This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Optimal Control of Wireless Networks with Finite Buffers

Long Bao Le[†], Eytan Modiano[†], and Ness B. Shroff[♯]
[†]Massachusetts Institute of Technology
[♯]The Ohio State University
Emails: {longble, modiano}@mit.edu, and shroff@ece.osu.edu

***Abstract*—This paper considers network control for wireless networks with finite buffers. We investigate the performance of joint flow control, routing, and scheduling algorithms which achieve high network utility and deterministically bounded backlogs inside the network. Our algorithms guarantee that buffers inside the network never overflow. We study the tradeoff between buffer size and network utility and show that if internal buffers have size $(N-1)/\epsilon$ then a high fraction of the maximum utility can be achieved, where ϵ captures the loss in utility and N is the number of network nodes. The underlying scheduling/routing component of the considered control algorithms requires ingress queue length information (IQI) at all network nodes. However, we show that these algorithms can achieve the same utility performance with delayed ingress queue length information. Numerical results reveal that the considered algorithms achieve nearly optimal network utility with a significant reduction in queue backlog compared to the existing algorithm in the literature. Finally, we discuss extension of the algorithms to wireless networks with time-varying links.**

***Index Terms*—Network control, wireless scheduling, flow control, routing, delay control, throughput region, finite buffer, utility maximization**

I. Introduction

Design of low delay wireless networks that optimally utilize the network capacity is one of the most important problems in network theory and engineering. Since the seminal paper of Tassiulas and Ephremides which proposed a joint routing and scheduling algorithm that achieves the maximum network throughput [1], significant efforts have been invested in developing more efficient network control algorithms [2]-[22]. Most existing works, however, focus on achieving a guaranteed fraction of the maximum throughput region with low communication and computation complexity.

In addition, most papers on network control assume that all buffers in the network are infinite so buffer overflows never occur. In practice, network buffers are finite. Therefore, sizing buffers such that buffer overflow inside the network can be alleviated or completely avoided is clearly an important engineering problem. Moreover, buffer sizing should be performed in such a way that the network capacity is not wasted. In fact, it has been shown in [23] that networks with finite buffers may suffer from significant throughput loss if they are not designed appropriately. The problem of buffer sizing is closely related to the delay control problem because finite buffers imply deterministically bounded backlogs which by Little's formula lead to bounded average delay.

There have been some recent papers that analyze delay performance of cross-layer scheduling algorithms [24]-[28]. In particular, it has been shown that the well-known maximum weight scheduling algorithm achieves order-optimal delay in the uplink-downlink of cellular networks [24] and in most practical large-scale multihop wireless networks [25]. Other works on delay analysis for different scheduling algorithms in wireless networks can be found in [27]-[28]. In [4], [26], [29], and [30], it has been shown that by combining the principle of shortest-path routing and differential backlog routing, end-to-end delay performance can be improved. In [15], [31], the virtual queue technique has been used to improve network delay performance. These existing works, however, do not consider the problem of providing backlog or delay performance guarantees.

In this paper, we employ flow controllers to deterministically bound the queue backlogs inside the network. Specifically, we combine the Lyapunov optimization technique of [2], [3], [30] and the scheduling mechanism proposed in [6] to construct joint flow control, routing and scheduling algorithms for wireless networks with finite buffers. Note that in [2], [3], [30] the problem of network utility maximization is considered assuming that all buffers in the network are infinite. The authors of [6] propose scheduling algorithms for network with finite buffers assuming static routing and traffic arrival rates that are inside the throughput region. This paper considers the general setting where traffic arrival rates can be either inside or outside the throughput region, internal buffers in the network are finite, and dynamic routing is used to achieve the largest possible network throughput. Our contributions can be summarized as follows.

- We consider control algorithms that achieve high network utility and deterministically bounded backlogs for all buffers inside the network. Moreover, these algorithms ensure that internal buffers never overflow.
- We demonstrate a tradeoff between buffer sizes and achievable network utility.
- We show that delayed ingress queue information does not affect the utility of the control algorithms albeit at the cost of added queue backlogs.

This work was supported by NSERC Postdoctoral Fellowship and by ARO Muri grant number W911NF-08-1-0238, NSF grant numbers CNS-0915988 and DTRA grant HDTRA1-07-1-0004.

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

- We show via simulation that the considered control algorithms perform very well in both the under and overloaded traffic regimes. Specifically, they achieve nearly optimal utility performance with very low and bounded backlogs.

The remainder of this paper is organized as follows. The system model is described in section II. In section III, we analyze the performance of the control algorithm in the heavy traffic regime. Section IV focuses on the performance analysis of the control algorithm for arbitrary traffic arrival rates. Some extensions are presented in section V. Numerical results are presented in section VI followed by conclusion in section VII.

II. System Model

We consider a wireless network which is modeled as a graph $G = (\Gamma, E)$ where Γ is the set of nodes and E is the set of links. Let N and L be the number of nodes and links in the network, respectively. We assume a time-slotted wireless system where packet arrivals and transmissions occur at the beginning of time slots of unit length. There are multiple network flows in the network each of which corresponds a particular source-destination pair.

Arrival traffic is stored in input reservoirs and flow controllers are employed at source nodes to inject data from input reservoirs into the network in each time slot. Let n_c be the source node and d_c be the destination node of flow c. We will call the buffer at the source node n_c of flow c an ingress buffer. All other buffers storing packets of flow c inside the network are called internal buffers. Let $R_{n_c}^{(c)}(t)$ be the amount of traffic of flow c injected from the input reservoir into the network at node n_c in time slot t. Note that a particular node can be a source node for several flows. Let C_n be the set of flows whose source node is n. Hence, for any flow $c \in C_n$ its source node n_c is node n. It is assumed that $\sum_{c \in C_n} R_n^{(c)} \leq R_n^{\mathsf{max}}$ where the parameter R_n^{max} can be used to control the burstiness of admitted traffic from node n into the network. Let $R_{\mathsf{max}} = \max_{\{n\}} \{R_n^{\mathsf{max}}\}$ which will be used in the analysis. Let C denote the total number of flows in the network.

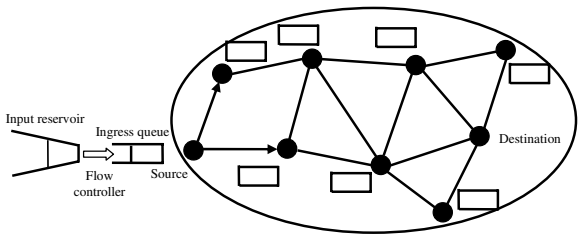

Fig. 1. Wireless network with finite internal buffers

Each internal node maintains multiple finite buffers (one per flow) while ingress buffers at all source nodes are assumed to be unlimited. This assumption is justified by the fact that in many wireless networks (e.g., wireless sensor networks) buffer space is limited. However, buffers in ingress routers or devices are relatively large. Moreover, since ingress buffers only need to store traffic from a small number of end-users, they can be made large enough to accommodate all incoming traffic with high probability. The model characterizes both the finiteness of the buffers within the network and uses unlimited buffers at the ingress to explicitly characterize this discrepancy.

