1}^{n_l[t]} \Big[\sigma_{li}[t]\xi_{li}[t] + q_{li}^{mac}[t]\Big] \tag{3}$$

where $n_l[t]$ is the number of files at link l at the beginning of time slot t, which is known if the network state $\mathcal{S}[t]$ is given. Define $\Delta n_l[t]$ as the number of new files arriving at link l at time slot t. The dynamics of $\bar{q}_l[t]$ involves the dynamics of $\boldsymbol{q}_l^{mac}[t]$, $\boldsymbol{\xi}_l[t]$ and $n_l[t]$, and, thus, it consists of: (i) departure of MAC-layer packets, (ii) new file arrivals, (iii) injection of packets into the MAC layer, and (iv) departure of files from the Transport layer:

$$\bar{q}_l[t+1] = \bar{q}_l[t] - d_l^{mac}[t] + a_l[t] + a_l^{mac}[t] - d_l^{tcp}[t]. \tag{4}$$

where $d_l^{mac}[t]$ is the number of packets that depart from the MAC layer, $a_l[t] = \sum_{i=n_l[t]+1}^{n_l[t]+\Delta n_l[t]} \sigma_{li}[t]$ is the expected number of packet arrivals of new files, $a_l^{mac}[t]$ is the total number of packets injected into the MAC layer to fill up the congestion window after scheduling and new file arrival, and $d_l^{tcp}[t] = \sum_{i=1}^{n_l[t]+\Delta n_l[t]} \sigma_{li}[t] I_{li}[t]$ is the Transport-layer "expected packet departure" because of the MAC-layer injection. Here, $I_{li}[t] = 1$ indicates the last packet of file i leaves the Transport layer during time slot t; otherwise, $I_{li}[t] = 0$. Recall that $\mathbb{E}\big[a_l[t]\big] = \rho_l$ is the mean packet arrival rate at link l.

Let $b_l[t] := a_l^{mac}[t] - d_l^{tcp}[t]$, then we rewrite (4) as

$$\begin{aligned} \bar{q}_l[t+1] &= \bar{q}_l[t] - d_l^{mac}[t] + b_l[t] + a_l[t], \\ &= \bar{q}_l[t] - x_l[t] + b_l[t] + a_l[t] + u_l[t]. \end{aligned} \tag{5}$$

where $u_l[t] = \max\{x_l[t] - q_l^{mac}[t], 0\}$ is the wasted service, i.e., when l is included in the schedule but it does not have packets to transmit. Define $\mathbb{E}_{\mathcal{S}}[\cdot] = \mathbb{E}[\cdot|\mathcal{S}[t]]$. Lemma 1 characterizes the first and the second moments of $b_l[t]$.

Lemma 1. *For the process* $\{b_l[t]\}$,

(i) $\mathbb{E}_{\mathcal{S}}\Big[b_l[t]\Big] = 0.$
(ii) $\mathbb{E}_{\mathcal{S}}\big[b_l[t]^2\big] \le \big(\lambda_l + 1\big) \max\Big\{W_{max}^2, 1/\eta_{min}^2\Big\}.$

where $\eta_{min} = \min_{1\le i\le K} \eta_i$.

The weight of a link based on its MAC queue or the total expected queue length differs by a constant when weight is chosen carefully as stated by the following Lemma.

Lemma 2. *Let* $f(x) = \frac{\log(1+x)}{g(x)}$, *then*

$$0 \le f(\bar{q}_l[t]) - f(q_l^{mac}[t]) \le c_1, \tag{6}$$

where $c_1 = \frac{\log(1+1/\eta_{min})}{g(0)}$.

Proof of Lemma 2: Because $q_l^{mac}[t] \le \bar{q}_l[t]$ and $f(x)$ is an increasing function, the first inequality is straight-forward. From the definition of $\bar{q}_l[t]$ in (3),

$$\begin{aligned} \bar{q}_l[t] &= \sum_{i=1}^{n_l[t]} \Big[\sigma_{li}[t]\eta_{li}[t] + q_{li}^{mac}[t]\Big] \\ &\le 1/\eta_{min} n_l[t] + q_l^{mac}[t] \le (1 + 1/\eta_{min}) q_l^{mac}[t] \end{aligned}$$

Therefore,

$$\begin{aligned} f(\bar{q}_l[t]) &\le f\Big((1 + 1/\eta_{min}) q_l^{mac}[t]\Big) \\ &= \frac{\log\Big(1 + (1 + 1/\eta_{min}) q_l^{mac}[t]\Big)}{g\Big((1 + 1/\eta_{min}) q_l^{mac}[t]\Big)} \\ &\le \frac{\log\Big((1 + 1/\eta_{min})(1 + q_l^{mac}[t])\Big)}{g\Big(q_l^{mac}[t]\Big)} \\ &\le f(q_l^{mac}[t]) + \frac{\log(1 + 1/\eta_{min})}{g(0)}. \end{aligned}$$

