Designing Multihop Wireless Backhaul Networks with Delay Guarantees

Girija Narlikar Gordon Wilfong Lisa Zhang
Bell Laboratories, Lucent Technologies
700 Mountain Ave, Murray Hill, NJ 07974
{girija,gtw,ylz}@research.bell-labs.com

***Abstract*— As wireless access technologies improve in data rates, the problem focus is shifting towards providing adequate backhaul from the wireless access points to the Internet. Existing wired backhaul technologies such as copper wires running at DSL, T1, or T3 speeds can be expensive to install or lease, and are becoming a performance bottleneck as wireless access speeds increase. Longhaul, non-line-of-sight wireless technologies such as WiMAX (802.16d) hold the promise of enabling a high speed wireless backhaul as a cost-effective alternative. However, the biggest challenge in building a wireless backhaul is achieving guaranteed performance (throughput and delay) that is typically provided by a wired backhaul.**

This paper explores the problem of efficiently designing a multihop wireless backhaul to connect multiple wireless access points to a wired gateway. In particular, we provide a generalized link activation framework for scheduling packets over this wireless backhaul, such that *any* existing wireline scheduling policy can be implemented locally at each node of the wireless backhaul. We also present techniques for determining good interference-free routes within our scheduling framework, given the link rates and cross-link interference information. When a multihop wireline scheduler with worst case delay bounds (such as WFQ or Coordinated EDF) is implemented over the wireless backhaul, we show that our scheduling and routing framework guarantees approximately twice the delay of the corresponding wireline topology. Finally, we present simulation results to demonstrate the low delays achieved using our framework.

I. Introduction

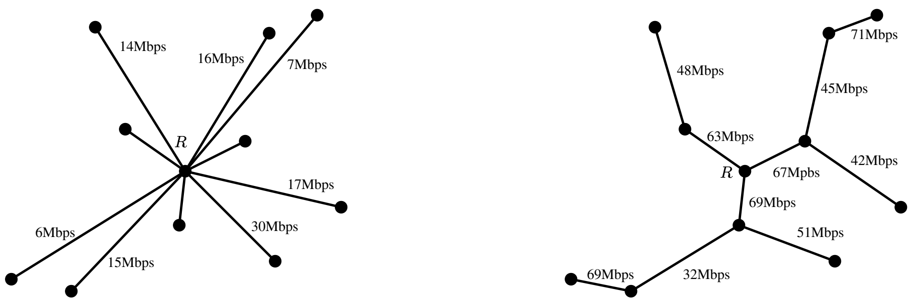
Wireless access technologies such as 802.11x and 3G/4G have been improving in data rates and reducing in cost, paving the way for ubiquitous, high-speed broadband wireless coverage. The biggest challenge in making this vision a reality, however, is to cost effectively connect all the access points or base stations to the Internet using some backhaul technology. Current backhaul technologies such as T1 or DSL lines are fast becoming a bottleneck as access speeds rise, and faster wired technologies such as T3 or fiber connections can be expensive to lease or install in a number of scenarios.

With the technical improvements and standardization of long-haul, non-line-of-sight (NLOS) wireless technologies such as WiMAX (802.16d, also called 802.16-2004 [16]), wireless backhaul is fast becoming a cost effective alternative to wired technologies. The recent standardization of the WiMAX standard is expected to drive down prices of WiMAX-compliant hardware, making it possible for service providers to extensively deploy WiMAX-based backhaul links at competitive costs. The added benefits of a wireless backhaul include ease of deployment and the ability to bypass an incumbent wireline carrier's networks. Further, with spectral availability widening globally (esp. in the 5GHz range) standardized wireless backhaul equipment can now be deployed in a large number of locations. Wireless backhaul networks will be useful to provide high speed Internet access to residential, small and medium business customers, as well as Internet access for WiFi hot spots and cellular base stations. Public safety services and private networks could also be built using wireless backhaul links. A detailed business case analysis was recently conducted for a number of these applications [32], showing that WiMAX-based backhaul deployments can be profitable in a wide variety of demographic environments.