Let l_c be the internal buffer size used to store packet of flow c at each network node. We denote the queue length of flow c at node n at the beginning of time slot t by $Q_n^{(c)}(t)$. Note that data packets of any flow are delivered to the higher layer upon reaching the destination node, so $Q_{d_c}^{(c)}(t) = 0$. Assume that the capacity of any link is one packet per time slot. In addition, let $\mu_{nm}^{(c)}(t)$ be the number of packets of flow c transmitted over link (n, m) in time slot t. We assume that $\mu_{nm}^{(c)}(t) = 1$ if we transmit a packet of flow c over link (n, m) and $\mu_{nm}^{(c)}(t) = 0$, otherwise. In the following, we will use $\mu_{nm}^{(c)}(t)$ or $\mu_l^{(c)}(t)$ to denote the number of packet transmitted over link (n, m) or link l, respectively (i.e., subscripts to denote links can be formed by the corresponding transmitting and receiving nodes or just a single letter denoting the link's index). Let Ω_n^{in} and Ω_n^{out} be the set of incoming and outgoing links at node n. The network model is illustrated in Fig. 1. For notational convenience, when there is no ambiguity, we omit the time index t in related variables.

We assume that traffic of any flow is not routed back to its source node. Therefore, we have $\mu_{mn_c}^{(c)} = 0, \forall m, c$. It is clear that this restriction does not impact the achievable throughput of the considered control algorithms. To guarantee that packets transmitted from a particular source node n_c can reach the corresponding destination node d_c, any links (n_c, m) will be blocked from transmitting data of flow c if there is no loop-free route from m to d_c. We assume that a node can communicate with (i.e., transmit or receive) at most one neighboring node at any time (i.e., node exclusive interference constraints).[1]

We further assume that node n will not transmit data of flow c along any link (n, m) whenever $Q_n^{(c)} < 1$ (i.e., a node will not transmit traffic of any flow if the corresponding queue does not have enough data to fill the link capacity). Under this assumption, the queue evolutions can be written as

$$Q_n^{(c)}(t+1) = Q_n^{(c)}(t) - \sum_{l \in \Omega_n^{\mathsf{out}}} \mu_l^{(c)}(t) + \sum_{l \in \Omega_n^{\mathsf{in}}} \mu_l^{(c)}(t) + R_n^{(c)}(t) \quad (1)$$

where $R_n^{(c)}(t) = 0$, $\forall t$ and $n \neq n_c$. Note that $\sum_{l \in \Omega_n^{\mathsf{out}}} \mu_l^{(c)}(t) = 1$ only if $Q_n^{(c)}(t) \geq 1$ and one of the outgoing links of node n is activated for flow c. Let $\overline{r}_n^{(c)}(t)$ be the time average rate of admitted traffic for flow c at the corresponding source node n_c up to time t, that is

$$\overline{r}_{n_c}^{(c)}(t) \triangleq \frac{1}{t} \sum_{\tau=0}^{t-1} E\left\{R_{n_c}^{(c)}(\tau)\right\}. \quad (2)$$

[1]This assumption is made for simplicity of the derivations and can be easily relaxed at the expense of larger backlog bounds. Moreover, the analysis in this paper can be extended to other interference models (e.g., k-hop interference model) easily.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

The long-term time-average admitted rate for flow c is defined as

$$\overline{r}_{n_c}^{(c)} \triangleq \lim_{t\to\infty} \overline{r}_{n_c}^{(c)}(t). \tag{3}$$

Now, we recall the definitions of network stability and the maximum throughput region (or throughput region for brevity) [1] which will be used in our analysis. A queue for a particular flow c at node n is called strongly stable if

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} E\left\{Q_n^{(c)}(\tau)\right\} < \infty. \tag{4}$$

In addition, the network is called strongly stable (or stable for simplicity) if all individual queues in the network are stable. The maximum throughput region Λ of a wireless network with unlimited ingress and internal buffers is the set of all traffic arrival rate vectors such that there exists a network control algorithm to stabilize all individual queues in the network. Note that when internal buffers are finite, stability corresponds to maintaining bounded backlogs in ingress buffers (since internal buffers are finite by design, the issue of stability does not arise there). In this context, it is necessary to devise a control algorithm that can achieve high throughput (utility) with finite buffers. To quantify the performance of the control algorithms, we need some more definitions. First, let us define the ϵ-stripped throughput region as follows:

$$\Lambda_\epsilon \triangleq \left\{\left(r_{n_c}^{(c)}\right) \mid \left(r_{n_c}^{(c)}+\epsilon\right) \in \Lambda\right\}. \tag{5}$$

where $\left(r_n^{(c)}\right) \triangleq \left(r_{n_1}^{(1)}, r_{n_2}^{(2)}, \ldots, r_{n_C}^{(C)}\right)$. Also, let $(r_{n_c}^{*(c)}(\epsilon))$ be the optimal solution of the following optimization problem

$$\text{maximize} \quad \sum_c g\left(r_{n_c}^{(c)}\right) \tag{6}$$

$$\text{subject to} \quad \left(r_{n_c}^{(c)}\right) \in \Lambda_\epsilon \tag{7}$$

$$r_{n_c}^{(c)} \le \lambda_{n_c}^{(c)} \tag{8}$$

where $\left(\lambda_{n_c}^{(c)}\right) = \left(\lambda_{n_1}^{(1)}, \lambda_{n_2}^{(2)}, \cdots, \lambda_{n_C}^{(C)}\right)^T$ is the average traffic arrival rate vector, $(.)^T$ denotes the vector transposition, and $g^{(c)}(.)$ are increasing and concave utility functions. Here, utility functions express the level of satisfaction of users with respect to admitted rates. We will quantify the performance of the considered control algorithms in terms of $(r_{n_c}^{*(c)}(\epsilon))$. Note that $(r_{n_c}^{*(c)}(\epsilon))$ tends to the optimal solution $(r_{n_c}^{*(c)})$ as $\epsilon \to 0$ where $(r_{n_c}^{*(c)})$ is the optimal solution of the optimization problem (6)-(8) where Λ_ϵ is replaced by Λ (i.e., the original throughput region). Also, for constantly backlogged sources the constraints on traffic arrival rates (8) are not needed.

III. Network Optimization in the Heavy Traffic Regime

We start by considering the case where all sources are constantly backlogged. We seek a balance between optimizing the total network utility and bounding total queue backlog inside the network. Specifically, we want to solve the following optimization problem

$$\text{maximize} \quad \sum_c g^{(c)}\left(\overline{r}_{n_c}^{(c)}\right) \tag{9}$$

$$\text{subject to} \quad \left(\overline{r}_{n_c}^{(c)}\right) \in \Lambda \tag{10}$$

$$Q_n^{(c)} \le l_c, \ \forall c, \text{ and } n \ne n_c \tag{11}$$

where $\overline{r}_{n_c}^{(c)}$ is the time average admitted rate for flow c at node n_c, $\left(\overline{r}_{n_c}^{(c)}\right) = \left(\overline{r}_{n_1}^{(1)}, \overline{r}_{n_2}^{(2)}, \cdots, \overline{r}_{n_C}^{(C)}\right)^T$ is the time average admitted rate vector, and l_c is the buffer size. Constraint (11) ensures that the backlogs in internal buffers are finite and bounded by l_c at all times.

