2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

The Stability of Longest-Queue-First Scheduling With Variable Packet Sizes

Siva Theja Maguluri, Bruce Hajek and R. Srikant

Abstract— **It is well known that the MaxWeight scheduling algorithm is throughput-optimal in wireless networks. However, its complexity is exponential in the number of links in an ad hoc network. In this work, we consider a greedy variant of the MaxWeight algorithm, called Longest Queue First (LQF). A synchronous version of LQF is known to be throughput-optimal under a topological condition called local pooling. Here we study an asynchronous version of LQF which is suitable for implementation in networks with variable packet sizes. We show that asynchronous LQF is also throughput-optimal under the local pooling condition.**

I. Introduction

We consider the problem of scheduling in an ad hoc wireless network. An ad hoc network consists of a collection of wireless nodes with no infrastructure for centralized coordination of scheduling decisions. Here we only consider single-hop transmissions, i.e., a sender and a receiver directly communicating without any intermediate relays. A link in such a network refers to a transmitter-receiver pair. Not all the links can be simultaneously active because of interference. These constraints are represented by an *interference graph*. Vertices in the interference graph correspond to the links. If there is an edge between two vertices, then the corresponding links interfere and so cannot transmit at the same time. An example is shown in Figure 1.

Packets arrive to be transmitted over the links, and are queued. Given the queue lengths at each link, a scheduling algorithm has to choose a set of links that can transmit at each given time, without violating interference constraints. In other words, at any given time, the scheduler should choose an independent set from the interference graph.

A wireless network is said to be stable if the queues in the network are finite (to be defined more precisely later). A scheduling algorithm is throughput-optimal if it can stabilize the system for all sets of arrival rates that are stabilizable under some algorithm. Thus, loosely speaking, a throughput-optimal algorithm is able to sustain the maximum possible throughput in the network. Throughput-optimality is a natural performance criterion to evaluate a scheduling algorithm.

A well-known throughput-optimal algorithm is the maxweight algorithm: each link is associated with a weight which is a function of the queue length, usually the queue length itself, and a schedule with the maximum weight is chosen in each time slot from all possible schedules. Tassiulas and Ephremides [1] have shown that the MaxWeight scheduling algorithm is throughput-optimal under the assumption that time is slotted and synchronized across links. However its complexity increases exponentially with the number of nodes, and so, it is difficult to implement. Moreover, it cannot be implemented in a distributed fashion.

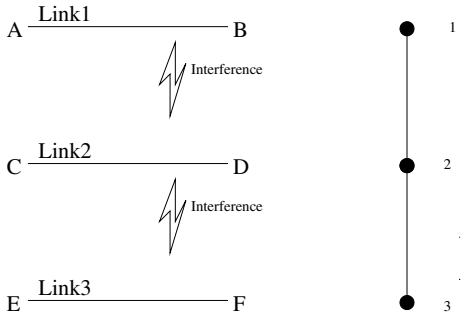

Fig. 1. Interference constraints for six users and three links and the corresponding interference graph

Another class of scheduling algorithms are CSMA (carrier sense multiple access) type random access algorithms. Under CSMA, a node will sense whether the channel is busy before it transmits a packet. If it detects that the channel is busy, it will wait for a random back-off time. Since CSMA-type algorithms can be easily implemented in a distributed manner, they are widely used in practice (e.g., the IEEE 802.11 MAC protocol). Although the recent results on CSMA-type random access algorithms show throughput-optimality, simulation results indicate that the delay performance of these algorithms can be significantly worse than that of the MaxWeight algorithm under certain traffic conditions [2], [3], [4], [5], [6].

Another alternative is a greedy approximation of MaxWeight, viz., Longest Queue First (LQF). A link with the longest queue is first added to the schedule, and all the links interfering with it are removed, and this process is recursively repeated till a maximal schedule is obtained. Ties are broken at random.

LQF scheduling algorithm has very good performance for a variety of network scenarios in simulations and experiments. When time is slotted, Dimakis and Walrand [7] have shown that LQF is throughput-optimal under a topological constraint called local pooling. Several classes of graphs such as trees, trees of cliques, perfect graphs, chordal graphs, satisfy local pooling [8], [9], [10].