A *multihop* wireless architecture for backhaul has the potential to provide ubiquitous, high-speed wireless access to consumers. In principle, wireless multihop networks are easy to deploy and expand incrementally, can adjust to failures or changing traffic demands by reconfiguring backhaul routes, and can provide ubiquitous service, especially in areas with no installed wired base. Further, multihopping allows us to reduce the distances over which the access points need to transmit on the backhaul links. This can help increase network throughput due to lower pathloss and better spatial reuse (see Figure 1 for an example). Recent commercial deployments of mesh backhauls in the real world are beginning to demonstrate some of these advantages. However, several challenges remain in allowing multihop wireless backhaul networks to match the throughput and delay guarantees of wired backhauls. As shown in Figure 1, multihopping can help boost throughput, but the network must now be more carefully scheduled to reduce interference and maintain low delays over multiple hops. In particular, providing guaranteed rates while keeping end-to-end delays low is necessary for any backhaul network that will carry delay sensitive traffic (such as VoIP, video and interactive applications).

In this paper, we explore the problem of efficiently designing a multihop, wireless backhaul to connect multiple wireless access points to a wired gateway. In particular, we make the following novel contributions:

1) We provide a simple yet generalized link activation framework, which we call the Even-Odd framework, for scheduling packets over this wireless backhaul. The Even-Odd framework activates each link in alternate timeslots, and applies subchannelization to adjust link bandwidths and to allow access points to receive simultaneously over multiple links. Combined with our interference-aware routing (see below) and a TDMA MAC layer, the Even-Odd framework allows us to eliminate interference and map a wireless backhaul into an equivalent, half-idle wireline network. As a result, *any* existing wireline scheduling policy can be implemented *locally* at each access point in the wireless backhaul.
2) We present an optimal formulation as well as heuristic approaches to constructing efficient backhaul routes. Given the channel rates for every point-to-point link and the set of link pairs that interfere with each other, our routing solutions aim to maximize the throughput demands of all nodes while ensuring that interfering links are not simultaneously active in the Even-Odd framework. To avoid interference, routes are constructed by first labeling each access point as "even" or "odd", and only allowing routes to contain links connecting access points with different labels (parities).



(a) f = 1552 Kbps (b) f = 6692 Kbps

Fig. 1. (a) A one-hop topology connecting ten access points to the wired access point R, (b) a multihop topology for the same set of access points. Link capacities were generated using a standard pathloss model. Here f is the backhaul throughput (uplink + downlink) for *each* access point. We assume spatial reuse in the network, where each access point may be sending or receiving over at most one link at any time.

3) Our Even-Odd framework provides the first step towards efficiently bounding end-to-end delay in a multihop wireless backhaul, allowing it to become a practical alternative to wired backhauls. A careful analysis shows that when a multihop wireline scheduler with worst case delay bounds (such as WFQ [28] or Coordinated EDF [2], [24]) is implemented over a wireless backhaul, our framework guarantees approximately twice the delay compared to the corresponding wireline topology.
4) Finally, we present simulation results to demonstrate the effectiveness of our routing algorithms, as well as the low end-to-end delays achieved using bounded-delay schedulers with the Even-Odd framework. The results include the benefits of using our link activation framework for delay-sensitive applications such as Voice-over-IP (VoIP).

The remainder of this paper is organized as follows. Section II lays out the model of the backhaul network and the physical layer for the wireless links that we assume in this work. The Even-Odd link activation framework is outlined in Section III; Section IV covers routing for the backhaul network. An analysis of the end-to-end delay guarantees provided by the Even-Odd framework is presented in Section V. The simulation results are presented in Section VI. Section VII lists some of the related work in the area of scheduling over multihop networks. We conclude with a discussion of future work in Section VIII, and summarize in Section IX.