In [3], [30], the corresponding problem without the buffer limit is considered. As a result, queue backlogs inside the network at internal buffers can be very large (although with bounded expectations). The large backlog accumulation inside the network is not desirable because it can lead to increased delays, buffer overflows, and throughput reduction due to retransmission of lost packets. Now, consider the following algorithm which solves the problem formulated in (9)-(11).

Algorithm 1: Constantly Backlogged Sources

1) Flow Control: Each node n injects an amount of traffic into the network which is equal to $R_{n_c}^{(c)}(t) = x_{n_c}^{(c)}$ where $x_{n_c}^{(c)}$ is the solution of the following optimization problem

$$\begin{array}{ll} \text{maximize} & \sum_c \left[V g^{(c)}\left(x_{n_c}^{(c)}\right) - 2Q_{n_c}^{(c)} x_{n_c}^{(c)}\right] \\ \text{subject to} & 0 \le \sum_{c\in C_n} x_n^{(c)} \le R_n^{\max}, \ \forall n. \end{array} \tag{12}$$

where V is a controlled parameter.

2) Routing/Scheduling: Each link (n,m) calculates the differential backlog for flow c as follows:

$$dB_{n,m}^{(c)} \triangleq \begin{cases} \frac{Q_{n_c}^{(c)}}{l_c}\left[l_c - Q_m^{(c)}\right], & \text{if } n = n_c \\ \frac{Q_{n_c}^{(c)}}{l_c}\left[Q_n^{(c)} - Q_m^{(c)}\right], & \text{if } n, m \ne n_c \end{cases}. \tag{13}$$

Then, link (n,m) calculates the maximum differential backlog as follows:

$$W_{(n,m)}^* \triangleq \max_c \left\{dB_{n,m}^{(c)}\right\}. \tag{14}$$

Let $\vec{S}$ be the schedule where its l-th component $S_l = 1$ if link l is scheduled and $S_l = 0$ otherwise. The schedule $\vec{S}^*$ is chosen in each time slot t as follows:

$$\vec{S}^* = \underset{\vec{S}\in\Phi}{\operatorname{argmax}} \vec{S}^T \vec{W}^*. \tag{15}$$

where Φ is the set of all feasible schedules as determined by the underlying wireless interference model. We will not schedule any link l with $W_l^* \le 0$. For each scheduled link, one packet of the flow that achieves the maximum differential backlog in (13) is transmitted if the corresponding buffer has at least one packet waiting for transmission.

Before investigating the performance of this algorithm, we would like to make some comments. First, it can be observed

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

that the flow controller of this algorithm admits less traffic of flow c if the corresponding ingress buffer is more congested (i.e., large $Q_{n_c}^{(c)}$). Also, a larger value V results in higher achievable utility albeit at the cost of larger queue backlogs. Moreover, the routing component of this algorithm (13) is an adaptation of the differential backlog routing of Tassiulas and Ephremides [1] to the case of finite buffers [6]. The scheduling rule (15) is the well-known max-weight scheduling algorithm.

In fact, the flow controller of this algorithm is the same as that proposed in [30], [3] while the differential backlog in (13) is the same as that in [6]. However, [30], [3] assume all network buffers are infinite while [6] assumes that traffic arrival rates are inside the corresponding throughput region and static routing. In contrast, we consider dynamic routing to achieve the largest possible network throughput. Also, we consider both heavy traffic and arbitrary arrival rates and employ flow controllers to control the utility-backlog tradeoff.

The differential backlog in (13) ensures that internal buffers never overflow. In fact, the differential backlog of any source link (n_c, m), given by $Q_{n_c}^{(c)}/l_c \left[l_c - Q_m^{(c)}\right]$, bounds the backlogs of all internal buffers by l_c. The differential backlogs of all other links (n, m), given by $Q_{n_c}^{(c)}/l_c \left[Q_n^{(c)} - Q_m^{(c)}\right]$, is essentially the multiplication of the standard differential backlog $\left[Q_m^{(c)} - Q_m^{(c)}\right]$ from [1] and the normalized ingress queue backlog $Q_{n_c}^{(c)}/l_c$. Incorporating ingress queue backlog into the differential backlog in (13) prioritizes the flow whose ingress queue is more congested. This helps stabilize ingress queues because the scheduling component of algorithm 1 still has the "max-weight" structure as that proposed in [1].

Assume that for each flow c, each wireless node maintains a buffer space of at least l_c packets. Assume that all buffers in the network are empty initially. Then, the queue backlogs in all internal buffers for flow c are always smaller than or equal to l_c. That is

$$Q_n^{(c)}(t) \leq l_c, \ \forall c, t \text{ and } n \neq n_c. \tag{16}$$

To see this, note that all internal buffers in the network always have an integer number of packets. This is because the scheduler of algorithm 1 does not transmit a fraction of packet from any ingress buffer to the corresponding internal buffer. Because at most one packet can be transmitted from any queue in one time slot, buffer overflow can only occur at a particular internal buffer if the number of packets in that buffer is l_c and it receives one more packet from one of its neighboring nodes. We will show that this could not happen by considering the following cases.

- Consider any link (n_c, m) for a particular flow c. It can be observed that if $Q_m^{(c)} = l_c$ then the buffer at node m for flow c will never receive any more packets from node n_c. This is because the differential backlog of link (n_c, m) in this case is $dB_{n_c,m}^{(c)} = Q_{n_c}^{(c)}/l_c \left[l_c - Q_m^{(c)}\right] = 0$. Therefore, link (n_c, m) is not scheduled for flow c.
- Consider any link (n, m) for $n \neq n_c$. Suppose the first buffer overflow event in the network occurs at node m for flow c. Right before the time slot where this first buffer overflow occured, we must have $Q_m^{(c)} = l_c$. However, consider the differential backlog of any link (n, m) in this previous time slot, we have $dB_{n,m}^{(c)} = Q_{n_c}^{(c)}/l_c \left[Q_n^{(c)} - Q_m^{(c)}\right] = Q_{n_c}^{(c)}/l_c \left[Q_n^{(c)} - l_c\right] \leq 0$. This means any such link (n, m) will not be scheduled for flow c. Therefore, overflow could not occur at node m which contradicts our assumption.

Therefore, we have deterministically bounded backlogs inside the network at all times. Now, we are ready to state one of the main results of this paper.