In all practical networks, packets have variable sizes. However, one can segment the packets at the MAC (Medium

Research supported by AFOSR Grant FA-9550-08-1-0432, ARO MURI W911NF-08-1-0233, ARO MURI W911NF-07-1-0287, and AFOSR MURI FA9550-10-1-0573.

The authors are with the Department of Electrical and Computer Engineering, and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, IL 61801 USA. (e-mail: magulur1,b-hajek,rsrikant@illinois.edu)

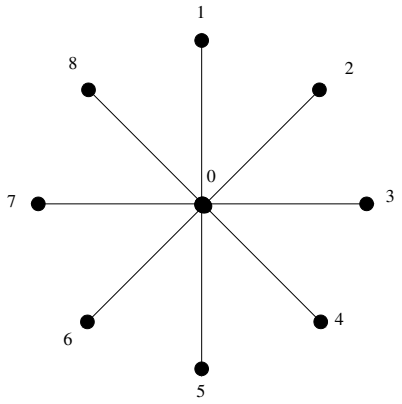

Fig. 2. A star interference graph

Access Control) layer to be of equal size and implement LQF. But this could lead to packet fragmentation and necessitate reassembly at the receiver. Therefore, it is interesting to investigate if LQF can be implemented directly on the original packets. The problem with variable packet sizes is that when a packet transmission is completed, other packets in the network will, in general, be in the middle of their transmission. So it is difficult to implement LQF which requires sequential scheduling of links, starting with the most congested links first. It should be noted that practical scheduling algorithms such as the widely-used 802.11 suite of protocols allow variable packet sizes.

A natural alternative is to do longest queue first scheduling among the idle links, i.e., those that are not transmitting, whenever some link finishes transmitting a packet, without disturbing other transmitting links. In other words, include the link with the longest queue among those that can be allowed to transmit respecting interference constraints. But it is not clear if this is stable. As an example, consider a star interference graph with one link in the middle interfering with many other links as shown in Figure 2. Under this algorithm, once any of the outer links start transmitting, the middle link will not get a chance unless all the outer links empty their queues. But once the middle link gets a chance, it transmits till its queue length is almost the same as the others. Due to such extreme oscillations in behavior, at this point, we are unable to determine if this algorithm is stable or not. So, we introduce a small exponential delay between the time a link finishes transmission and the next time any other link is scheduled, so that there is finite probability that more links finish transmission in this wait time. This is modeled by having a centralized system clock that goes off every $\exp(1/\kappa)$ seconds, and new links are added to the schedule only at these clock tick times, where $\kappa > 0$ can be made arbitrarily close to zero. Such a centralized clock can be implemented by using a common random generator seed at all links. We will see that with this wait period, asynchronous LQF is throughput-optimal.

This paper is organized as follows. We will define the system and explain the notation in the next section. In section III, we will prove throughput-optimality of asynchronous LQF. We will do this by defining fluid limit of the system, showing that the fluid limit exists. We will then show stability using the fluid limit. In section IV, we will present some simulation results studying the importance of the wait period for asynchronous LQF. We then present a short discussion about a distributed implementation of this algorithm.

II. System, Model and Notation

In this section, we will describe a model for the wireless network and explain the scheduling algorithm.

A. System

Consider a network of links, indexed from a set K. The interference constraints are represented by an interference graph. At any given time, the set of transmitting links should form an independent set in this graph. A schedule is a binary vector of length $|K|$, with 1's corresponding to the links that are allowed to transmit. Let $M[K]$ be the set of all maximal schedules of K, which correspond to the maximal independent sets in the interference graph. Let $\text{Co}(M[K])$ be the convex hull of $M[K]$. Let $A_i(t)$ be the cumulative arrival process, i.e., the total number of arrivals, to link i up to time t. It is assumed to be a Poisson process with rate λ_i. Thus

$$A_i(t) = N_{1,i}(\lambda_i t),$$

where $N_{1,i}$ for $i \in K$ are independent Poisson processes of unit rate. Similarly $D_i(t)$ is the cumulative departure process from link i. Then the queue length of the i^{th} link at time t is given by

$$Q_i(t) \quad = \quad Q_i(0) + A_i(t) - D_i(t). \tag{1}$$

Define

$$\begin{aligned} Q(t) &= (Q_1(t), ...Q_{|K|}(t)) \\ A(t) &= (A_1(t), ..., A_{|K|}(t)) \\ D(t) &= (D_1(t), ..., D_{|K|}(t)) \\ N_1(\lambda t) &= (N_{1,1}(\lambda_1 t), N_{1,2}(\lambda_2 t), ..., N_{1,|K|}(\lambda_{|K|} t)). \end{aligned}$$

