Delay Minimization in Multihop Wireless Networks: Static Scheduling Does It

Sharad Birmiwal, Unnikrishnan Nair, D. Manjunath, Ravi Mazumdar

Delay Minimization in Multihop Wireless Networks: Static Scheduling Does It

Sharad Birmiwal[1], Jayakrishnan Nair[2], D. Manjunath[3], and Ravi R. Mazumdar[1]

[1]Department of Electrical and Computer Engineering, University of Waterloo
[2]Department of Electrical Engineering, California Institute of Technology
[3]Department of Electrical Engineering, IIT Bombay

***Abstract*—In this paper, we address two issues in multihop wireless networks—poor end-to-end delay performance and high per-slot computational overhead of the classical max-weight algorithm. To reduce the end-to-end delay, we first propose a simple modification to the classical maximum weight scheduling algorithm that promotes the use of shorter paths by the packets. The significantly lower delays are shown via simulation. The modification that we suggest does not reduce the schedulable region and has the same complexity as the classical algorithm. Next, we propose a *static* routing and scheduling scheme that is obtained by adapting the classical optimal routing problem of wireline networks to multihop wireless networks. The static scheme slows the timescale of routing and scheduling computations from per-slot to the timescale of change in the network traffic pattern; thus the computation complexity is reduced. We also show, via simulations, that the delay performance in the static scheme is comparable to that of the dynamic scheme that we have proposed.**

I. Introduction

Routing and link scheduling in multihop wireless networks is now a well studied problem for which a large number of variations have also been studied. Much of the recent research has its roots in the work of [1] that introduced a 'maximum-weight' (MW) link-scheduling and routing scheme to maximize the achievable throughput of the network. In this scheme, each node maintains a per-flow queue, and in each slot the scheduling algorithm uses the queue occupancies to assign weights to each link. The allowable schedules (links to activate) is assumed known and the MW schedule is chosen in each slot; notice that routing and scheduling are performed on a per-slot timescale. The requirement to collect the queue length information at a central place, compute the MW schedule and disseminate it to all the nodes implies that the scheme is only 'ideal' and a more practical scheme is needed. Also, a well known drawback of the MW algorithm is that the end-to-end delays at low and medium loads are high, and can in some cases be higher than at higher loads [2]. This paper addresses both the computation and the delay issues of the MW algorithm.

Our starting point is to consider a rather simple modification to the standard MW algorithm in which the link-weights are adjusted to promote the use of shorter paths in the network (Section II). As we will see, this leads to substantially lower delays, without any reduction of the stability region. However, this new algorithm retains the per-slot computational overhead of the original MW algorithm.

Next, seeking to address simultaneously the computation and the delay issues of the MW algorithm, we adapt the delay minimizing optimal routing problem of wireline networks to wireless networks (Section III). We begin by assuming that the flow volumes between source-destination pairs in the network are known. We then define a *static* optimal routing and scheduling problem for wireless networks. The solution yields a *static* routing and scheduling scheme; the qualifier 'static' means that routing and scheduling decisions do not depend on the instantaneous network state. We thus slow the timescale of routing and scheduling computations from per-slot to the timescale of change in the network traffic pattern. The wireless setting of course introduces several complexities, e.g., the link capacities also become variables in the optimization and this makes the objective function non-convex.

Unlike dynamic routing and scheduling algorithms, our proposed static schemes are non-opportunistic, in that they do not exploit instantaneous state information to route and schedule packets. In spite of this, as we will see in Section IV, they achieve a delay performance comparable to the best dynamic schemes.

A. Previous Work

Most of the results in the literature, starting with the seminal work of [1], propose a dynamic routing and link scheduling algorithm. This in turn is some variant of a maximum weight (where 'weight' is suitably defined) schedule in each slot or group of slots. The schedule is typically designed to maximize the throughput of the network. Throughput optimality is usually derived via Foster's criteria by a suitable choice of a Lyapunov function to prove the stability of an underlying Markov chain.

Early follow-ups to [1] concentrated on making the scheduling algorithms more practical. Results on delay performance, typically mean delay, have been available, e.g., [3], [4]. More detailed analysis are now available, e.g., [4]. Algorithms to determine the schedules in each slot to reduce the mean delay have also been proposed in recent literature, e.g., [5].

There are several new results for networks with only one-hop flows, e.g., [6]. Scheduling to guarantee per flow end-to-end delay bounds are derived in [6]; however a constant factor

978-3-901882-45-6/2012 - Copyright is with IFIP

reduction in the schedulable region is introduced. [7] describes a randomized algorithm to reduce the per-hop delay, also with a reduction in the schedulable region.

These and other such scheduling algorithms are rather complicated, and even complex (in terms of the computation times), and need to be performed frequently, possibly in every slot. Furthermore, all of them lead to a reduction in the schedulable region. This can mean a possible negation of the effect of the delay minimization scheduling algorithm, especially at loads closer to the reduced capacity of the new algorithm.