Letting $c_1 = \log(1 + 1/\eta_{min})/g(0)$ concludes the proof. ■

Let $F(q) = \int_0^q f(x)dx$. We define the Lyapunov function $V(\mathcal{S})$ as follows:

$$V(\mathcal{S}) = \sum_{l=1}^{L} F(\bar{q}_l). \tag{7}$$

Next, we calculate the Lyapunov drift

$$\begin{aligned}
&\mathbb{E}_{\mathcal{S}}\big[V(\mathcal{S}[t+1]) - V(\mathcal{S}[t])\big] \qquad (8)\\
=\ & \sum_{l=1}^{N}\Big(\mathbb{E}_{\mathcal{S}}\big[F(\bar{q}_l[t+1])\big] - F(\bar{q}_l[t])\Big)\\
=\ & \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\big[f(y_l[t])(\tilde{a}_l[t] - x_l[t] + u_l[t])\big]\\
=\ & \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\Big[\big(f(y_l[t]) - f(\bar{q}_l[t])\big)(\tilde{a}_l[t] - x_l[t])\Big]\\
+\ & \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\Big[f(\bar{q}_l[t])(\tilde{a}_l[t] - x_l[t])\Big]\\
+\ & \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\big[f(y_l[t])u_l[t]\big], \qquad (9)
\end{aligned}$$

where, by Mean Value Theorem, $y_l[t]$ is some value between $\bar{q}_l[t]$ and $\bar{q}_l[t+1]$ and $\tilde{a}_l[t] := a_l[t] + b_l[t]$.

The first term and the third term of (8) are bounded as stated by the following Lemmas.

Lemma 3. *There exists a positive constant c_2 such that, for all $\mathcal{S}[t]$, $\sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\big[f(y_l[t])u_l[t]\big] \le c_2$.*

Lemma 4. *Assume $\mathbb{E}\big[a_l[t]^2\big] < \infty$, then there exists a positive constant c_3 such that*

$$\sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\Big[\big(f(y_l[t]) - f(\bar{q}_l[t])\big)(\tilde{a}_l[t] - x_l[t])\Big] \le c_3, \forall t, \mathcal{S}[t].$$

Hence, letting $c_{23} = c_2 + c_3$, the Lyapunov drift can be bounded by

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}\big[\Delta V(\mathcal{S}[t])\big] &\le \sum_{l=1}^{N}\mathbb{E}_{\mathcal{S}}\Big[f(\bar{q}_l[t])(\tilde{a}_l[t] - x_l[t])\Big] + c_{23}\\
&= \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - \sum_{l\in\tilde{s}[t]} f(\bar{q}_l[t]) + c_{23}. \qquad (10)
\end{aligned}$$

Let $s^*[t]$ be the optimal scheduling algorithm which maximizes the following objective

$$s^*[t] \in \arg\max_{s\in\mathcal{M}}\sum_{l\in s} f(\bar{q}_l[t]). \qquad (11)$$

Then, based on definitions (2), (11), and Lemma 2, it can be shown that

$$0 \le \sum_{l\in s^*[t]} f(\bar{q}_l[t]) - \sum_{l\in\tilde{s}[t]} f(\bar{q}_l[t]) \le Nc_1. \qquad (12)$$

To complete the proof of Theorem 1, assume that the traffic load is strictly within the capacity region, i.e., there exists $\varepsilon > 0$ such that $\boldsymbol{\rho} \in \mathcal{C}/(1+\varepsilon)$. The capacity region is a N-dimensional polyhedron which is a convex hull of all possible schedules. In linear programming, one of the corner points of a linear optimization over the linear constraints is one of the optimal solutions. Hence, letting

$$\boldsymbol{\mu}^*[t] = \arg\max_{\boldsymbol{\mu}\in\mathcal{C}}\sum_{l=1}^{N} f\big(\bar{q}_l[t]\big)\mu_l, \qquad (13)$$

the following holds:

$$\sum_{l\in s^*[t]} f\big(\bar{q}_l[t]\big) = \sum_{l=1}^{N} f\big(\bar{q}_l[t]\big)\mu_l^*[t] \ge \sum_{l=1}^{N} f\big(\bar{q}_l[t]\big)\rho_l(1+\varepsilon).$$

Consider the largest sphere centered at the origin and tangent to the boundary of the capacity region $\mathcal{C}$. Let r^* be the radius of such a sphere. The radius r^* is determined by the capacity region uniquely and $r^* > 0$. Define

$$\boldsymbol{\mu}^r[t] = \left(\frac{f(\bar{q}_1[t])}{||f(\bar{\boldsymbol{q}}[t])||_2}r^*, \cdots, \frac{f(\bar{q}_N[t])}{||f(\bar{\boldsymbol{q}}[t])||_2}r^*\right), \qquad (14)$$

where $||.||_2$ is the Euclidian norm in $\mathbb{R}^N$. Note that $\boldsymbol{\mu}^r[t] \in \mathcal{C}$ by construction. From (12), the Lyapunov drift can be bounded as follows