Packet lengths are assumed to be exponentially distributed with mean $1/\mu_i$ at link i. Packet lengths are assumed to be mutually independent and independent of the arrival processes. Thus, when a link i is scheduled, packets depart at rate μ_i. Let $\mu = (\mu_1, \mu_2, ...\mu_K)$. The set of all pairs (λ, μ) so that there is some scheduling policy under which the system is stable is called the *capacity region*. We say that the system is stable if

$$\lim_{C \to \infty} \limsup_{t \to \infty} P(|Q(t)| \geq C) = 0 \tag{2}$$

where $|Q(t)|$ is the total queue size in the network. Let $\mathcal{C}$ be the set of all pairs (λ, μ) for which there is a $\phi \in \text{Co}(M[K])$ such that $\frac{\lambda}{\mu} < \phi$ where $\lambda/\mu < \phi$, means $\lambda_i/\mu_i < \phi_i$ for all i. We will use this notation throughout the paper when comparing vectors. The following proposition establishes that $\mathcal{C}$ is an outer bound on the capacity region. We will

show in the next section that asynchronous LQF algorithm stabilizes the system for any $(\lambda, \mu) \in \mathcal{C}$. Thus $\mathcal{C}$ is the capacity region.

Proposition 1: No scheduling policy can stabilize the system if $(\lambda, \mu) \in \mathcal{C}^c$.

We skip the proof, as it follows along the same lines of a similar proof in [1], Lemma 3.3.

B. Algorithm and Model

There is a centralized scheduling clock. Scheduling is done only when this clock ticks, so the tick times of this clock are called scheduling times. At each scheduling time, the clock is reset by $\exp(1/\kappa)$, an exponentially distributed amount of time with mean κ. Let $C(t)$ be the last scheduling time before or equal to t.

When a link finishes transmitting a packet, it stops transmission, and waits for the next scheduling time. At a scheduling time, a link with the longest queue that does not interfere with any of the transmitting links is included in the schedule. This process is done recursively i.e., if there are more links that can be added to the schedule without interfering with the existing links, the largest among them is included in the schedule. When a link is chosen in a schedule, it turns ON if it has non-zero queue length, transmits one packet, and then turns OFF. It remains in the OFF state till the next scheduling time. We will call this time duration till the next scheduling time, the *wait period* of a link.

Define $S_i(t)$ to be a binary function showing the ON-OFF state of link i with 1 for ON state and 0 for OFF state. Define $T_m(t) \geq 0$ to be the cumulative time the schedule m was chosen up until time t. Note that during a time period when a schedule is chosen, not all of the links in that schedule are ON because some links may have finished transmission and are waiting for the next scheduling time and some links may not have any packets to transmit.

Recall that packet transmission durations at link i are assumed to be exponentially distributed with mean $1/\mu_i$, so the cumulative departure process is a Poisson process as long as the link is ON. Assuming $\{N_{2,i}(t) : i \in K\}$ to be a set of independent Poisson processes with rate 1, which are also independent of the arrival processes, we have

$$D_i(t) \quad = \quad N_{2,i}(\mu_i \int_0^t S_i(s)ds). \tag{3}$$

Let $N_2(\mu t) = (N_{2,1}(\mu_1 t), N_{2,2}(\mu_2 t), ...N_{2,|K|}(\mu_{|K|} t))$. At a scheduling time, all the links with nonzero queue lengths that are scheduled, are ON. So when $t = C(t)$,