In this paper we hark back to wireline networks, where multipath routing for average delay minimization is an old jungle problem with early algorithms dating back to the works [8]–[11]. Taking inspiration from this seminal line of work, we will formulate and analyze a static routing and scheduling algorithm to minimize the mean delay in a multihop wireless network. We will see that the extension is non-trivial because unlike in the wireline network, the link capacities are now variables! We will analyze the problem in some detail.

II. Adapting Link Weights to Prioritize Shorter Paths

We now describe a simple variation of the MW dynamic routing and scheduling algorithm, the *weighted back-pressure algorithm* (WBPA) which reduces delay without reduction of throughput. Consider a wireless network, defined by a set $\mathcal{N}$ of nodes, and a set $\mathcal{L}$ of directed links connecting the nodes. Assume that time is slotted and that in each slot, a set of links, say s, may be scheduled to transmit. A set $\mathcal{F}$ of flows uses the network, with s_f, d_f and λ_f being, respectively, the source node, the destination node, and the arrival rate in packets per slot of flow $f \in \mathcal{F}$. The set of all feasible schedules, $\mathcal{S}$, is assumed to be given a priori.

Recall that in [1], link weights are defined by $w_l = \max_{f\in\mathcal{F}}(Q_{f,b_l} - Q_{f,e_l})$, where $Q_{f,i}$ is the backlog of the queue for flow f at node i, and b_l and e_l denote, respectively, the transmitting and receiving node on link l. Let $Q = (Q_{f,i}, f \in \mathcal{F}, i \in \mathcal{N})$. Define modified link weights $w'_l = \max_{f\in\mathcal{F}} \beta_{f,l}(Q_{f,b_l} - Q_{f,e_l})^+$. The factor $\beta_{f,l} > 0$ allows a link on a shorter path to have a higher weight even with small backlogs. Consider the packet arrival and service model of [1] and modify the routing and scheduling algorithm as follows. Let $\mathcal{Q}_z := \{Q : \sum_{f\in\mathcal{F}}\sum_{i\in\mathcal{N}} Q_{f,i}^2 < z\}$ where Q is the state of the Markov chain like in [1]. Now consider the following routing and scheduling algorithm with the choice of schedule given by

$$\Pi = \begin{cases} \arg\max_{s\in\mathcal{S}} \sum_{l\in\mathcal{S}} w'_l & \text{if } Q \in \mathcal{Q}_z \\ \arg\max_{s\in\mathcal{S}} \sum_{l\in\mathcal{S}} w_l & \text{otherwise.} \end{cases} \tag{1}$$

Here, if the system state is in $\mathcal{Q}_z$, then routing and scheduling is performed using $\{w'_l\}$, otherwise it is performed using $\{w_l\}$. The routing and scheduling in states in $\mathcal{Q}_z$ affect the transition probabilities of the Markov chain in a finite set of states. Thus the stability remains unaffected. This follows from the analysis of [1] because the drift in the 'large' states outside $\mathcal{Q}_z$, is still negative. $\mathcal{Q}_z$ can be made arbitrarily large but finite. The choice of $\beta_{f,l}$ is discussed next.

Let $H_{f,i}$ be the length of the shortest path (number of hops) from node i to node d_f. Let

$$\beta_{f,l} := (|\mathcal{N}| - H_{f,i})^\alpha \quad \text{for } l \in O_i, \tag{2}$$

where O_i is the set of all egress links at i, and $\alpha > 1$. With this choice of $\beta_{f,l}$, links on shorter paths towards the destination have higher preference, thus preventing packets from being misdirected frequently at low and moderate loads, as in the MW algorithm. In other words, our choice of $\beta_{f,l}$ creates *virtual* back-pressures that favor shorter paths even at low and moderate loads. Contrast this with the algorithm of [2] where shorter paths are used until the traffic intensity of unity is attained; WBPA is more opportunistic and avoids congested spots because of dynamic routing. We evaluate WBPA via simulations and see in Section IV that it achieves significantly lower delays compared to the MW algorithm.

III. A Static Formulation for Delay Minimization

In this section, we pose the static optimization problem for minimizing average delay, along the lines of [8]–[10]. Consider an ergodic schedule in which schedule s is used for fraction ϕ_s of time. We will say that $\phi := (\phi_s,\ s \in \mathcal{S})$ is the *scheduling* in the network. Clearly, $\phi_s \geq 0$ and $\sum_{s\in\mathcal{S}} \phi_s = 1$; this defines the set Φ of feasible scheduling vectors. The capacity (in packets/slot) of link l is thus $\mu_l := \sum_{s\in\mathcal{S}:\ l\in s} \phi_s$. We will assume ergodic routing and let $x_{f,l}$, $f \in \mathcal{F}$ and $l \in \mathcal{L}$, denote the rate at which packets of flow f are transmitted on link l. Let $x_f := (x_{f,l}, l \in \mathcal{L})$, $x := (x_f, f \in \mathcal{F})$. We will say that x is the *routing* in the network. Clearly, for each $f \in \mathcal{F}$, the vector x_f is non-negative and satisfies the flow conservation constraints. These conditions define the feasibility set X_f for x_f; let $X := \prod_{f\in\mathcal{F}} X_f$. The rate at which packets arrive to be transmitted on link l is $\gamma_l := \sum_{f\in\mathcal{F}} x_{f,l}$.