II. Network Model

We are given a set of access points V, hereby also called nodes, that need to be connected by a wireless backhaul, along with the uplink (to the Internet) and downlink (from the Internet) demands for each access point. Direct, backhaul connections between access points is specified by the set E of directed links. Each pair of access points is connected by a directed link $e \in E$, with a given channel rate $C(e)$. The rate is computed assuming no interference from nearby transmissions, and depends on the distance and the environment between its two end points. One of the access points, $R \in V$ is called the *gateway*; it has a direct, high-speed wired connection to the Internet. The remaining access points must communicate with the gateway via the wireless backhaul to send data to or receive data from the Internet.

In general, a wireless backhaul network may have more than one gateway for higher redundancy and capacity, with each access point being connected to multiple gateways. In this paper, we focus on the problem of scheduling and routing to and from a single gateway. However, our model can be extended to handle multiple gateways (see Section VIII).

A. Physical layer

We assume that the backhaul wireless links between neighboring access points use the WiMAX (802.16d) standard (or some similar longhaul NLOS technology), with links communicating in the time division duplex (TDD) mode. Further, the MAC layer (as in 802.16d) is assumed to schedule data to multiple receivers across timeslots using time division multiple access (TDMA). We also assume the use of subchannelization within a timeslot. Subchannelization allows the available spectrum to be subdivided into multiple orthogonal subchannels (subcarriers in an OFDM system [22] or spreading codes in a CDMA system), so that multiple separate data streams can be received in the same timeslot over separate subchannels at the same or nearby receivers with no interference[1]. We will show in Section VI that subchannelization is useful for reducing end-to-end latency for both uplink and downlink transmissions. Since both endpoints of a link in the wireless backhaul are static, we assume the channel experiences negligible fast fading; however, the channels may experience shadow fading.

B. Interference Model

Access traffic from the users to their respective access points is assumed to be transmitted in a separate frequency band, and does not interfere with the wireless backhaul traffic. We also assume that all backhaul transmissions take place over a contiguous frequency band. Therefore, an access point cannot transmit and receive at the same time even if it uses different subchannels for the transmitted and received signals[2]. We augment previous interference models [21], [30] with subchannelization, such that an access point can simultaneously receive from multiple neighbors without interference as long as the received signals use different subchannels.

Separate links in the backhaul may interfere with each other, depending on the locations of their receiving end points and the antenna

[1] 802.16d currently supports supports subchannelization in units of 1, 2, 4, 8, or 16 OFDM subcarriers in the uplink.

[2] Due to the spectral proximity of the subchannels the transmitted signal will drown out the received signal at an access point.

technology used for transmission and reception. Such pairs of directed links that cannot be simultaneously active in our backhaul network can be precomputed. Therefore, similar to previous work [17], we assume that interfering link pairs are specified as input.

C. The Backhaul Design Problem

There are two types of interference in our backhaul network: (a) self-interference that prevents a single access point from simultaneously transmitting and receiving, or from simultaneously receiving from multiple neighbors on the same subchannel; and (b) cross-link interference, caused by transmissions using the same subchannel over two separate links with distinct receivers that are located close to each other. Together, the routing and scheduling components of our wireless backhaul design must aim to avoid such interference while maximizing throughput and minimizing delay.

The routing component finds routes for each access point to and from the gateway. Given the set of routes, the scheduling component needs to supply two parts: (a) a *link activation scheme* that specifies the set of directional links that are active at each timeslot along with the set of subchannels they use, and (b) a *scheduling policy* that determines the set of packets to be transmitted along an active link(s) at each time slot. Ideally, the latter decision should be made locally at each access point, without requiring any additional communication between access points. In contrast, neighboring access points need to agree on a link activation schedule, and a centralized link activation scheme is acceptable as long as it does not change very often.

III. A Generalized Link Activation Framework

The goal of our link activation framework is to enable local scheduling of packets such that interference is avoided, and the end-to-end delay experienced by all the connections is minimized. We present a simple link activation scheme which we call the *Even-Odd* scheme; this scheme allows directed links to be activated in alternating timeslots. We then describe conditions for admissible traffic that ensure that the backhaul network is not overloaded by the given connections under our Even-Odd scheme.