$$S_i(t) = \begin{cases} \sum_{m \in M[K]} m_i \ \frac{dT_m}{dt}\big|_{t+} & \text{if } Q_i(t) > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\frac{dT_m}{dt}\big|_{t+}$ is the right derivative of $T_m(t)$ at t and is 1 only for the schedule that is chosen at $t = C(t)$ while it is 0 for all other maximal schedules. For scheduled links, $S_i(t)$ changes to 0 (OFF) when there is a departure. It remains zero for unscheduled links. So for all t,

$$S_i(t) \quad = \quad S_i(C(t)) - [D_i(t) - D_i(C(t))] \, .$$

Moreover, only a maximal schedule is chosen at any time. So,

$$\sum_{m \in M[K]} T_m(t) = t. \tag{4}$$

We say that (λ, μ) is *κ-feasible* if there is a $\phi \in \text{Co}(M[K])$ such that $\frac{\lambda_i}{\mu_i} + \kappa\lambda_i < \phi_i$ for all $i \in K$.

Let $U_i(t) = \int_0^t S_i(C(\tau))d\tau$. In words, between any two scheduling times, either U_i increases at rate one or it is constant. It increases at rate one between two scheduling times if link i is in the ON state at the first scheduling time. Intuitively, $U_i(t)$ is the amount of time that the server for link i is called upon, including the leftover bits of scheduling intervals after service completion times. Note that, if t_1 is a scheduling time and $Q_i(t) > 0$ for all $t \in (t_1, t_2)$,

$$U_i(t_2) - U_i(t_1) = \sum_{m \in M[K]} m_i \left(T_m(t_2) - T_m(t_1)\right). \tag{5}$$

Let $Y_i(u) = D_i(U_i^{-1}(u))$ for $u \geq 0$, where $U_i^{-1}(u) = \min\{t : U_i(t) \geq u\}$. Call Y_i the *service yield process* for link i. In words, $Y_i(u)$ is the number of service completions at link i when the total amount of time the link was scheduled and has non zero queue (this includes the amount of time the link was in the ON state and the time it was waiting for a scheduling time after service completion within a scheduling interval) reaches u. The service yield processes for different links are dependent in a complicated way, due to correlations induced by the scheduling policy and the fact that the scheduling times are global. However, for fixed i, the distribution of the random process $(Y_i(u) : u \geq 0)$ does not depend on the arrival process or scheduling policy. It is the same as if i were the only link in the network and the queue at link i had an infinite backlog. $Y_i(u)$ is a counting process and the distribution of time between two consecutive increments is a sum of two exponential distributions (one corresponding to service completion i.e., link being in ON state and the other corresponding to the time waiting for the next scheduling time). Specifically, $Y_i(u)$ is a renewal process with rate $\frac{1}{\kappa + 1/\mu_i}$.

Since the arrival process is a Poisson process and packet lengths and scheduling times are exponentially distributed, the system is a Markov chain with state $X = (Q, S)$. Therefore, stability of this system is equivalent to positive recurrence of the underlying Markov chain.

C. Local Pooling

We will show throughput-optimality of asynchronous LQF when the interference graph satisfies a condition called local pooling[7].

Definition 1: A set of links $L \subset K$ is said to satisfy *local pooling* if there exists a nonzero vector $\alpha \in \mathbb{R}_+^K$ such that $\alpha^T \phi$ is a positive constant for all $\phi \in \text{Co}(M[L])$. We say

that local pooling is satisfied if every subset of K satisfies local pooling.

Remark 1: If $L \subset K$ satisfies local pooling, then there are no two vectors $\tilde{\phi}, \hat{\phi} \in \text{Co}(M[L])$ such that $\tilde{\phi} > \hat{\phi}$.

Lemma 1: If $L \subset K$ satisfies local pooling, for any κ-feasible (λ, μ) and for any $\phi \in \text{Co}(M[L])$, there is a $k \in L$ such that $\frac{\lambda_k}{\mu_k} + \kappa\lambda_k < \phi_k$.