We assume that the mean packet delay on link l can be expressed as a function of γ_l and μ_l, say, $D_l(\gamma_l, \mu_l)$ (this is a standard assumption in the literature; e.g., see [9], [10], [12]). The mean packet delay in the network is minimized by minimizing the average number of packets in the system, i.e., by minimizing $G := \sum_l \gamma_l D_l(\gamma_l, \mu_l)$ (from Little's Law).

We make the following regularity assumptions on the link delay function D_l. Define $Z := \{(\gamma_l, \mu_l) \in \mathbb{R}^2 \ :\ \mu_l \geq \epsilon,\ 0 \leq \gamma_l < \mu_l\}$, where $\epsilon > 0$ is a small positive constant. We impose a lower bound ϵ on μ_l to eliminate a possible discontinuity at the origin of the function $\gamma_l D_l(\gamma_l, \mu_l)$. D_l is twice continuously differentiable over Z, and is defined to be ∞ outside this set. Over its effective domain, $D_l(\gamma_l, \mu_l)$ is strictly increasing and convex with respect to γ_l, and strictly decreasing and convex with respect to μ_l. Finally, $\lim_{\gamma_l\uparrow\mu_l} D_l(\gamma_l, \mu_l) = \infty$ for all $\mu_l \geq \epsilon$.

We are interested in optimally choosing routing x and scheduling ϕ to minimize the mean packet delay in the network. Formally, this optimization problem can be stated

as follows.

$$\begin{array}{ll} \text{Minimize} & G(x,\phi) \\ \text{s.t.} & (x,\phi) \in X \times \Phi \end{array} \quad \text{(P1)}$$

We assume that the packet arrival rates λ_f, $f \in \mathcal{F}$, are within the capacity region of the network; this ensures that the above optimization problem is feasible. Also, note that we do not need to explicitly include a stability constraint for each link, since the objective function is defined to be ∞ if this constraint is violated on any link.

The function $D_l(\gamma_l, \mu_l)$ is meant to model the average delay on link l. A standard approach would be to model D_l using the Pollaczek-Khinchin formula for the M/G/1 FCFS queue (as in [9], [10], [12]), with γ_l and μ_l denoting the arrival rate and the service rate respectively. Note that in all these models, $D_l(0,\mu_l) = \frac{1}{\mu_l}$. The following proposition states that for such 'standard' queueing delay models, the average number of packets in the queue (including the one being served), given by $\gamma_l D_l(\gamma_l,\mu_l)$ is not jointly convex with respect to (γ_l,μ_l). This suggests that (P1) is in general a non-convex optimization problem for standard queueing delay models.

***Proposition** 1:* If D_l satisfies the regularity conditions listed above, and $D_l(0,\mu_l) = \frac{1}{\mu_l}$, then $\gamma_l D_l(\gamma_l,\mu_l)$ cannot be convex over the interior of Z.

Proof: Let $F(\gamma,\mu) := \gamma D_l(\gamma,\mu)$. Pick $\mu_0 > \epsilon$. Since $\frac{\partial D_l(\gamma,\mu)}{\partial \mu}$ is assumed to be continuous over Z, $\lim_{\gamma \downarrow 0} \frac{\partial D_l(\gamma,\mu_0)}{\partial \mu} = -\frac{1}{\mu_0^2}$.

A necessary condition for F to be convex over Z is that the determinant of its Hessian (denoted by $\det(\nabla^2 F)$) be non-negative. $\det(\nabla^2 F)$ is easily computed to be the following.

$$\begin{aligned} \det(\nabla^2 F(\gamma,\mu)) &= \gamma \frac{\partial^2 D}{\partial \mu^2}\left(2\frac{\partial D}{\partial \gamma} + \gamma \frac{\partial^2 D}{\partial \gamma^2}\right) \\ &\quad - \left(\frac{\partial D}{\partial \mu} + \gamma \frac{\partial^2 D}{\partial \mu \partial \gamma}\right)^2. \end{aligned}$$

Since D is assumed to be twice continuously differentiable over Z, all the derivatives on the right hand side of the above equation must be bounded over the compact set $\{(\gamma,\mu_0) : 0 \le \gamma \le \mu_0/2\}$. Therefore,

$$\lim_{\gamma \downarrow 0} \det(\nabla^2 F(\gamma,\mu_0)) = -\frac{1}{\mu_0^4} < 0.$$