Theorem 1: Given $\epsilon > 0$, if the internal buffer size satisfies

$$l_c \geq \frac{N-1}{2\epsilon}. \tag{17}$$

Then, we have the following bounds for utility and ingress queue backlog

$$\liminf_{t\to\infty} \sum_c g^{(c)}\left(\overline{r}_{n_c}^{(c)}(t)\right) \geq \sum_c g^{(c)}\left(r_{n_c}^{*(c)}(\epsilon)\right) - \frac{D_1}{V} \tag{18}$$

$$\overline{\sum_c Q_{n_c}^{(c)}} \leq \frac{D_1 + V G_{\mathsf{max}}}{2\lambda_{\mathsf{max}} - \frac{(N-1)}{l_c}} \tag{19}$$

where $D_1 \triangleq C(R_{\mathsf{max}}^2+2)+C(N-1)R_{\mathsf{max}}l_c$ is a finite number, C is the number of flows in the network, R_{max} is the maximum amount of traffic injected into the network from any node, V is a design parameter which controls the utility and backlog bound tradeoff, λ_{max} is the largest number such that $\vec{\lambda}_{\mathsf{max}} \in \Lambda$ where $\vec{\lambda} = (\lambda, \lambda, \cdots, \lambda)^T$ is a column vector with all elements equal to λ,

$$G_{\mathsf{max}} \triangleq \max_{\left\{\sum_{c\in C_n} r_n^{(c)} \leq R_n^{\mathsf{max}}\right\}} \left\{\sum_c g^{(c)}\left(r_{n_c}^{(c)}\right)\right\}, \tag{20}$$

$$\overline{\sum_c Q_{n_c}^{(c)}} \triangleq \limsup_{M\to\infty} \frac{1}{M} \sum_{\tau=0}^{M-1} \sum_c E\left\{Q_{n_c}^{(c)}(\tau)\right\}. \tag{21}$$

Proof: A sketch of the proof is given in Appendix A. ■

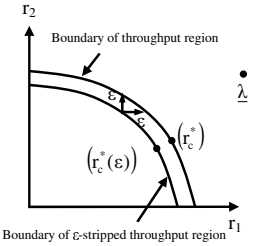

Fig. 2. Geometric illustration of optimal rate and lower-bound of achievable rate in the heavy load regime

Here, we would like to make some comments. First, as $V \to \infty$, the total utility achieved by the time-average admitted rate $\left(\overline{r}_{n_c}^{(c)}\right)$ is lower-bounded by that due to $\left(r_{n_c}^{*(c)}(\epsilon)\right)$ from (6)-(7). This lower-bound is parameterized by the design parameter ϵ which in turn determines the queue backlog bound

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

in all internal buffers as given by (17). Specifically, suppose we choose $l_c = (N-1)/(2\epsilon)$ as suggested by (17) . Then, when ϵ is larger, the buffer size l_c is smaller but the utility lower-bound achieved by $\left(r_{n_c}^{*(c)}(\epsilon)\right)$ is also smaller as given in (18). This achievable admitted rate vector $\left(r_{n_c}^{*(c)}(\epsilon)\right)$ is illustrated in Fig. 2. Second, the inequality (19) gives the upper-bound for the total ingress queue backlog which is controlled by another design parameter V. In particular, when V increases, the utility lower bound in (18) also increases but the upper-bound on the total ingress queue backlog becomes larger.

Remark 1: Without flow controllers, [6] has shown how to choose the internal buffer size to stabilize the ingress buffers. However, this chosen buffer size depends on the relative distance between the traffic arrival rate vector and the boundary of the throughput region. In practice, it is not easy to determine this distance parameter. Moreover, ingress queues will become unstable when the traffic arrival rate is outside the throughput region. In contrast, by appropriately designing a flow controller, our cross-layer algorithm can always stabilize the ingress queues. In addition, the internal buffer size in our design depends on a tunable design parameter ϵ which only impacts the utility lower bound and does not depend on the traffic itself.

Remark 2: Given ϵ, if we choose the smallest possible buffer size suggested by proposition 1 (i.e., $l_c = (N-1)/(2\epsilon)$), the backlog bound becomes

$$\frac{1}{M}\sum_{\tau=0}^{M-1}\sum_{c} E\left\{Q_{n_c}^{(c)}(\tau)\right\} \leq \frac{D_1 + VG_{\mathsf{max}}}{2\lambda_{\mathsf{max}} - \epsilon}. \tag{22}$$

Taking $\epsilon = 0$, we have a backlog bound of same form as that derived in [3].

IV. Network Optimization with Arbitrary Arrival Rates

Now, we consider the more general case where traffic arrival rates $\left(\lambda_{n_c}^{(c)}\right)$ can be inside or outside the maximum throughput region. In this case, the long-term time average admitted rates should be constrained to be smaller than the average traffic arrival rates. For this case, our objective is to solve the following stochastic optimization problem

$$\text{maximize} \quad \sum_{c} g^{(c)}\left(\overline{r}_{n_c}^{(c)}\right) \tag{23}$$

$$\text{subject to} \quad \left(\overline{r}_{n_c}^{(c)}\right) \in \Lambda \tag{24}$$

$$0 \leq \overline{r}_{n_c}^{(c)} \leq \lambda_{n_c}^{(c)} \tag{25}$$

$$Q_n^{(c)} \leq l_c, \ \forall c, \text{ and } n \neq n_c. \tag{26}$$

In this network setting, because the traffic arrival rates may be inside the throughput region, we rely on newly-introduced variables $Y_n^{(c)}(t)$ which capture "virtual queues backlogs" to perform flow control. These virtual queues track the difference between the instantaneous admitted traffic rates $R_n^{(c)}(t)$ and the "potential" admitted rates $z_{n_c}^{(c)}(t)$. In particular, we consider the following control algorithm which achieves our objective.

Algorithm 2: Sources with Arbitrary Arrival Rates

1) Flow Control: Each node n injects an amount of traffic of flow c into the network $R_{n_c}^{(c)}(t)$ which is the solution of the following optimization problem

$$\begin{array}{ll} \text{maximize} & \sum_{c\in C_{n_c}}\left[Y_{n_c}^{(c)}(t) - Q_{n_c}^{(c)}(t)\right] R_{n_c}^{(c)}(t) \\ \text{subject to} & \sum_{c\in C_{n_c}} R_{n_c}^{(c)}(t) \leq R_{n_c}^{\mathsf{max}} \\ & R_{n_c}^{(c)}(t) \leq A_{n_c}^{(c)}(t) + L_{n_c}^{(c)}(t) \end{array} \tag{27}$$

where $L_{n_c}^{(c)}(t)$ is the backlog of the input reservoir, $A_{n_c}^{(c)}(t)$ is the number of arriving packets in time slot t, and $Y_{n_c}^{(c)}(t)$ represents "backlog" in the virtual queue which has the following queue-like evolution

$$Y_{n_c}^{(c)}(t+1) = \max\left\{Y_{n_c}^{(c)}(t) - R_{n_c}^{(c)}(t), 0\right\} + z_{n_c}^{(c)}(t) \tag{28}$$

where $z_{n_c}^{(c)}(t)$ are auxiliary variables which are calculated from the following optimization problem

$$\begin{array}{ll} \text{maximize} & \sum_{c}\left\{Vg\left(z_{n_c}^{(c)}\right) - 2Y_{n_c}^{(c)} z_{n_c}^{(c)}\right\} \\ \text{subject to} & \sum_{c\in C_n} z_n^{(c)} \leq R_n^{\mathsf{max}}. \end{array} \tag{29}$$

We then update the virtual queue variables $Y_{n_c}^{(c)}$ according to (28) every time slot.

2) Scheduling and routing are performed as in algorithm 1.