Proof: If this were not true, then we have a $\phi \in \text{Co}(M[L])$ such that $\frac{\lambda_k}{\mu_k} + \kappa\lambda_k \geq \phi_k$ for all $k \in L$. Also, there is a $\tilde{\phi} \in \text{Co}(M[L])$ such that $\frac{\lambda_k}{\mu_k} + \kappa\lambda_k < \tilde{\phi}_k$ for all $k \in L$ since (λ, μ) is κ-feasible. This means $\tilde{\phi}_k > \phi_k$ for all $k \in L$, contradicting local pooling. ∎

When local pooling is satisfied, for a fixed κ-feasible pair (λ, μ), define

$$\epsilon_* = \inf_{L \subset K} \left\{ \inf_{\phi \in \text{Co}(M[L])} \left\{ \max_{k \in L} (\phi_k - \frac{\lambda_k}{\mu_k} - \kappa\lambda_k) \right\} \right\}.$$

From Lemma 1, we have that $\epsilon_* > 0$ since we have a maximization over k. So, for any (λ, μ) that is κ feasible and $L \subset K$ and $\phi \in \text{Co}(M[L])$, there is a $k \in L$ such that $(\phi_k - \frac{\lambda_k}{\mu_k} - \kappa\lambda_k) \geq \epsilon_*$.

III. Throughput-Optimality of Asynchronous LQF

In this section, we will show that the Markov chain describing the system is positive recurrent as long as (λ, μ) lies within a region that is slightly smaller (depending on κ) than $\mathcal{C}$. One way to show positive recurrence is using the idea of fluid limits as shown in [11]. We will first show that the fluid limit exists and satisfies certain properties. We will then use these properties of fluid limits to show positive recurrence of the Markov chain.

A. Fluid Limits

We need the following definitions and lemma to show the existence of fluid limit.

Definition 2: A deterministic trajectory $q = (q_i(t) : t \geq 0, i \in K)$ satisfies the *κ-LQF constraint* if q is absolutely continuous and

$$\left(-(\kappa + 1/\mu_i) \frac{(q_i(b) - q_i(a)) - (b-a)\lambda_i}{(b-a)} \right)_{i \in A} \in Co(M(A))$$

whenever

$$0 \leq a < b, \ A \subset K, \text{ such that}$$
$$q_i(t) > q_j(t) \text{ for } a \leq t \leq b, i \in A, j \in K \setminus A$$
$$\text{and } q_i(t) > 0 \text{ for } i \in A.$$

A set of links L is said to be a *dominating set* at time t if $Q_i(t) > Q_j(t)$ for all $i \in L, j \in K \setminus L$.

Definition 3: A schedule is called an LQF schedule if its restriction to any dominating set $L \subset K$ is maximal in $M[L]$.

A scheduling time instant t is called a *reset time* if all the links in the network are in the OFF state just before time t. Note that an LQF schedule is chosen at any scheduling time that is a reset time.

Lemma 2: Given $C > 0$, the probability that there is no reset time in a time interval of duration C decays exponentially in C. In other words, there exists $\alpha, \beta > 0$ so that for any time t, the probability that there is no reset time in the interval $[t, t+C]$ is less than or equal to $\beta e^{-\alpha C}$.

Proof: Consider a fixed time t. Let $N_0 = \lfloor C/2\kappa \rfloor$ where $\lfloor . \rfloor$ is the floor function.

Let $\overline{E}_1$ be the event that there are fewer than N_0 scheduling points in $[t, t+C]$. Let $\overline{E}_2$ be the event that there is no reset time among the first N_0 scheduling intervals. Then, $P(\overline{E_1} \cup \overline{E_2})$ is an upper bound on the probability that there is no reset time in the interval $[t, t+C]$.

Since the time between any two scheduling times is exponentially distributed, the number of scheduling times within a time interval of length C is a Poisson random variable with mean C/κ. Therefore, $P(\overline{E}_1) = p_1 = \sum_{n=0}^{N_0} e^{-C/\kappa} \frac{(C/\kappa)^n}{n!}$. Then

$$\begin{aligned} \frac{p_1}{2^{N_0}} &= e^{-C/\kappa} \sum_{n=0}^{N_0} \frac{(C/\kappa)^n}{n!} \frac{1}{2^{N_0}} \\ &\leq e^{-C/\kappa} \sum_{n=0}^{N_0} \frac{(C/\kappa)^n}{n!} \frac{1}{2^n} \\ &\leq e^{-C/\kappa} \sum_{n=0}^{\infty} \frac{(C/2\kappa)^n}{n!} \\ &= e^{-C/2\kappa}. \end{aligned}$$

Thus, we have $p_1 \leq (e/2)^{-C/2\kappa}$, which is of the form $e^{-\alpha_1 C}$ for some $\alpha_1 > 0$.