This implies that for small enough $\gamma > 0$, $\det(\nabla^2 F(\gamma,\mu_0)) < 0$. Therefore, F cannot be convex over Z. ∎

If we fix the scheduling vector ϕ, then it is easy to see that (P1) is convex and reduces to the optimal routing problem of [9], [10]. Similarly, if we fix the routing x, then (P1) reduces to a convex optimization, and is therefore easy to solve. Proposition 1 above suggests that with both routing and scheduling as variables, (P1) is in general a non-convex optimization. We now provide algorithms to compute a local minimum of (P1).

First, we characterize the set of local minimizers of (P1). Define $J := \{(x,\phi) \in X \times \Phi \mid G(x,\phi) < \infty\}$. Since we have assumed that the optimization (P1) is feasible, the set J is non-empty. Define

$$J^* := \left\{ (\bar{x},\bar{\phi}) \in J \;\middle|\; \begin{array}{l} \bar{x} \in \arg\min_{x \in X} G(x,\bar{\phi}) \\ \bar{\phi} \in \arg\min_{\phi \in \Phi} G(\bar{x},\phi) \end{array} \right\}.$$

***Lemma** 1:* J^* is the set of local minimizers of G over $X \times \Phi$.

Proof: The proof follows easily from the fact that X and Φ are convex, and that a minimization with respect to either variable x or ϕ keeping the other fixed is a convex minimization.

Since X and Φ are convex sets, $(\bar{x},\bar{\phi})$ is a local minimizer of G over $X \times \Phi$ iff.

$$\begin{aligned} & \nabla_x G(\bar{x},\bar{\phi}) \cdot (x - \bar{x}) + \nabla_\phi G(\bar{x},\bar{\phi}) \cdot (\phi - \bar{\phi}) \ge 0 \\ & \qquad \forall \quad (x,\phi) \in X \times \Phi \\ \iff & \begin{array}{ll} \nabla_x G(\bar{x},\bar{\phi}) \cdot (x - \bar{x}) \ge 0 & \forall \quad x \in X, \\ \nabla_\phi G(\bar{x},\bar{\phi}) \cdot (\phi - \bar{\phi}) \ge 0 & \forall \quad \phi \in \Phi \end{array} \\ \iff & \begin{array}{l} \bar{x} \in \arg\min_{x \in X} G(x,\bar{\phi}), \\ \bar{\phi} \in \arg\min_{\phi \in \Phi} G(\bar{x},\phi). \end{array} \end{aligned}$$

∎

The preceding lemma states that the set of local minimizers of (P1) are precisely the tuples (x,ϕ) satisfying the property that with routing vector fixed at x, the scheduling vector ϕ is optimal, and vice-versa. Clearly, if G is convex, then J^* is the set of global minimizers of G over $X \times \Phi$.

We now provide two approaches to compute a local minimizer of (P1). The first is a block descent algorithm, which cyclically performs (convex) optimizations with respect to the routing and the scheduling. Next, we describe a class of algorithms that chooses between a routing update and a scheduling update in each iteration.

Block descent algorithm

We now describe an algorithm (see Algorithm 1) for computing a local minimum of (P1) based on the block descent algorithm (see Proposition 2.7.1 of [13]). Let $(x^0,\phi^0) \in J$ denote a starting feasible point for (P1). Such a point is easy to compute since J is defined by linear constraints. The algorithm is parametrized by a positive constant c.

Algorithm 1 Block descent

for $i \ge 0$ **do**
$\phi^{i+1} \leftarrow \arg\min_{\phi \in \Phi} G(x^i,\phi) + \frac{1}{c} \parallel \phi - \phi^i \parallel^2$
$x^{i+1} \leftarrow \arg\min_{x \in X} G(x,\phi^{i+1}) + \frac{1}{c} \parallel x - x^i \parallel^2$
end for

Note that the optimizations involved in each iteration of Algorithm 1 are convex, and hence can be performed by standard techniques. The following lemma guarantees the convergence of this block descent algorithm.

***Lemma** 2:* The sequence $\{(x^i,\phi^i)\}$ generated by Algorithm 1 converges to an element of J^*.

Proof: Invoking Proposition 2.7.1 of [13], the sequence $\{(x^i,\phi^i)\}$ converges to a local minimum of (P1) if, for any

feasible point (x^0, ϕ^0) of (P1), the optimizations

$$\min_{x \in X} G(x, \phi^0) + \frac{1}{c} \parallel x - x^0 \parallel^2, \text{ and}$$
$$\min_{\phi \in \Phi} G(x^0, \phi) + \frac{1}{c} \parallel \phi - \phi^0 \parallel^2$$

yield unique minimizers. This uniqueness follows easily from the fact that the objective function in both minimizations is strictly convex. ■

A class of iterative algorithms

Next, we introduce a class of iterative algorithms (see Algorithm 2) that guarantee convergence to J^*. One such algorithm is specified by two functions: a 'routing update' function, $A_{RT} : J \to X$, which provides descent by updating the routing vector, and a 'scheduling update' function $A_{SC} : J \to \Phi$, which provides descent by updating the scheduling vector. Algorithm 2 describes the algorithm derived from A_{RT} and A_{SC}. As before, we assume that a starting feasible point $(x^0, \phi^0) \in J$ is available.