The flow controller in this algorithm is the same as that in [2]. Its operation can be interpreted intuitively as follows. The auxiliary variables $z_{n_c}^{(c)}(t)$ plays the role of $R_{n_c}^{(c)}(t)$ in algorithm 1 for the heavy traffic regime. In fact, the optimization problem (29) from which we calculate $z_{n_c}^{(c)}(t)$ is similar to the one in (12) where $Q_{n_c}^{(c)}$ is replaced by $Y_{n_c}^{(c)}$. Hence, $z_{n_c}^{(c)}$ represents the potential rate that would have been admitted if sources were backlogged. The virtual queue $Y_{n_c}^{(c)}(t)$ captures the difference between the potential admitted rate $z_{n_c}^{(c)}(t)$ and the actual admitted rate $R_{n_c}^{(c)}(t)$ based on which the flow controller determines the amount of admitted traffic $R_{n_c}^{(c)}(t)$ from (27). Here, the "virtual differential backlogs" $\left[Y_{n_c}^{(c)} - Q_{n_c}^{(c)}\right]$ determines from which flow to inject data into the network.

In addition, it can be observed that (28) captures the dynamics of a virtual queue with service process $R_{n_c}^{(c)}(t)$ and arrival process $z_{n_c}^{(c)}(t)$. Hence, if these virtual queues are stable, then the time-average rates of $R_{n_c}^{(c)}(t)$ are equal to the time-average of $z_{n_c}^{(c)}(t)$. That means we must have $\overline{r}_{n_c}^{(c)} = \overline{z}_{n_c}^{(c)}$ where $\overline{z}_{n_c}^{(c)}$ denotes the time average of $z_{n_c}^{(c)}(t)$.

It can be verified that the amount of admitted traffic $R_{n_c}^{(c)}(t)$ calculated from (27) only takes integer values assuming that $R_{n_c}^{\mathsf{max}}$ are integer. Specifically, the solution of (27) can be calculated as follows. First, we sort the quantities $\left[Y_n^{(c)} - Q_n^{(c)}\right]$ for all flows c whose source nodes are node n. Then, starting from the flow with the largest positive value in the sorted list, we admit the largest possible amount of traffic considering the constraints in (27). Because $A_{n_c}^{(c)}$, $L_{n_c}^{(c)}$, and $R_{n_c}^{\mathsf{max}}$ are all integer, $R_{n_c}^{(c)}(t)$ will take integer values. Because $R_{n_c}^{(c)}(t)$ are integer, each ingress buffer is either empty or contains

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

an integer number of packets. We state the performance of algorithm 2 in the following theorem.

Theorem 2: Given $\epsilon > 0$, if the internal buffer size satisfies

$$l_c \geq \frac{N-1}{2\epsilon}. \tag{30}$$

Then, we have the following bounds for utility and ingress queue backlog

$$\liminf_{M\to\infty} \sum_c E\left\{g^{(c)}\left(\overline{r}_{n_c}^{(c)}(M)\right)\right\} \geq \sum_c g^{(c)}\left(r_{n_c}^{*(c)}(\epsilon)\right) - \frac{D_2}{V} \tag{31}$$

$$\overline{\sum_c Q_{n_c}^{(c)}} \leq \frac{D_2 + VG_{\max}}{2\lambda_{\max} - \frac{(N-1)}{l_c}}. \tag{32}$$

where $D_2 \triangleq 3CR_{\max}^2 + C(N-1)R_{\max}l_c$ is a finite number.

Proof: The proof is omitted due to space limitation but can be found in a technical report online [32]. ■

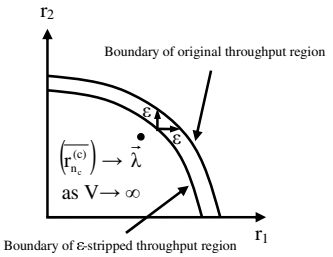

Fig. 3. Geometric illustration of optimal rate and lower-bound of achievable rate in the low load regime

Note that when $\left(\lambda_{n_c}^{(c)}\right) \in \Lambda_\epsilon$, we have $\left(r_{n_c}^{*(c)}(\epsilon)\right) = \left(\lambda_{n_c}^{(c)}\right)$ where $\left(\lambda_{n_c}^{(c)}\right)$ is the average traffic arrival rate vector. Therefore, we have $\left(\overline{r}_{n_c}^{(c)}(\epsilon)\right) \to \left(\lambda_{n_c}^{(c)}\right)$ as $V \to \infty$ in this case. That means we can achieve the optimal utility in this case by letting $V \to \infty$. This observation is illustrated in Fig. 3. Otherwise, if $\left(\lambda_{n_c}^{(c)}\right) \neq \Lambda_\epsilon$ then we return to the heavy traffic case considered in the previous section. In this case, the lower-bound on the achieved utility depends on the chosen parameter ϵ.

V. Further Extensions

In this section, we show how to extend the results in the previous sections in two important directions, namely the impact of delayed IQI and extensions to wireless networks with ON/OFF links.

A. Delayed Ingress Queue Length Information

Assume that the same scheduling/routing algorithm as in algorithm 1 is employed. However, only delayed IQI is available at all network nodes. Let T be the time delay of ingress queue length $Q_{n_c}^{(c)}(t)$ at all other network nodes (i.e, only $Q_{n_c}^{(c)}(t-T)$ is available at other nodes in time slot t). In the following, we investigate the performance of algorithm 1 when delayed IQI is used for scheduling. In particular, each link (n,m) and flow c calculate the differential backlog in slot t as follows:

$$dB_{n,m}^{(c)}(t) \triangleq \begin{cases} \frac{Q_{n_c}^{(c)}(t)}{l_c}\left[l_c - Q_m^{(c)}(t)\right], & \text{if } n = n_c \\ \frac{Q_{n_c}^{(c)}(t-T)}{l_c}\left[Q_n^{(c)}(t) - Q_m^{(c)}(t)\right], & \text{if } n, m \neq n_c \end{cases}.$$

This differential backlog is used for routing/scheduling while the same congestion controller as in algorithm 1 is employed. Note that the flow controller for flow c at node n_c only requires its local ingress queue length information $Q_{n_c}^{(c)}(t)$. Hence, this queue length information is available without delay. Now, we state the following result for this cross-layer algorithm with delayed IQI.

Theorem 3: If the buffer size satisfies

$$l_c \geq \frac{N-1}{2\epsilon}. \tag{33}$$

Then, we have the following bounds for utility and ingress queue backlog

$$\liminf_{t\to\infty} \sum_c g^{(c)}\left(\overline{r}_{n_c}^{(c)}(t)\right) \geq \sum_c g^{(c)}\left(r_{n_c}^{*(c)}(\epsilon)\right) - \frac{D_3}{V} \tag{34}$$

$$\overline{\sum_c Q_{n_c}^{(c)}} \leq \frac{D_3 + VG_{\max}}{2\lambda_{\max} - \frac{(N-1)}{l_c}}. \tag{35}$$

where V is predetermined number, $D_3 \triangleq D_1 + C(N+2R_{\max}+1)T$, and D_1 is given in theorem 1.

Proof: The proof is omitted due to space limitation but can be found in a technical report online [32]. ■

This theorem says that the same network utility lower-bound as that in algorithm 1 can be achieved as $V \to \infty$. However, the backlog upper-bound derived in (35) is larger than that derived in theorem 1 as D_3 depends on T.