Resets in different scheduling intervals are dependent, due to the scheduling policy and queue length. But the probability that there is a reset during an interval is minimized if initially ALL of the links are transmitting at the beginning of that interval. In that case, there is still a positive probability p that a reset occurs since each transmission ends in an exponential amount of time independent of others, and the scheduling interval also ends in an exponential amount of time independent of the transmission times. Thus, for each scheduling interval, no matter what happened in the earlier scheduling intervals, the conditional probability of a reset in that interval is at least p. Thus, $P(\overline{E}_2) \leq p_2 = (1-p)^{N_0}$, which is of the form $e^{-\alpha_2 C}$ for some $\alpha_2 > 0$.

Let $\alpha = \min(\alpha_1, \alpha_1)$. Using the union bound, we have that $P(\overline{E_1} \cup \overline{E_2}) \leq p_1 + p_2$. Thus, the probability that there is no reset time in interval $[t, t+C]$ is less than or equal to $2e^{-\alpha C}$. ∎

Now we will establish the existence of fluid limits.

Proposition 2: Consider a sequence of systems, indexed by n and $n \to \infty$, with the initial queue length for the n^{th} system, $Q^n(0)$, such that $\sum_{i \in K} Q_i^n(0) \leq n$. The arrival process is the same for all the systems, and the queue evolves according to the asynchronous LQF scheduling policy described in section II-A. Then, a limit $(\overline{Q}(t), \overline{D}(t), \overline{T}(t), \overline{Y}(u))$ of $\left(\frac{Q^n(nt)}{n}, \frac{D^n(nt)}{n}, \frac{T^n(nt)}{n}, \frac{Y^n(nu)}{n} \right)$ exists almost surely as $n \to \infty$, in the topology of uniform convergence on compact

sets, along some subsequence. Moreover, there is a set Ω in the probability space with probability measure 1 over which $\overline{Q}(t)$ satisfies the κ-LQF constraint and

$$\overline{Q_i}(t) = \overline{Q_i}(0) + \lambda_i t - \overline{D_i}(t), \quad t \geq 0 \quad (6)$$

$$\sum_{m \in M[K]} \frac{d\overline{T}_m}{dt} = 1, \quad a.e.\, t \geq 0 \quad (7)$$

$$\overline{Y}(u) = \frac{\mu_i}{1 + \kappa\mu_i} u, \quad u \geq 0 \quad (8)$$

We skip the proof here because of space constraints. The Proof uses the Arzela-Ascoli theorem and a functional law of large numbers to show the existence of a limit $(\overline{Q}(t), \overline{D}(t), \overline{T}(t), \overline{Y}(u))$. We then use Lemma 2 and the Borel-Cantelli lemma to prove that $\overline{Q}(t)$ satisfies the κ-LQF constraint with probability 1. The Complete proof can be found in [12].

Note that the fluid limit is in general a random process and need not be a deterministic function. Also note that the fluid limit need not be unique. One can in general obtain different fluid limits by choosing a different subsequence of $\{1, 2, 3, ...\}$. However, it should be noted that the above theorem states that every fluid limit satisfies the κ-LQF constraint.

B. Stability of the Fluid Limit and Positive Recurrence of the Markov Chain

For any κ-feasible (λ, μ), we will show that the continuous-time Markov chain, $X = (Q, S)$, is positive recurrent by showing that the fluid limits reach zero in a finite time.

The following lemma is similar to the one proved for the case of discrete time in [7]. We skip the proof here for lack of space and it can found in [12].

Lemma 3: For any κ-feasible (λ, μ), if local pooling holds, then any (deterministic) trajectory, $(\overline{Q}(t) : t \geq 0)$ satisfying the κ-LQF, $\overline{Q}(t) = 0$ holds for any $t \geq \tau$, where $\tau = \max_{i \in K} \overline{Q}_i(0) \frac{\max_{i \in K}(\kappa + 1/\mu_i)}{\epsilon_*}$.