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}[\Delta V(\mathcal{S}[t])] &\le \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - \sum_{l\in\tilde{s}[t]} f(\bar{q}_l[t]) + c_{23}\\
&\le \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - \sum_{l\in s^*[t]} f(\bar{q}_l[t]) + c_4\\
&\le \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - \sum_{l=1}^{N} f(\bar{q}_l[t])\frac{\mu_l^*[t]}{1+\varepsilon}\\
&\quad -\frac{\varepsilon}{1+\varepsilon}\sum_{l=1}^{N} f(\bar{q}_l[t])\mu_l^*[t] + c_4\\
&\le -\frac{\varepsilon}{1+\varepsilon}\sum_{l=1}^{N} f(\bar{q}_l[t])\mu_l^r[t] + c_4\\
&= -\frac{\varepsilon}{1+\varepsilon}r^*||f(\bar{\boldsymbol{q}}[t])||_2 + c_4,
\end{aligned}$$

where $c_4 = c_{23} + Nc_1$. Therefore, the Lyapunov drift will be negative when $\|\bar{\boldsymbol{q}}\|$[1] is sufficiently large, or it is sufficient that the number of files is sufficiently large. Specifically, consider any constant $\delta > 0$ and let

$$\mathcal{B} = \Big\{\mathcal{S} : \|\mathbf{n}\| \ge f^{-1}\left(\frac{1+\varepsilon}{\varepsilon r^*}(c_4+\delta)\right)\Big\}.$$

Then, for any $\mathcal{S} \in \mathcal{B}$, the Lyapunov drift is less than $-\delta$. Also it is easy to check that $\mathcal{B}^c$ contains only a finite set of states with finite drift. Therefore, the system is stable by the Foster-Lyapunov stability theorem [14].

C. Some Performance Results

Consider the simple wireless network of Figure 2 under the 1-hop interference model, which implies links sharing a node can not be active simultaneously. There are 22 distinct maximal schedules for this network. Each link has a unit link capacity. The distribution of files is a mixture of geometric

[1]When there is no subscript, $\|z\| = \|z\|_\infty = \max_i z_i = z_{max}$.

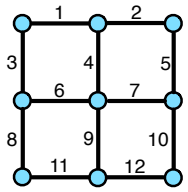

Fig. 2. A wireless network containing 12 communication links.

distributions as described next. Once a file is generated, the file size is geometrically distributed with mean 2 with probability $5/6$ and is geometrically distributed with mean 100 with probability $1/6$. All file arrivals are generated by Bernoulli processes. We say a file is short if its file size is less than half of the mean file size; otherwise, it is a long file. While this definition is somewhat arbitrary, our conclusions continue to hold if this assumption is removed. For simplicity, we divide the links into the following two sets: $\{4, 6, 7, 9\}$ and $\{1, 2, 3, 5, 8, 10, 11, 12\}$, where the links in each set have an equal arrival rate but the arrival rate of links in the second set is half of the arrival rate of links in the first set. We do simulations for different traffic intensities where the traffic intensity is a number such that the load vector divided by the traffic intensity lies on the boundary of the capacity region.

We consider a very naive window flow control algorithm under which the window size is always 1 for links $\{1, 4, 7, 10\}$, 2 for links $\{2, 5, 8, 11\}$ and 3 for links $\{3, 6, 9, 12\}$. We assume that MAC-layer packets are removed in a FIFO order. Once a packet is removed, a packet from the same file is injected from Transport layer to the tail of the MAC-layer FIFO queue as long as there are packets at the Transport layer. In addition to FIFO, we also do simulations for the case that MAC-layer receives one-bit information notifying whether the file is short or long. In this case, we give higher priority to short files. Each class (short or long) of files is served in a FIFO order. In the simulation, our scheduling algorithm chooses the weight of link l to be $f(q_l^{mac}[t])$ where $f(x) = \log(1 + x)$ by letting $g(x) = 1$. We compare our algorithm with the scheduling algorithm which uses $f(q_l[t])$ as the weight of link l.