Algorithm 2 Algorithm for solving (P1)

for $i \geq 0$ **do**
  if $G(A_{RT}(x^i, \phi^i), \phi^i) \leq G(x^i, A_{SC}(x^i, \phi^i))$ **then**
    $x^{i+1} \leftarrow A_{RT}(x_i, \phi_i)$; $\phi^{i+1} \leftarrow \phi^i$
  else
    $x^{i+1} \leftarrow x^i$; $\phi^{i+1} \leftarrow A_{SC}(x^i, \phi^i)$
  end if
end for

In each iteration, the algorithm performs either the routing update or the scheduling update, whichever produces the greater descent in the objective function value. The following theorem states that so long as the routing update A_{RT} and the scheduling update A_{SC} are continuous, and satisfy certain simple descent criteria, the sequence $\{(x^i, \phi^i)\}$ generated by the algorithm converges to the set J^*. Most importantly, the descent properties required of the routing and scheduling updates are decoupled, implying these update rules can be designed independently.

Theorem 1: For any $(\hat{x}, \hat{\phi}) \in J$ satisfying $G(\hat{x}, \hat{\phi}) \leq G(x_0, \phi_0)$, assume that the routing update A_{RT} satisfies the following descent property.

(a) If $\hat{x} \in \arg\min_{x \in X} G(x, \hat{\phi})$, then $G(A_{RT}(\hat{x}, \hat{\phi}), \hat{\phi}) = G(\hat{x}, \hat{\phi})$;

(b) if $\hat{x} \notin \arg\min_{x \in X} G(x, \hat{\phi})$, then there exists $\delta > 0$ such that for all $(\tilde{x}, \tilde{\phi}) \in X \times \Phi$ satisfying $\parallel (\tilde{x}, \tilde{\phi}) - (\hat{x}, \hat{\phi}) \parallel < \delta$, $G(A_{RT}(\tilde{x}, \tilde{\phi}), \tilde{\phi}) < G(\hat{x}, \hat{\phi})$.

Similarly, for any $(\hat{x}, \hat{\phi}) \in J$, satisfying $G(\hat{x}, \hat{\phi}) \leq G(x_0, \phi_0)$, assume that the scheduling update A_{SC} satisfies the following descent property.

(a') If $\hat{\phi} \in \arg\min_{\phi \in \Phi} G(\hat{x}, \phi)$, then $G(\hat{x}, A_{SC}(\hat{x}, \hat{\phi})) = G(\hat{x}, \hat{\phi})$;

(b') if $\hat{\phi} \notin \arg\min_{\phi \in \Phi} G(\hat{x}, \phi)$, then there exists $\delta > 0$ such that for all $(\tilde{x}, \tilde{\phi}) \in X \times \Phi$ satisfying $\parallel (\tilde{x}, \tilde{\phi}) - (\hat{x}, \hat{\phi}) \parallel < \delta$, $G(\tilde{x}, A_{SC}(\tilde{x}, \tilde{\phi})) < G(\hat{x}, \hat{\phi})$.

Then every limit point of the sequence $\{(x^i, \phi^i)\}$ generated by Algorithm 2 lies in J^*.

Proof: The descent assumptions on A_{RT} and A_{SC} imply that $\{G(x_i, \phi_i)\}$ is a non-increasing sequence (bounded below by the value of G at the solution of (P1)). Therefore, there exists $G^* \in \mathbb{R}$ such that

$$\lim_{i \to \infty} G(x_i, \phi_i) = G^*, \qquad G(x_i, \phi_i) \geq G^* \ \forall \ i. \tag{3}$$

Now, since the sequence $\{(x_i, \phi_i)\}$ is contained in the compact space $X \times \Phi$, it must have a converging subsequence $(x_{i(n)}, \phi_{i(n)}) \overset{n \uparrow \infty}{\to} (x^*, \phi^*)$. Note that since G is continuous, $G(x^*, \phi^*) = G^*$. We need to prove that $(x^*, \phi^*) \in J_r^*$.

Let us assume (for the sake of obtaining a contradiction) that $(x^*, \phi^*) \notin J_r^*$. Then from the definition of J^*, at-least one of the following conditions must hold.