From Proposition 2 and Lemma 3, we have that the limit $\overline{Q}(t) : t \geq 0$ is such that $\overline{Q}(t) = 0$ for all $t \geq \tau$. This will then imply positive recurrence of the original system. This can be proved using the standard techniques in literature [11]. Thus, in summary we have the following theorem.

Theorem 1: If (λ, μ) is κ-feasible, then the system is positive recurrent. $\diamond$

For any $(\lambda, \mu) \in \mathcal{C}$, there is a $\kappa > 0$ such that (λ, μ) is κ-feasible. Thus, by choosing κ small enough, any rate pair (λ, μ) in the capacity region is stabilizable with asynchronous LQF.

IV. Discussion

In this section we will present simulation results to understand if the scheduling clock is important. We will then discuss whether a distributed implementation of asynchronous LQF is feasible.

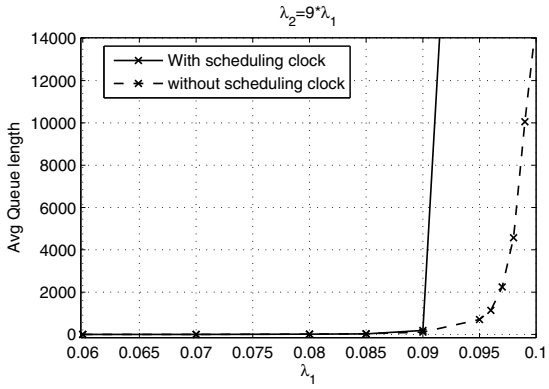

Fig. 3. Comparison of performance with and without wait period

A. Importance of Scheduling Clock

It is not clear if the small wait period which we introduced for the purpose of proving stability is really required. The fluid limit approach that we have used will probably not work in the case without wait period. As an example, consider the star interference graph, and a sequence of initial conditions, such that $\lim_{n \to \infty} \frac{Q_i^n(0)}{n} = 1$ for all i. If the outer links are scheduled at the beginning, no two outer links finish transmitting a packet at the same time. So the middle link gets a chance to transmit only after all but one of the outer queues are empty. The queue length of the middle link increases in fluid limit till that time. Therefore, we do not have a negative drift of maximum queue length in fluid limit.

Consider the network with star interference graph with six outer links. Figure 3 shows simulations on this network. The arrival rate to the outer links, λ_2, is chosen to be 9 times that to the central link, λ_1; i.e., $\lambda_2 = 9 * \lambda_1$. The average service rate of each packet is 1 unit and the average rate of the scheduling clock is 10; i.e., the average duration between scheduling times is 0.1 units. Rate $\lambda_1 = 0.1$ is the boundary of the capacity region, and $\lambda_1 = .09$ is the boundary of the 0.1-feasible region (i.e., feasible region with wait period $1/10$). Figure 3 suggests that the system would be stable even without the wait period. The figure has been plotted by uniformizing the system and averaging the observed queue lengths at 100 million events, where each event can be either an arrival, departure or scheduling clock tick.

Figure 4 shows a plot of average queue length as the wait period changes when $\lambda_2 = 9 * \lambda_1$ and $\lambda_1 = .097$. The queue length plotted here is averaged over observations at one billion events. We see no increase in the average queue length when kappa decreases to zero, which suggests that the algorithm is stable even without a wait period after completion of transmissions. The average queue length increases for higher κ because, for fixed arrival and service rates, a longer wait period is closer to the boundary of the corresponding κ-feasible region.

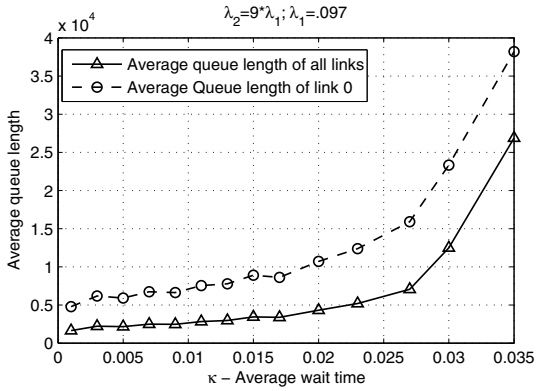

Fig. 4. Average queue length with the wait period

B. Distributed Implementation of Asynchronous LQF

The assumption of having a centralized clock can easily be removed. A common seed can be used at all the nodes for generating pseudo-random numbers that are used to simulate the scheduling clock. Then, a distributed decision can be made at the scheduling times, similar to the synchronous LQF case as follows.