Figure 3 shows the average delays of short files for different traffic intensities. For both FIFO and the short-file-first service disciplines for the MAC-layer packet transmission, the short-file delay performance of the scheduling algorithm which only uses the MAC-layer queue information is much better than the performance of the regular maximum weight scheduling that uses the total number of packets as the weight of a link. This is because a large total queue length at a link does not necessarily imply a large number of files at that link. It is possible that the scheduling algorithm chooses a link with large queue length but containing only a few files. Therefore, a link which contains many short files, but has only a small number of packets, will be scheduled infrequently and short files at such links suffer high latency. Also, as we expect, if the MAC layer knows 1-bit additional information to identify

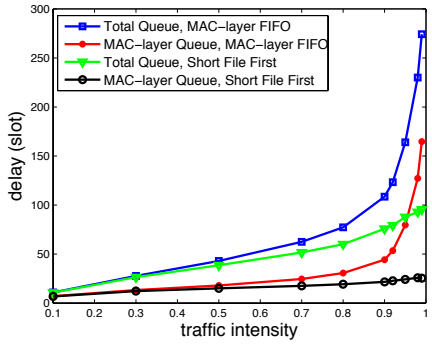

Fig. 3. Average delay of short files.

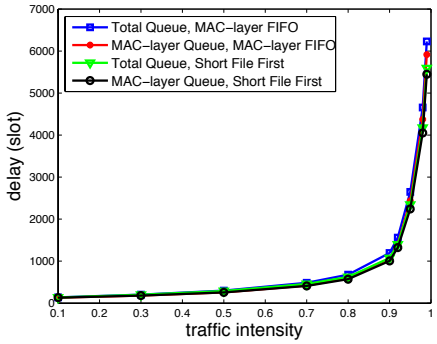

Fig. 4. Average delay of long files.

short files and uses short-file-first in-link scheduling, the delay performance will be even better.

Figure 4 shows the average delay performance of long files at various traffic intensities. As it is observed, the delay performance of long files is almost the same under both algorithms. However, for the same weight function, using 1-bit extra information can still yield a slightly better delay performance compared to the FIFO performance.

IV. Distributed Implementation of the Scheduling Algorithm

Recall that the optimal scheduling algorithm requires to find a maximum weight-type schedule at each time, i.e., needs to solve (2) at each time t. This is a formidable task, hence, in this section, we design a distributed algorithm version of the algorithm based on *Glauber Dynamics*.

A. Basic CSMA Algorithm

For our algorithm, based on the MAC layer information, we choose the weight of link l to be

$$\widetilde{w}_l[t] = \max\left(w_l[t], w_{min}[t]\right) \tag{15}$$

where

$$w_l[t] := f(q_l^{mac}[t]), \tag{16}$$

$$w_{min}[t] := \frac{\varepsilon}{2N} f(q_{max}^{mac}[t]), \tag{17}$$

and $q_{max}^{mac}[t]$ is the length of the largest MAC queue in the network at time t and assumed to be known. The function $f(x)$ is the same as defined for the centralized algorithm by (1).

Consider the conflict graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ of the network as defined earlier. Denote the neighbors of i by a set $\mathcal{N}(i) = \{k \in \mathcal{V} : (i,k) \in \mathcal{E}\}$. At each time slot t, a node i is chosen uniformly at random, with probability $\frac{1}{N}$, then

(i) If $x_j[t-1] = 0$ for all nodes $j \in \mathcal{N}(i)$, then $x_i[t] = 1$ with probability $\frac{\exp(\tilde{w}_i[t])}{1+\exp(\tilde{w}_i[t])}$, and $x_i[t] = 0$ with probability $\frac{1}{1+\exp(\tilde{w}_i[t])}$.
Otherwise, $x_i[t]$=0.

(ii) $x_j[t] = x_j[t-1]$ for all $j \neq i$.

The following Theorem states the main result regarding the throughput optimality of the basic CSMA algorithm.

Theorem 2. *Consider any $\varepsilon > 0$. The basic CSMA algorithm can stabilize the network for any $\boldsymbol{\rho} \in \mathcal{C}/(1+2\varepsilon)$, if the weight function is chosen to be in the form of $f(x) = \frac{\log(1+x)}{g(x)}$. The function $g(x)$ is a strictly increasing function chosen such that f is a strictly concave increasing function. In particular, the algorithm with the following weight functions is throughput-optimal: $f(x) = \frac{\log(1+x)}{\log(e+\log(1+x))}$ or $f(x) = (\log(1+x))^{1-\theta}$ for any $0 < \theta < 1$.*

B. Distributed Implementation

The basic algorithm is based on Glauber-Dynamics with one site update at each time. For distributed implementation, we need a randomized mechanism to select a link uniformly at each time slot. We use the Q-CSMA idea [11] to perform the link selection as follows. Each time slot is divided into a control slot and a data slot. The control slot, which is much smaller than the data slot, is used to generate a transmission schedule for the data slot. First, the network selects a set of links $m[t]$ that do not conflict with each other. Then, it performs the Glauber-Dynamics updates, in parallel, over links $m[t]$ to produce a transmission schedule $s[t]$ for data transmission. $m[t]$ is called the decision schedule at time t. For example, a simple randomized mechanism to generate $m[t]$ is as follows. In control slot t, each link l sends an INTENT message with probability $1/2$. If l does not hear any INTENT messages from its neighboring links $\mathcal{N}(l)$, it will be included in $m[t]$, otherwise it will not be included in $m[t]$. Therefore, by the end of the control slot, any feasible decision schedule $m[t] \subseteq \mathcal{M}$ could be selected with a positive probability $\alpha(m[t])$. 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 the activation probability for the current slot (based on its queue length).