(i) $x^* \notin \arg\min_{x \in X} G(x, \phi^*)$
(ii) $\phi^* \notin \arg\min_{\phi \in \Phi} G(x^*, \phi)$

Let us say (i) holds. Then from the descent assumption on A_{RT}, there exists $\epsilon > 0$ such that for all $(\tilde{x}, \tilde{\phi}) \in X \times \Phi$ satisfying $\parallel (\tilde{x}, \tilde{\phi}) - (x^*, \phi^*) \parallel < \epsilon$, we must have $G(A_{RT}(\tilde{x}, \tilde{\phi}), \tilde{\phi}) < G(\hat{x}, \hat{\phi})$. Since $(x_{i(n)}, \phi_{i(n)}) \overset{n \uparrow \infty}{\to} (x^*, \phi^*)$, there exists $n_0 \in \mathbb{N}$ such that $\parallel (x_{i(n_0)}, \phi_{i(n_0)}) - (x^*, \phi^*) \parallel < \epsilon$, which implies that

$$G(A_{RT}(x_{i(n_0)}, \phi_{i(n_0)}), \phi_{i(n_0)}) < G(x^*, \phi^*) = G^*. \tag{4}$$

Algorithm 2 picks $(x_{i(n_0)+1}, \phi_{i(n_0)+1})$ such that

$$G(x_{i(n_0)+1}, \phi_{i(n_0)+1}) \leq G(A_{RT}(x_{i(n_0)}, \phi_{i(n_0)}), \phi_{i(n_0)}). \tag{5}$$

Combining (4) and (5), we conclude that $G(x_{i(n_0)+1}, \phi_{i(n_0)+1}) < G^*$, which is a contradiction. This means Condition (i) above cannot hold. Using an identical argument, it can be shown that Condition (ii) also cannot hold. Therefore, $(x^*, \phi^*) \in J_r^*$. ■

Since (P1) reduces to a convex optimization when either the routing vector or the scheduling vector is held fixed, the update rules A_{RT} and A_{SC} may be designed by standard techniques in convex optimization. For example, these update rules can be designed using projected gradient methods [13].

IV. Evaluation

In this section we evaluate the mean delay performance, via simulations, of the weighted back-pressure algorithm (a dynamic routing and scheduling algorithm that generalizes the max-weight algorithm), and the static formulation in (P1). The performance of the two algorithms are compared with those of the max-weight algorithm and the model of [2]. The simulations are on a 4×4 grid topology considered under two scenarios. The first, with only two flows (see Fig. 1), highlights the role of α and z. The second scenario has a flow between every pair of source and destination nodes, i.e., there are 240 interacting flows.

A. Weighted Back-pressure Algorithm

The weighted back-pressure algorithm assigns a higher priority to shorter paths to the sink and thus reduces the mean end-to-end delay. The priority is increased by introducing multiplicative weights, $\beta_{f,l}$, as in (2) to links that are on shorter paths of a flow. The increased link weights make activation of shorter paths more likely. A closer inspection reveals that the priority of flows on links to nodes with smaller distance to sink , let us call them short flows (as compared to shorter paths of a single flow previously), increases. As α increases, the (maximal) schedule activated in a slot is now usually determined by the backlogs on shorter flows and shorter paths. In Fig. 2, we plot the mean delay of the short flow (Flow 2) for different values of α. The intuition that we provide above is verified in this plot. We also observe that the mean delay of the longer flow, Flow 1 (Fig. 3), shows no particular trend with changing α as the schedules are dominantly determined by the backlogs of Flow 2.

The effect of varying z is examined next. For a given z, the algorithm switches to max-weight algorithm when the queue backlogs are sufficiently high. Thus, for arrival rates beyond a certain magnitude, the mean delay resembles that of the max-weight algorithm. As z increases, this transitioning arrival rate increases as greater queue backlogs are required before switching to the max-weight algorithm. Such intuition is verified by our simulations (see Fig. 4).

B. The Static System

The delay performance when x and ϕ are chosen as a solution to (P1) is evaluated in this section. Clearly, several routing and scheduling schemes can achieve x and ϕ and the actual delay performance will depend on the chosen scheme. Let us first consider the static routing and scheduling schemes that are possible. A simple static routing algorithm suggests itself. At node i, a packet of flow f is routed on link l with probability $p_{f,l}$ independent of all other packets, where $p_{f,l} := \frac{x_{f,l}}{\sum_{l' \in O_i} x_{f,l'}}$. A simple scheduler, we call the *independent scheduler,* also suggests itself. In each slot, schedule $s \in \mathcal{S}$ is chosen with probability ϕ_s independent of all other slots. In this scheme, the interval of time between the activation of a schedule has a high variance. Reducing this variance can reduce the mean delays on the links. This leads us to the *max-delta scheduler* which is a generalization of the round-robin scheduler. Let $\Pi(t)$ be the schedule activated in slot t and define $\hat{\phi}_s(t) := \frac{1}{t}\sum_{u=0}^{t-1} \mathbf{1}(\Pi(u) = s)$. $\hat{\phi}_s(t)$ is the fraction of time schedule s has been activated up to time t. The schedule activated during time slot t is determined by $\Pi(t) = \arg\max_{s\in\mathcal{S}}\{\phi_s - \hat{\phi}_s(t)\}$ where ties are broken according to some pre-determined rule. Observe that this gives us a deterministic sequence of schedules which can be pre-computed.