It has been shown in [6] that a slightly larger buffer size is required when scheduling is performed based on queue length vector (for both ingress and internal buffers) with estimation errors. In this theorem, we show that with flow control when only IQI is delayed while internal queue information is available without delay, we do not need a larger buffer size. In addition, there are some other existing works along this line in the literature [19], [20] where it has been shown that infrequent or delayed channel estimation may reduce the maximum throughput region. However, delayed queue backlog information does not decrease utility.

Remark 3: Although we have assumed that the IQI delay value is the same for all nodes, the same utility performance guarantee can be achieved even if routing and scheduling is performed based on IQI with different delay values at different nodes.

B. Wireless Networks with ON/OFF Wireless Links

Our algorithms can be extended to multihop wireless networks with ON/OFF links due to channel fading [21]. Specifically, we assume that during each time slot a link may be

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

either ON or OFF and transmissions can only take place along ON links. Now, let us define the channel state vector $\vec{X} = (X_l)$, $l \in E$ where X_l denotes the channel state of link l. Note that this channel model is a special case of that considered in [3], [4] in which the authors have characterized the maximum throughput region.

In fact, the same flow controller as in algorithms 1 and 2 can be employed in this time-varying channel model along with the differential backlog routing. However, link scheduling must take into account the channel state. Specifically, let $\Phi_0(\vec{X}(t))$ be the set of feasible schedules in time slot t where for each of these feasible schedules we only activate wireless links which are in the ON state. Then, the schedule $\vec{S}^*$ is chosen in each time slot t as follows:

$$\vec{S}^* = \underset{\vec{S} \in \Phi_0(\vec{X}(t))}{\text{argmax}} \ \vec{S}^T \vec{W}^*. \tag{36}$$

For each scheduled link, one packet of the flow which achieves the maximum differential backlog is transmitted if the corresponding buffer has at least one packet waiting for transmission. It can be shown that the same performance as presented in Theorems 1 and 2 can be achieved for this modified algorithm although the maximum throughput region in this case is smaller than that considered in the previous sections where links are always ON.

VI. Numerical Results

We study the performance of the control algorithms using computer simulation. We consider a simple network of six nodes as shown in Fig. 4. All links are assumed to be bi-directional and the maximum weight scheduling algorithm is invoked in each time slot assuming the node exclusive (i.e., one-hop) interference model. We simulate four traffic flows whose (source, destination) pairs are (1, 6), (3, 4), (6, 2), (5, 2). Packet arrival processes to all input reservoirs are assumed to Bernoulli. The utility function is chosen to be $g(r) = \ln(r)$, $R_n^{max} = 2$, $\forall n$, $V = 1000$ (it is observed that increasing V further does not improve the performance of the algorithm in this case). We run our simulation over 10^7 time slots.

We compare performance of algorithm 2 with algorithm CLC2b proposed in [2] which assumed infinite internal buffer size (i.e., $l_c = \infty$ for all flows). We show time average admitted rates for each of the flows, total admitted rate of all flows, and the total average ingress and internal queue length. Specifically, the total average ingress and internal queue length are calculated as $Q_{\text{ingress}} \stackrel{\triangle}{=} \sum_c \overline{Q}_{n_c}^{(c)}$ and $Q_{\text{internal}} \stackrel{\triangle}{=} \sum_c \sum_{n \neq n_c} \overline{Q}_n^{(c)}$ where $\overline{Q}_n^{(c)}$ is the average queue length for flow c at node n.

We present the performance of algorithm 2 for the over-loaded case in Table I for different values of internal buffer size. Recall that given the buffer size l_c, theorem 2 says that the lower bound of the network utility corresponds to a rate vector which lies inside the ϵ-stripped throughput region for $\epsilon = (N-1)/(2l_c)$ as $V \to \infty$. However, table I shows that this performance bound is quite conservative. In particular, while $l_c = 2$ results in a slightly smaller total admitted rate and non-negligible deviation from optimal rates, $l_c = 10$ achieves time-average admitted rates which are very close to the optimal ones. Note that with $l_c = 50$, the corresponding ϵ is $\epsilon = (N-1)/(2l_c) = 5/100 = 0.05$. However, the actual loss in utility is negligible.

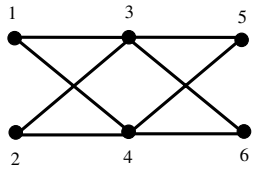

Fig. 4. Simulation network with primary interference model

Also, table I shows that our proposed algorithm achieves significantly smaller average queue lengths in both ingress and internal buffers compared to those due to algorithm CLC2b in [2]. For example, with $l_c = 10$, we can achieve admitted rates very close to the optimal values while the average ingress queue length is 6 times smaller, and the average total queue length in internal buffers is about 150 times smaller than those due to algorithm CLC2b in [2]. In addition, it can be observed that oversizing the internal buffers does not change the total average ingress queue length although it increases the total internal queue length.

We present numerical results for the underloaded traffic case in Table II. This table shows that algorithm CLC2b [2] achieves small average queue lengths and our proposed algorithm has comparative performance with $l_c = 10$. In addition, it is shown that under-sizing the internal buffers results in significant increase in ingress queue length. Intuitively, this is because with finite buffers throughput is slightly reduced leading to greater backlogs in ingress buffers. Also, over-sizing internal buffers improves queue length of ingress buffers but increases total queue length in internal buffers. Finally, we show the performance of our algorithm with delayed IQI for the overloaded traffic case, and $T = 20$ in Table III. This table shows that delayed IQI does not impact the throughput performance while it only results in marginal increase in the total average queue length of internal buffers.

VII. Conclusion

In this paper, we proposed and analyzed the performance of control algorithms for wireless networks with finite buffers. Two different algorithms have been investigated for both scenarios of heavy traffic and arbitrary traffic arrival rates. We also analyzed the performance of the control algorithms with delayed IQI. Numerical results confirm the superior performance of our proposed algorithms in terms of the backlog-utility tradeoff compared to existing algorithms in the literature. Our paper provides a unified framework for the problems of utility maximization, buffer sizing, and delay control in a very general setting. Developing decentralized control algorithms for wireless networks with finite buffers is an open problem which will be considered in our future work.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

TABLE I
OVERLOADED TRAFFIC FOR $\vec{\lambda} = (0.9, 0.9, 0.9, 0.9)$

		$\overline{r}_1^{(1)}$	$\overline{r}_3^{(2)}$	$\overline{r}_6^{(3)}$	$\overline{r}_5^{(4)}$	r_{tot}	Q_{ingress}	Q_{internal}
CLC2b [2]	$l_c = \infty$	0.2503	0.3004	0.3003	0.3001	1.1511	42×10^3	8.02×10^3
Proposed	$l_c = 2$	0.2788	0.2788	0.2789	0.2787	1.1152	7.18×10^3	16.3836
	$l_c = 10$	0.2501	0.3002	0.3003	0.3001	1.1507	7.00×10^3	53.2008
	$l_c = 50$	0.2501	0.3002	0.3004	0.3001	1.1507	7.00×10^3	237.1939
	$l_c = 100$	0.2502	0.3002	0.3004	0.3000	1.1507	7.00×10^3	466.3795