To determine the weight at each link l, $q_{max}^{mac}[t]$ is needed. Instead, each link l can maintain an estimate of $q_{max}^{mac}[t]$. We can use the procedure suggested in [12] to estimate $q_{max}^{mac}[t]$, and use Lemma 2 of [12] to complete the stability proof. So we do not pursue this issue here. In practical networks $\frac{\varepsilon}{2N}\log(1+q_{max}^{mac})$ is small and we can use the weight function f directly, and thus, there may not be any need to know $q_{max}^{mac}[t]$.

Corollary 1. *Under the weight function f specified in Theorem 2, the distributed algorithm can stabilize the network for any $\boldsymbol{\rho} \in \mathcal{C}/(1+2\varepsilon)$.*

C. Proof of Throughput Optimality

Consider the basic CSMA algorithm over a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$. Assume that the weights are constants, i.e., the basic algorithm uses a weight vector $\tilde{W} = [\tilde{w}_1, \tilde{w}_2, ..., \tilde{w}_N]$ at all times. Then, the basic algorithm is essentially an irreducible, aperiodic, and reversible Markov chain (called Glauber Dynamics) to generate the independent sets of $\mathcal{G}$. So, the state space $\mathcal{M}$ consists of all independent sets of $\mathcal{G}$. Let $|\mathcal{V}| = N$. The stationary distribution of the chain is given by

$$\pi(s) = \frac{1}{Z}\exp(\sum_{i\in s}\tilde{w}_i); \quad s \in \mathcal{M}, \tag{18}$$

where Z is the normalizing constant.

The basic algorithm uses a time-varying version of the Glauber dynamics, where the weights change with time. This yields a time-inhomogeneous Markov chain but we will see that, for the choice of weights (15), it behaves similarly to the Glauber dynamics.

1) Mixing time of Glauber dynamics: For simplicity, we index the elements of $\mathcal{M}$ by $1, 2, ..., r$, where $r = |\mathcal{M}|$. Then, the eigenvalues of the corresponding transition matrix are ordered in such a way that

$$\lambda_1 = 1 > \lambda_2 \geq ... \geq \lambda_r > -1.$$

The convergence to steady state distribution is geometric with a rate equal to the *second largest eigenvalue modulus* (SLEM) of the transition matrix [13]. In fact, for any initial probability distribution μ_0 on $\mathcal{M}$, and for all $n \geq 1$,

$$\|\mu_0\mathbf{P}^n - \pi\|_{\frac{1}{\pi}} \leq (\lambda^*)^n\|\mu_0 - \pi\|_{\frac{1}{\pi}}, \tag{19}$$

where $\lambda^* = \max\{\lambda_2, |\lambda_r|\}$ is the SLEM. Note that, by definition, $\|z\|_{1/\pi} = \left(\sum_{i=1}^{r} z(i)^2\frac{1}{\pi(i)}\right)^{1/2}$.

The following Lemma gives an upper bound on the SLEM λ^* of Glauber dynamics.

Lemma 5. *For the Glauber Dynamics with the weight vector $\tilde{W}$ on a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with $|\mathcal{V}| = N$,*

$$\lambda^* \leq 1 - \frac{1}{16^N\exp(4N\tilde{w}_{max})},$$

where $\tilde{w}_{max} = \max_{i\in\mathcal{V}}\tilde{w}_i$.

See the appendix of [17] for the proof. We define *the mixing time* as $T = \frac{1}{1-\lambda^*}$, so

$$T \leq 16^N\exp(4N\tilde{w}_{max}) \tag{20}$$

Simple calculation, based on (19), reveals that the amount of time needed to get close to the stationary distribution is proportional to T.

2) A key lemma: At any time slot t, given the weight vector $\tilde{W}[t] = [\tilde{w}_1[t], ..., \tilde{w}_N[t]]$, the Max Weight-type algorithm should solve

$$\max_{s\in\mathcal{M}} \sum_{l\in s} \tilde{w}_l[t],$$

instead, our algorithm tries to simulate a distribution

$$\pi_t(s) = \frac{1}{Z}\exp(\sum_{l\in s}\tilde{w}_l[t]); \quad s \in \mathcal{M}, \tag{21}$$

i.e., the stationary distribution of Glauber dynamics with the weight vector $\tilde{W}[t]$ at time t.