Fig. 5 shows the well known poor delay performance of the MW algorithm. We see that independent scheduling with static routing performs significantly better than the MW algorithm at low loads because the static routing prohibits random walk like behavior. However, since the schedules are chosen randomly in each slot, the interval between consecutive link activations can have a high variance. This and the fact that the schedule in each slot is independent of the queue occupancies causes the delay to continue to be high for low loads; the scheme of [2] offers a better performance here.

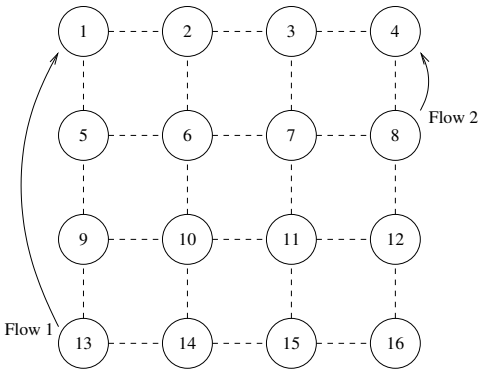

Fig. 1. The 4×4 grid topology with 2 flows

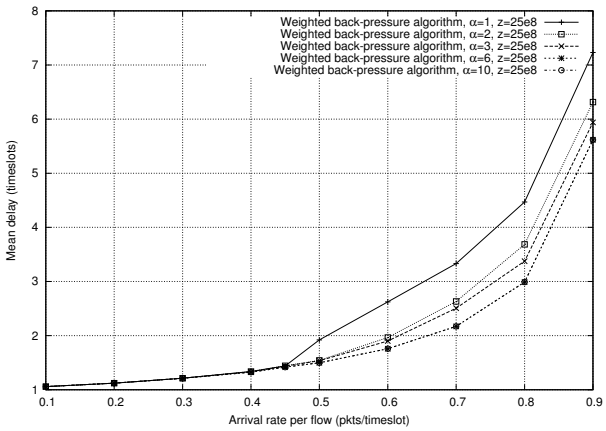

Fig. 2. The effect of varying α on the mean delay of the short flow in the topology of Fig. 1.

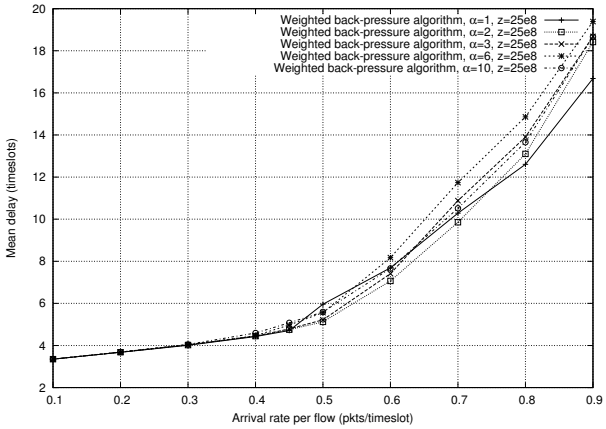

Fig. 3. The effect of varying α on the mean delay of the long flow in the topology of Fig. 1.

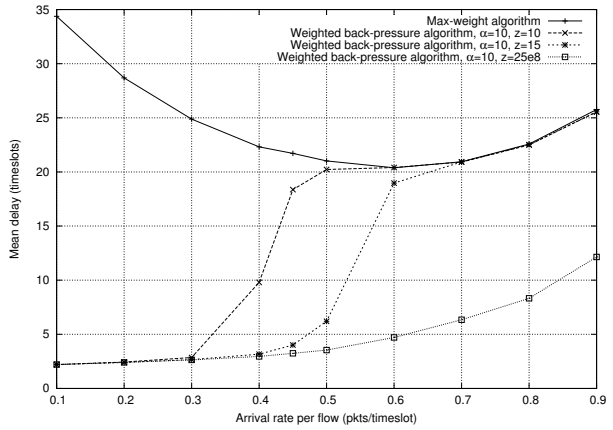

Fig. 4. Effect of varying z on the mean delay of the network in Fig. 1.