TABLE II
UNDERLOADED TRAFFIC FOR $\vec{\lambda} = (0.24, 0.29, 0.29, 0.29)$

		$\overline{r}_1^{(1)}$	$\overline{r}_3^{(2)}$	$\overline{r}_6^{(3)}$	$\overline{r}_5^{(4)}$	r_{tot}	Q_{ingress}	Q_{internal}
CLC2b [2]	$l_c = \infty$	0.2400	0.2899	0.2901	0.2897	1.1097	25.2774	43.2078
Proposed	$l_c = 2$	0.2400	0.2894	0.2896	0.2894	1.1084	6.72×10^3	16.3073
	$l_c = 10$	0.2399	0.2900	0.2900	0.2900	1.1099	26.6385	68.3762
	$l_c = 50$	0.2397	0.2899	0.2899	0.2900	1.1095	22.1105	296.5923
	$l_c = 100$	0.2401	0.2901	0.2902	0.2898	1.1102	21.6539	559.7127

TABLE III
OVERLOADED TRAFFIC WITH DELAYED IQI FOR $\vec{\lambda} = (0.9, 0.9, 0.9, 0.9)$, $T = 20$

		$\overline{r}_1^{(1)}$	$\overline{r}_3^{(2)}$	$\overline{r}_6^{(3)}$	$\overline{r}_5^{(4)}$	r_{tot}	Q_{ingress}	Q_{internal}
Proposed	$l_c = 2$	0.2736	0.2737	0.2738	0.2736	1.0947	7.31×10^3	16.0158
	$l_c = 10$	0.2502	0.3002	0.3003	0.3000	1.1507	7.00×10^3	56.9672
	$l_c = 50$	0.2501	0.3002	0.3003	0.3001	1.1507	7.00×10^3	240.2404
	$l_c = 100$	0.2502	0.3002	0.3003	0.3000	1.1507	7.00×10^3	468.9563

APPENDIX A
A SKETCH OF THE PROOF OF THEOREM 1

Consider the following Lyapunov function

$$L(\vec{Q}) \triangleq \sum_c \left(Q_{n_c}^{(c)}(t)\right)^2 + \sum_c \sum_{n:n\neq n_c} \frac{1}{l_c}\left(Q_n^{(c)}(t)\right)^2 Q_{n_c}^{(c)}(t). \quad (37)$$

Note that this is the same Lyapunov function as that proposed in [6]. Now, consider the Lyapunov drift

$$\Delta(t) \triangleq E\left\{L(\vec{Q})(t+1) - L(\vec{Q})(t)|\vec{Q}(t)\right\}. \quad (38)$$

Using the queue evolution equation in (1) and node exclusive interference constraints, we have

$$\Delta(t) - V\sum_c E\left\{g^{(c)}\left(R_{n_c}^{(c)}\right)|\vec{Q}(t)\right\} = D_4 + \sum_c \frac{(N-1)Q_{n_c}^{(c)}}{l_c} - \Psi_1(\vec{Q}) - \Psi_2(\vec{Q}). \quad (39)$$

where $D_4 = CR_{\mathsf{max}}^2 + CR_{\mathsf{max}}(N-1)l_c$ is a finite number and

$$\Psi_1(\vec{Q}) \triangleq 2\sum_c Q_{n_c}^{(c)} E\left\{\sum_{l\in\Omega_{n_c}^{\mathsf{out}}} \mu_l^{(c)}|\vec{Q}(t)\right\} + 2\sum_c \sum_{n:n\neq n_c} \frac{Q_n^{(c)}Q_{n_c}^{(c)}}{l_c} E\left\{\sum_{l\in\Omega_n^{\mathsf{out}}} \mu_l^{(c)} - \sum_{l\in\Omega_n^{\mathsf{in}}} \mu_l^{(c)}|\vec{Q}(t)\right\}, \quad (40)$$

$$\Psi_2(\vec{Q}) \triangleq \sum_c E\left\{Vg^{(c)}\left(R_{n_c}^{(c)}\right) - 2Q_{n_c}^{(c)}R_{n_c}^{(c)}|\vec{Q}\right\}. \quad (41)$$

Now, it can be verified that

$$\Psi_1(\vec{Q}) = 2\sum_c \sum_{(n_c,m)\in\Omega_{n_c}^{\mathsf{out}}} \frac{Q_{n_c}^{(c)}}{l_c} E\left\{\mu_{n_cm}^{(c)}|\vec{Q}\right\}\left[l_c - Q_m^{(c)}\right] + 2\sum_c \sum_{(n,m)\in E:n\neq n_c} \frac{Q_{n_c}^{(c)}}{l_c} E\left\{\mu_{(n,m)}^{(c)}|\vec{Q}\right\}\left[Q_n^{(c)} - Q_m^{(c)}\right]. \quad (42)$$

Hence, the routing/scheduling and flow control components of algorithm 1 maximize $\Psi_1(\vec{Q})$ and $\Psi_2(\vec{Q})$, respectively. Given $r_{n_c}^{*(c)}(\epsilon) \in \Lambda_\epsilon$ defined before, we have $\left(r_{n_c}^{*(c)}(\epsilon) + \epsilon\right) \in \Lambda$. Recall that the routing/scheduling and flow control components of algorithm 1 maximize $\Psi_1(\vec{Q})$ and $\Psi_2(\vec{Q})$, respectively. Hence, we have

$$\Psi_1(\vec{Q}) \geq 2\sum_c Q_{n_c}^{(c)}\left[r_{n_c}^{*(c)}(\epsilon) + \epsilon\right] - 2C \quad (43)$$

$$\Psi_2(\vec{Q}) \geq \sum_c \left\{Vg\left(r_{n_c}^{*(c)}(\epsilon)\right) - 2Q_{n_c}^{(c)}r_{n_c}^{*(c)}(\epsilon)\right\} \quad (44)$$

where the constant $2C$ in (43) accounts for the fact that we do not transmit any packets of flow c on any link (n_c, m) if $Q_{n_c}^{(c)}(t) < 1$. Using the results of (43) and (44) in (39), we have

$$\Delta(t) - V\sum_c E\left\{g^{(c)}\left(R_{n_c}^{(c)}\right)|\vec{Q}(t)\right\} \leq D_1 + \sum_c \frac{(N-1)Q_{n_c}^{(c)}}{l_c} - V\sum_c g^{(c)}\left(r_{n_c}^{*(c)}(\epsilon)\right) - 2\epsilon\sum_c Q_{n_c}^{(c)} \quad (45)$$

where D_1 is the constant given in theorem 1. Taking the expectations over the distribution of $\vec{Q}$ and summing over

$t \in \{1, 2, \cdots, M\}$, we have

$$E\left\{L(\vec{Q}(M))\right\} - E\left\{L(\vec{Q}(0))\right\} - V\sum_{\tau=0}^{M-1}\sum_{c} E\left\{g^{(c)}\left(R_{n_c}^{(c)}(\tau)\right)\right\} \leq MD_1 + \sum_{\tau=0}^{M-1}\sum_{c}\frac{(N-1)Q_{n_c}^{(c)}(\tau)}{l_c} - VM\sum_{c} g^{(c)}\left(r_{n_c}^{*(c)}(\epsilon)\right) - 2\epsilon\sum_{\tau=0}^{M-1}\sum_{c} E\left\{Q_{n_c}^{(c)}(\tau)\right\}. \quad (46)$$