Let P_t denote the transition probability matrix of Glauber dynamics with the weight vector $\tilde{W}[t]$. Also let μ_t be the true probability distribution of the inhomogeneous-time chain, over the set of schedules $\mathcal{M}$, at time t. Therefore, we have $\mu_t = \mu_{t-1}P_t$. Let π_t denote the stationary distribution of the time-homogenous Markov chain with $P = P_t$ as in (21). By choosing proper w_{min} and $f(\cdot)$, we aim to ensure that μ_t and π_t are close enough, i.e., $\|\pi_t - \mu_t\|_{TV} \leq \delta$ for some δ arbitrary small. Let $w_{max}[t] = f(q^{mac}_{max}[t])$. The following lemma gives a sufficient condition under which the probability distribution of the inhomogeneous Markov chain is close to the stationary distribution of the homogenous chain.

Lemma 6. *Given any $\delta > 0$, $\|\pi_t - \mu_t\|_{TV} \leq \delta/4$ holds when $\|\boldsymbol{q}^{mac}[t]\| \geq q_{th} + t^*$, if there exists a q_{th} such that*

$$\alpha_t T_{t+1} \leq \delta/16 \text{ whenever } \|\boldsymbol{q}^{mac}[t]\| > q_{th}, \tag{22}$$

where

(i) $\alpha_t = 2Nf'(f^{-1}(w_{min}[t+1]) - 1)$
(ii) $T_t \leq 16^N \exp(4Nw_{max}[t])$
(ii) t^* *is the smallest t such that*

$$\sum_{k=t_1:\|q[t_1]\|=q_{th}}^{t_1+t^*} \frac{1}{T_k^2} \geq \log(4/\delta) + N(f(q_{th}) + \log 2)/2. \tag{23}$$

In the above Lemma, condition (ii) is based on the upper bound of (20) and the fact that $\tilde{w}_{max}[t] = w_{max}[t]$. See the appendix of [17] for the proof of the above Lemma. In other words, Lemma 6 states that when queue lengths are large, the observed distribution of the schedules is close to the desired stationary distribution. The key idea is that the weights change at the rate α_t while the system responds to these changes at the rate $1/T_{t+1}$. Condition (i) is to make the weight dynamics slow enough compared to response time of the chain such that it remains close to its equilibrium (stationary distribution).

3) Throughput optimality: It turns out that weight functions of the form $f(x) = \frac{\log(1+x)}{g(x)}$ where $g(x)$ is a strictly increasing function, chosen such that f is an increasing concave function, satisfy the requirements of Lemma 6, and hence yield a maximum throughput algorithm. See [17] for more details.

Roughly speaking, since the mixing time T is exponential in w_{max}, $f'(f^{-1}(w_{min}))$ must be in the form of $e^{-w_{min}}$; otherwise it will be impossible to satisfy $\alpha_t T_{t+1} < \delta/16$ for any arbitrarily small δ as $\|\boldsymbol{q}^{mac}[t]\| \to \infty$. The only function with such a property is the $\log(\cdot)$ function. In fact, it turns out that f must grow slightly slower than $\log(\cdot)$, as was shown in [17], to satisfy (22), and to ensure the existence of a finite t^* in Lemma 6. For example, by choosing functions that grow much slower than $\log(1+x)$, like $g(x) = \log(e+\log(1+x))$, we can make $f(x)$ behave approximately like $\log(1+x)$ for large ranges of x.

Next, the following Lemma states that, with high probability, the basic CSMA algorithm chooses schedules that their weights are close to the Max Wight schedule.

Lemma 7. *The basic CSMA algorithm has the following property: Given any $0 < \varepsilon < 1$ and $0 < \delta < 1$, there exists a $B(\delta,\varepsilon) > 0$ such that whenever $\|\boldsymbol{q}^{mac}[t]\| > B(\delta,\varepsilon)$, with probability larger than $1-\delta$, it chooses a schedule $s[t] \in \mathcal{M}$ that satisfies*

$$\sum_{l\in s[t]} w_l[t] \geq (1-\varepsilon)\max_{s\in\mathcal{M}}\sum_{l\in s} w_l[t].$$

Proof: Let $w^*[t] = \max_{s\in\mathcal{M}}\sum_{l\in s} w_l[t]$ and define

$$\chi_t := \{s \in \mathcal{M} : \sum_{l\in s} w_l[t] < (1-\varepsilon)w^*[t]\}$$