A. Even-Odd Link Activation

We assume that each node has been labeled as either an *even* or an *odd* node. This labeling is performed by the routing phase (Section IV), which ensures that every route contains alternately labeled nodes. Thus if a link is ever activated in our backhaul network then it must connect an even node and an odd node. We label every link (u, v) as *odd* if u is odd; otherwise it is labeled as an *even* link. Thus, by definition, all valid routes consist of links of alternating labels. Figure 2(a) shows valid uplink and downlink routes for node u in a sample backhaul network.

The Even-Odd scheduling framework uses a simple link activation scheme: each directed link is active every alternate timeslot. All even links are active in the even timeslots, while all odd links are active in the odd timeslots. Figure 2(b,c) illustrates this link activation scheme for a sample backhaul topology. Consider the even node u in Figure 2(a). Every even timeslot, u is transmitting to its neighbors, since links from u to its neighbors are all labelled odd; we say u is now in the *transmitting mode*. Every odd timeslot, u is receiving data from its neighbors; we say u is now in the *receiving mode*. Similarly, odd nodes are in the transmitting mode in every odd timeslot, and in the receiving mode every even timeslot. Thus when an access point joins the backhaul network, the centralized routing phase must assign it a label along with uplink and downlink routes connecting it to the gateway. It must also assign subchannels to its incoming and outgoing links (Section III-B), after which all scheduling decisions can be performed locally at the node.

B. Admissible Traffic and Subchannel Assignment

The traffic to be sent over the backhaul network can be described in terms of *connections*. A connection consists of a source node s, a sink node t, and a description of packet arrival at s. Note that in a backhaul network either the source or the sink is the gateway R, depending on downlink or uplink transmission. For ease of presentation we combine all uplink (downlink) demands from an access point into a single uplink (downlink) connection; thus our backhaul traffic consists of at most $2 \cdot (|V| - 1)$ connections. The routing phase computes the path P_i for each connection i. Let K_i be the number of links along the path P_i. The packet arrival into the backhaul network of each connection is leaky-bucket controlled. For connection i, let ρ_i be the average bit rate and σ_i be the burst size. For any duration of length t, at most $\sigma_i + \rho_i t$ bits are injected at the source node of connection i. Each packet from connection i has at most L_i bits. We study scheduling on *admissible* traffic only, which roughly means the connections do not overload any link or node in the backhaul network. Otherwise, no scheduling policy can possibly achieve bounded delay for all connections.

We first define $E_{in}(v)$ and $E_{out}(v)$ for every node v as:

$$\begin{array}{lcl} E_{in}(v) & = & \{(w,v) | \ (w,v) \in E\} \\ E_{out}(v) & = & \{(v,w) | \ (v,w) \in E\} \end{array} \qquad \forall v \in V.$$

Let $w(e)$ denote the fraction of the total subchannels that are allocated to the directed link e every time it is active. Since a node in the receiving mode receives data from neighbors simultaneously, these transmissions must use disjoint subchannels. Further, a node in transmitting mode uses disjoint subchannels to transmit data to its neighbors. Therefore, we require the following *node constraints*:

$$\left\{ \begin{array}{lcl} \sum_{e \in E_{in}(v)} w(e) & \leq & 1 \\ \sum_{e \in E_{out}(v)} w(e) & \leq & 1. \end{array} \right. \qquad \forall v \in V. \tag{1}$$

Note that if two neighbors are sufficiently far apart so that their receptions from a shared transmitting neighbor do not interfere (for example, with directional transmissions), then the above condition may be relaxed. However, here we conservatively assume they may interfere.