By arranging the terms of (46) appropriately and dividing both sides by VM, we have

$$\frac{1}{M}\sum_{\tau=0}^{M-1}\sum_{c} E\left\{g^{(c)}\left(R_{n_c}^{(c)}(\tau)\right)\right\} \geq \sum_{c} g^{(c)}\left(r_{n_c}^{*(c)}(\epsilon)\right) - \frac{D_1 + E\left\{L(\vec{Q}(0))\right\}/M}{V} + \frac{1}{VM}\sum_{\tau=0}^{M-1}\sum_{c}\left(2\epsilon - \frac{(N-1)}{l_c}\right) E\left\{Q_{n_c}^{(c)}(\tau)\right\} \quad (47)$$

where we have used the fact that $L(\vec{Q}(M)) \geq 0$ to obtain (47). Using the Jensen's inequality and taking the limit $M \to \infty$ in (47), we have

$$\liminf_{M\to\infty} \sum_{c} g^{(c)}\left(\bar{r}_{n_c}^{(c)}(M)\right) \geq \sum_{c} g^{(c)}\left(r_{n_c}^{*(c)}(\epsilon)\right) - \frac{D_1}{V}. \quad (48)$$

Therefore, we have proved the utility bound. To prove the backlog bound, we arrange the inequality (46) appropriately and divide both sides by M, we have

$$\frac{1}{M}\sum_{\tau=0}^{M-1}\sum_{c}\left(2\epsilon - \frac{(N-1)}{l_c}\right) E\left\{Q_{n_c}^{(c)}(\tau)\right\} - \frac{E\left\{L(\vec{Q}(0))\right\}}{M} \leq D_1 + VG_{\mathsf{max}}. \quad (49)$$

Note that the above inequalities hold for any $0 < \epsilon \leq \lambda_{\mathsf{max}}$. Also, suppose we choose the same buffer sizes for different flows as assumed in the theorem. By choosing $\epsilon = \lambda_{\mathsf{max}}$ and taking the limit for $M \to \infty$ in (49), we can obtain the backlog bound. The detailed proof of this theorem can be found in [32].

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, no. 12, pp. 1936-1948, Dec. 1992.

[2] L. Georgiadis, M. J. Neely, L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking,* vol. 1, no. 1, pp. 1-144, 2006.

[3] M. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Trans. Networking*, vol. 16, no. 2, April 2008.

[4] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.

[5] H.-W. Lee, E. Modiano, and L. B. Le, "Distributed throughput maximization in wireless networks via random power allocation," *WiOpt 2009*.

[6] P. Giaccone, E. Leonardi and D. Shah, "Throughput region of finite-buffered networks," *IEEE Trans. Parallel Distributed Systems*, vol. 18, no. 2, pp. 251-262, Feb. 2007.

[7] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," *ACM SIGMETRICS 2006*.

[8] S. Sanghavi, L. Bui, and R. Srikant, "Distributed link scheduling with constant overhead," *ACM SIGMETRICS 2007*, June 2007.

[9] X. Lin and N. B. Shroff, " The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Trans. Networking,* vol. 14, no. 2, pp. 302–315, April 2006.

[10] P. Charporkar, K. Kar, and S. Sarkar, "Throughput guarantees through maximal scheduling in wireless networks," *IEEE Trans. Inf. Theory*, vol. 54, no. 2, pp. 572-594, Feb. 2008.

[11] A. Eryilmaz, A. Ozdaglar, and E. Modiano, "Polynomial complexity algorithms for full utilization of multi-hop wireless networks," *IEEE INFOCOM 2007*.

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

[13] G. Zussman, A. Brzezinski, and E. Modiano, "Multihop local pooling for distributed throughput maximization in wireless networks," *IEEE INFOCOM*, April 2008.

[14] C. Joo and N. B. Shroff, "Performance of random access scheduling schemes in multi-hop wireless networks," *IEEE INFOCOM 2007*.

[15] R. Li, L. Ying, A. Eryilmaz, and N. B. Shroff, "A unified approach to optimizing performance in networks serving heterogeneous flows," *IEEE INFOCOM 2009*.

[16] A. Stolyar, "Large deviations of queues under QoS scheduling algorithms," *Allerton 2006*, Sept. 2006.

[17] L. Bui, A. Eryilmaz, R. Srikant, and X. Wu, "Asynchronous congestion control in multi-hop wireless networks with maximal matching-based scheduling," *IEEE/ACM Trans. Networking,* vol. 16, no. 4, pp. 826-839, Aug. 2008.

[18] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Trans. Networking,* vol. 15, no. 6, pp. 1333-1344, Dec. 2007.

[19] L. Ying and S. Shakkottai, "Scheduling in mobile wireless networks with topology and channel-state uncertainty," *IEEE INFOCOM 2009*.

[20] K. Kar, X. Luo, S. Sarkar, "Throughput-optimal scheduling in multi-channel access point networks under infrequent channel measurements," *IEEE Trans. Wireless Commun.,* vol. 7, no. 7, pp. 2619–2629, July 2008.

[21] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inf. Theory,* vol. 39, no. 2, pp. 466–478, Mar. 1993.

[22] A. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Systems,* vol. 50, no. 4, pp. 401–457, Aug. 2005.

[23] A. Baron, R. Ginosar, and I. Keslassy, "The capacity allocation paradox," *IEEE INFOCOM 2009*.

[24] M. Neely, "Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks," *IEEE/ACM Trans. Networking,* vol. 16, no. 5, pp. 1188–1199, Oct. 2008.

[25] L. B. Le, K. Jagannathan, and E. Modiano, "Delay analysis of maximum weight scheduling in wireless ad hoc networks," *CISS'2009*, Mar. 2009.

[26] W. Khan, L. B. Le, and E. Modiano, "Autonomous routing algorithms for networks with wide-spread failures," *IEEE MILCOM 2009,* Oct. 2009.

[27] M. Neely, "Delay analysis for maximal scheduling in wireless networks with bursty traffic," *IEEE INFOCOM 2008*.

[28] G. R. Gupta and N. B. Shroff, "Delay analysis for multi-hop wireless networks," *IEEE INFOCOM 2009*.

[29] L. Ying, S. Shakkottai, and A. Reddy, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE INFOCOM 2009*.

[30] M. J. Neely, "Dynamic power allocation and routing for satellite and wireless networks with time varying channels." PhD thesis, Massachusetts Institute of Technology, 2003.

[31] L. Bui, R. Srikant, and A. Stolyar, "Novel architecture and algorithms for delay reduction in back-pressure scheduling and routing," *IEEE INFOCOM 2009*.

[32] L. B. Le, E. Modiano, and N. B. Shroff, Optimal control of wireless networks with finite buffers. [Online]. Available: http://www.mit.edu/∼longble/pub/finitebuf.pdf