Therefore, we need to show that $\mu_t(\chi_t) \leq \delta$, for $\|\boldsymbol{q}^{mac}[t]\|$ large enough. For our choice of $f(\cdot)$ and w_{min}, it follows from Lemma 6 that, whenever $\|\boldsymbol{q}^{mac}[t]\| > q_{th} + t^*$, $2\|\mu_t - \pi_t\|_{TV} \leq \delta/2$, and consequently,

$$\sum_{s\in\chi_t}\mu_t(s) \leq \sum_{s\in\chi_t}\pi_t(s) + \delta/2.$$

Therefore, to ensure that $\sum_{s\in\chi_t}\mu_t(s) \leq \delta$, it suffices to have $\sum_{s\in\chi_t}\pi_t(s) \leq \delta/2$. But $\widetilde{w}_i[t] \leq w_i[t] + w_{min}[t]$, So,

$$\begin{aligned}\sum_{s\in\chi_t}\pi_t(s) &\leq \sum_{s\in\chi_t}\frac{1}{Z_t}\exp(\sum_{i\in s} w_i[t])\exp(|s|w_{min}[t]) \\ &\leq \sum_{s\in\chi_t}\frac{1}{Z_t}\exp((1-\varepsilon)w^*[t])\exp(Nw_{min}[t])\end{aligned}$$

and

$$Z_t = \sum_{s\in\mathcal{M}}\exp(\sum_{i\in s}\widetilde{w}_i[t]) > \sum_{s\in\mathcal{M}}\exp(\sum_{i\in s} w_i[t]) > e^{w^*[t]}.$$

Therefore,

$$\sum_{s\in\chi_t}\pi_t(s) \leq 2^N\exp(Nw_{min}[t] - \varepsilon w^*[t])$$

and $w^*[t] \geq w_{max}[t]$. So, it suffices to have

$$2^N\exp(Nw_{min}[t] - \varepsilon w_{max}[t]) \leq \delta/2$$

when $\|\boldsymbol{q}^{mac}[t]\| > q_{th} + t^*$. The choice of $w_{min}[t] = \frac{\varepsilon}{2N}w_{max}[t]$, satisfies the above condition for $\|\boldsymbol{q}^{mac}[t]\| > B(\delta,\varepsilon)$, where

$$B(\delta,\varepsilon) = \max\left\{q_{th} + t^*, f^{-1}\left(\frac{N\log 2 + \log\frac{2}{\delta}}{\varepsilon/2}\right)\right\}. \tag{24}$$

■

Now we are ready to prove the throughput optimality for the basic CSMA algorithm. The proof is parallel to the argument for the throughput optimality of the Max Weight-type algorithm. Especially, the inequality (10) still holds, i.e.,

$$\begin{aligned}
& \mathbb{E}_{\mathcal{S}}[\Delta V(\mathcal{S}[t])] \\
\leq & \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - \mathbb{E}_{\mathcal{S}}[\sum_{l\in s[t]} f(\bar{q}_l[t])] + c_{23} \\
\leq & \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - (1-\delta)(1-\varepsilon)\sum_{l\in \tilde{s}[t]} f(\bar{q}_l[t]) + c_{23} \\
\leq & \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - (1-\delta)(1-\varepsilon)\sum_{l\in s^*[t]} f(\bar{q}_l[t]) + c_4
\end{aligned}$$

whenever $\|\boldsymbol{q}^{mac}[t]\| > B(\delta,\varepsilon)$. The second inequality is by Lemma 7 and the last inequality is due to (12). Recall that $c_4 = c_{23} + Nc_1$ as defined before. Since $\boldsymbol{\rho}$ is strictly inside the capacity region, there exists $\varepsilon > 0$ such that $\boldsymbol{\rho} \in \mathcal{C}/(1+2\varepsilon)$. Since, for any fixed but arbitrary small ε, δ (in Lemma 7) can be made arbitrary small, we choose δ sufficiently small such that

$$(1-\delta)(1-\varepsilon) - \frac{1}{1+2\varepsilon} = \varepsilon' > 0.$$

Therefore, based on definitions (13) and (14), we have

$$\begin{aligned}
\mathbb{E}_{\mathcal{S}}[\Delta V(\mathcal{S}[t])] \leq & \sum_{l=1}^{N} f(\bar{q}_l[t])\rho_l - \sum_{i=1}^{N} f(\bar{q}_l[t])\frac{\mu_l^*[t]}{1+2\varepsilon} \\
& -\varepsilon' \sum_{i=1}^{N} f(\bar{q}_l[t])\mu_l^*[t] + c_4 \\
\leq & -\varepsilon' r^* \|f(\bar{\boldsymbol{q}}[t])\|_2 + c_4
\end{aligned}$$

for a positive constant r^* determined uniquely by the capacity region $\mathcal{C}$ (see the corresponding proof for the MWS-type algorithm). So the drift will be negative for sufficiently large $\|f(\bar{\boldsymbol{q}}[t])\|_2$. In particular, to get negative drift, $-\delta_1$, for some positive constant δ_1, it suffices that $\|\mathbf{n}[t]\| > \max\{f^{-1}(\frac{c_4+\delta_1}{\varepsilon' r^*}), B(\delta,\epsilon)\}$ because $\|f(\bar{\boldsymbol{q}}[t])\|_2 \geq \|f(\bar{\boldsymbol{q}}[t])\|$, $\|\bar{\boldsymbol{q}}[t]\| \geq \|\boldsymbol{q}^{mac}[t]\| \geq \|\mathbf{n}[t]\|$, and f is an increasing function. This concludes the proof of the main theorem.