Some time period after every scheduling clock tick is designated as a control time slot. This time slot is further divided into a finite number of subslots. A node is allowed to contend for transmission at a scheduling time t_0 only if none of its neighbors are transmitting at t_0. Each contending node at time t_0 chooses a number t, uniformly at random between k/Q_i and $k/(Q_i+1)$, for some fixed number k, where Q_i is its queue length. It then chooses the subslot into which $t+t_0$ falls. A node announces its intent to transmit in its chosen subslot only if none of its neighbors have announced before it. All the nodes that have successfully announced start transmission of data packets at the end of the control slot. Ties are broken at random.

If the division into subslots is infinitesimally fine, this will be an exact implementation of the LQF algorithm since a node with longer queue length has a higher priority. However even when we have a finite number of subslots, extensive simulations in [6] show that the above implementation can approximate LQF well.

V. Conclusions and Future directions

In this paper, we have studied an asynchronous version of the LQF algorithm for an ad hoc wireless network when packet lengths are variable. This algorithm is throughput-optimal if there is a small wait period between departures and schedules. It is not clear if this wait period is essential for throughput-optimality. Proving or disproving throughput-optimality without wait period is one open question.

While the distributed algorithm with infinitesimally fine subslots is throughput-optimal (under the local pooling condition), the required precision in the clocks and response speed of the nodes are both unbounded. With finite clock precision, we might not have a proof of throughput-optimality. So an open problem is to provide a provably optimal, distributed algorithm for which both the precision in the clocks and the response speed in the nodes are bounded independently of the number of nodes and number of packets in the network.

Throughput-optimality is a first order criterion. Delay performance of an algorithm is more important in practice. Understanding the delay performance of asynchronous LQF is an interesting topic for further research.

One extension of the results in this paper is to allow general arrival and departure processes with appropriate assumptions on the tails. We believe that this can be done using standard techniques from the literature as in [11].

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automatic Control,* vol. 37, pp. 1936-1948, Dec 1992.

[2] R. R. Boorstyn, A. Kershenbaum, B. Maglaris, and V. Sahin, "Throughput analysis in multihop CSMA packet radio networks," *IEEE Transactions on Communications*, vol. 35, no. 3, pp. 267–274, March 1987.

[3] S. C. Liew, C. Kai, J. Leung, and B. Wong, "Back-of-the-envelope computation of throughput distributions in CSMA wireless networks," *IEEE Trans. on Mobile Computing* vol. 9, no. 9, pp 1319-1331, Sept 2010.

[4] J. Shin and D. Shah. (2010, Apr). Randomized Scheduling Algorithm for Queueing Networks. [Online]. Available: http://arxiv.org/pdf/0908.3670

[5] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," presented at Allerton Conference on Communication, Control, and Computing, Monticello, Illinois, 2008.

[6] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," in *Proceedings of IEEE INFOCOM*, March 2010.

[7] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: Second-order properties using fluid limits," *Adv. in Appl. Probab.,* vol. 38, no. 2, pp. 505-521, 2006.

[8] A. Brzezinski, G. Zussman, and E. Modiano, "Enabling distributed throughput maximization in wireless mesh networks - A partitioning approach," in *Proceedings of ACM MOBICOM'06*, September 2006.

[9] 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'08*, April 2008.

[10] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks," in *Proceedings of the 10th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2009.

[11] J. G. Dai, "On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models," *Annals of Applied Probability*, vol. 5, pp. 49-77, 1995.

[12] S. Maguluri, B. Hajek and R. Srikant, "The Stability of Longest-Queue-First Scheduling With Variable Packet Sizes," Technical Report, http://hdl.handle.net/2142/26514