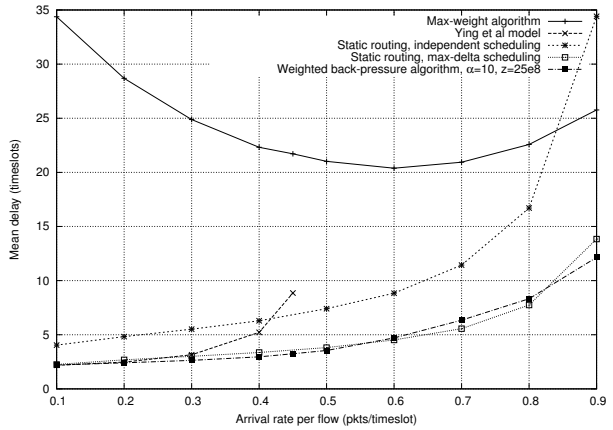

Fig. 5. Mean end-to-end delay averaged over both flows when x and ϕ are chosen from (P1) and implemented using the static routing and scheduling schemes. Performance of schemes of [1] and [2] are also provided.

Note that in the scheme of [2], the shortest path routes are saturated at approximately 0.45 packets/slot for the first scenario and the network does not stabilize for higher loads. The max-delta scheduler with static routing performs better than the independent scheduler. We attribute this to the reduced variance of the inter-activation time of the links.

Figure 5 also shows the results for the WBPA discussed in Section II. This algorithm performs significantly better than the MW algorithm and the algorithm of [2]. Remarkably, the performance of the WBPA is comparable to that of the static-routing max-delta scheduling scheme. Similar trends are observed in the scenario with 240 flows (see Fig. 6) though the WBPA performs poorer than max-delta scheduling possibly because of high priority to short flows.

V. Conclusion

This paper focuses on reducing mean delay in multihop wireless networks. Our formulation relies on solving a non-linear optimization problem for static scheduling and static routing variables. We prove an interesting property: no realistic delay function is convex for wireless networks, i.e., when

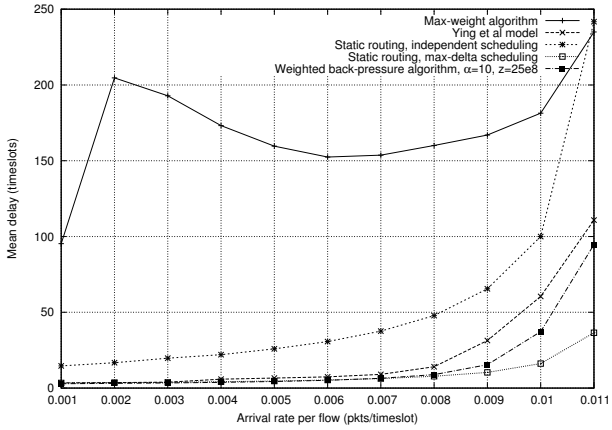

Fig. 6. Mean end-to-end delay averaged over 240 flows when x and ϕ are chosen from (P1) and implemented using the static routing and scheduling schemes. Performance of schemes of [1] and [2] are also provided.

both capacity and routing parameters are variables. We present two self-evident implementations and show that implementing static schemes can achieve significantly better mean delays, a benefit aside from the reduced complexity. We benchmark our results against the popular max-weight algorithm scheme in [1], the model of [2], and by a weighted back-pressure algorithm, we propose, that is of independent interest.

References

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

[2] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 841–854, June 2011.

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

[4] M. J. Neely, "Delay analysis for maximal scheduling with flow control in wireless networks with bursty traffic," *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1146–1159, August 2009.

[5] L. Huang and M. J. Neely, "Delay efficient scheduling via redundant constraints in multihop networks," *Performance Evaluation*, 2011.

[6] S. Jagabathula and D. Shah, "Optimal delay scheduling in networks with arbitrary constraints," in *Proceedings of ACM SIGMETRIC/Performance*, 2008.

[7] M. Lotfinezhad, B. Liang, and E. S. Sousa, "On stability region and delay performance of linear memory randomized scheduling for randomized scheduling for time varying networks," *IEEE Transactions on Networking*, vol. 17, no. 6, pp. 1860–1873, December 2009.

[8] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation network design," *Networks*, vol. 3, pp. 97–133, 1973.

[9] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Transactions on Communications*, vol. 25, no. 1, pp. 733–785, January 1977.

[10] D. Bertsekas, E. Gafni, and R. Gallager, "Second derivative algorithms for minimum delay distributed routing in networks," *IEEE Transactions on Communications*, vol. 32, no. 8, pp. 911–919, August 1984.

[11] J. Tsitsiklis and D. Bertsekas, "Distributed asynchronous optimal routing in data networks," *IEEE Transactions on Automatic Control*, vol. 31, no. 4, pp. 325–332, April 1986.

[12] Y. Xi and E. M. Yeh, "Node-based optimal power control, routing, and congestion control in wireless networks," *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 4081–4106, September 2008.

[13] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1999.