4) Extension of the proof to the distributed implementation: The distributed algorithm is based on multiple site-update (or parallel operating) Glauber dynamics as defined next. Consider the graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ as before and a constant weight vector $\tilde{W} = [\tilde{w}_1, \tilde{w}_2, ..., \tilde{w}_N]$. At each time t, a decision schedule $m[t] \subseteq \mathcal{M}$ is selected at random with positive probability $\alpha(m[t])$. Then, for all $i \in m[t]$, we perform the regular Glauber dynamics. Then, it can be shown that the Markov chain is reversible, it has the same stationary distribution as the regular Glauber dynamics in (21), and its mixing time is almost the same as (20). The rest of the analysis is the same as the argument for the basic algorithm. See [17] for more details. Let D and W denote the lengths of the data slot and the control slot. Thus, the distributed algorithm can achieve a fraction $\frac{D}{D+W}$ of the capacity region. In particular, recall the simple randomized machanism, in section IV-B, where each node joins the decison schedule by sending an INTENT message with probability $1/2$. Note that it sufficies to allocate a short mini-slot at the begining of the slot for the purpose of control. By choosing the data slot to be much larger than the control slot, the algorithm can approach the full capacity.

V. CONCLUSIONS

Since the scheduling algorithm is part of MAC, it is desirable to design an algorithm that uses only the MAC information. We showed that it is possible to design such algorithms that are still throughput optimal. Another advantage of such algorithms is that the long files do not block the transmission of short files and hence, the overall delay performance can be improved. The key element of the algorithm is an appropriate choice of a weight function. Interestingly, by using such weight functions, we can design a distributed version of the algorithm with proven throughput optimality. In the distributed algorithm, each node only needs to know its own MAC information and its carrier sensing information.

REFERENCES

[1] L. Tassiulas and A. Ephremides, Stability properties of constrained queueing systems and scheduling algorithms for maximal throughput in multihop radio networks, *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936-1948, 1992.

[2] X. Lin, N. Shroff, and R. Srikant, On the connection-level stability of congestion-controlled communication networks, *IEEE Transactions on Information Theory*, vol. 54, no. 5, pp. 2317-2338, 2008.

[3] T. Bonald, S. Borst, and A. Proutiere, How mobility impacts the flow-level performance of wireless data systems, *Proc. IEEE INFOCOM*, vol. 3, pp. 1872-1881, 2004

[4] J. Liu, A. Proutiere, Y. Yi, M. Chiang, and V. Poor, flow-level stability of data networks with non-convex and time-varying rate regions, *Proc. ACM SIGMETRICS*, 2007

[5] P. van de Ven, S. Borst, and S. Shneer, Instability of MaxWeight scheduling algorithms, *Proc. IEEE INFOCOM*, pp. 1701-1709, 2009

[6] S. Liu, L. Ying and R. Srikant, Throughput-optimal opportunistic scheduling in the presence of flow-level dynamics, *Proc. INFOCOM*, 2010.

[7] T. Bonald and M. Feuillet, On the stability of flow-aware CSMA, *Performance Evaluation*, vol. 67, no. 11, pp. 1219-1229, 2010.

[8] T. Ji and R. Srikant, Scheduling in wireless networks with connection arrivals and departures, Information Theory and Applications Workshop, 2011.

[9] M. Crovella and A. Bestavros, Self-Similarity in World Wide Web traffic: evidence and possible causes, *IEEE/ACM Transactions on Networking*, Vol. 5, No. 6, pp. 835-846, 1997.

[10] J. Ghaderi and R. Srikant, On the design of efficient CSMA algorithms for wireless networks, *IEEE Conference on Decision and Control*, 2010.

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

[12] S. Rajagopalan, D. Shah and J. Shin, Network adiabatic theorem: an efficient randomized protocol for contention resolution, *ACM SIGMETRICS/Performance*, pp. 133-144, 2009.

[13] P. Bremaud, Markov chains, Gibbs fields, Monte Carlo simulation, and queues, *Springer-Verlag*, New York 1999, 2nd edition, 2001.

[14] S. Asmussen, Applied probability and queues, Springer, 2003.

[15] L. Jiang and J. Walrand, A distributed CSMA algorithm for throughput and utility maximization in wireless networks, *46th Annual Allerton Conference on Communication, Control and Computing*, 2008.

[16] A. Proutiere, Y. Yi, and M. Chiang, Throughput of random access without message passing, *Proc. of CISS*, Princeton, 2008.

[17] J. Ghaderi, T. Ji, and R. Srikant, Connection-level scheduling in wireless networks using only MAC-layer information, *Technical Report*, avaiable online at http://www.ifp.illinois.edu/~jghaderi/MACscheduling.pdf