For a directed link e let $F(e)$ be the total bit rate along link e, i.e. $F(e)$ is the total bit rate of all connections whose paths contain link e. Now e can support a data rate of at most $C(e) \cdot w(e)$. Since each link is active every alternate timeslot, that is, for exactly half the time, we require the following *link constraint*:

$$F(e) \quad \leq \quad C(e) \cdot w(e)/2 \qquad \forall e \in E. \tag{2}$$

We say that the traffic is *admissible* if there exists a $w(e)$ for every link e such that node constraint (1) and link constraint (2) are satisfied. We further observe that if the node and link constraints have a feasible solution, then they must also be feasible with $w(e)$ defined for every link e as $w(e) = 2F(e)/C(e)$. This setting of $w(e)$ enforces (2). By rewriting (1) we can redefine admissibility as follows.

Definition 1: If $F(e)$ is the total bit rate of all connections on link e, then the connections are *admissible* in the Even-Odd scheduling framework if

$$\left\{ \begin{array}{lcl} \sum_{e \in E_{in}(v)} \frac{F(e)}{C(e)} & \leq & 1/2 \\ \sum_{e \in E_{out}(v)} \frac{F(e)}{C(e)} & \leq & 1/2. \end{array} \right. \qquad \forall v \in V. \tag{3}$$

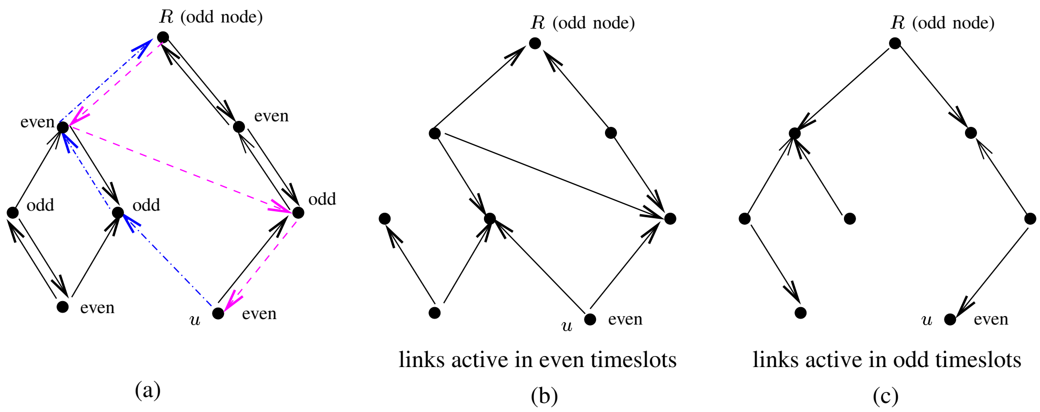


Fig. 2. (a) Possible valid uplink (in dash-dotted blue) and downlink (in dashed purple) routes for a node u: only links connecting odd nodes and even nodes are permitted in routes; (b) links that are activated in even timeslots in the Even-Odd framework; (c) links that are activated in odd timeslots.

For general graph topologies, (3) does not guarantee admissibility, since a subchannel assignment may not be feasible. However, since our routing phase only allows links between even and odd nodes, it effectively creates a bipartite graph. Based on results for coloring bipartite graphs [23], [9], Inequalities (3) ensure that a feasible assignment of subchannels exists and can be easily computed for every link.

C. Tackling Interference

Self-interference, that is, interference between link pairs that share a node, is eliminated in our framework with a combination of our Even-Odd link activation and subchannelization. The Even-Odd scheme prevents a node from simultaneously transmitting and receiving, while subchannelization ensures that simultaneous receptions to (or transmissions from) a single node do not interfere with each other. Therefore the routing component only needs to consider interfering link pairs that do not have a node in common. Such cross-link interference is eliminated by the routing component by ensuring that no two links that interfere will be assigned the same label; thus links with cross interference will not be active simultaneously.

IV. Routing

We require a routing of the connections that satisfies a number of conditions. First, it must be possible to label each node of the graph as either even or odd so that the path for each connection in the routing is a path of alternating even and odd nodes. Secondly, admissibility conditions (3) must be satisfied. Further, to avoid out-of-order delivery of packets, and to guarantee delay bounds (Section V), we assume that each connection must follow a single path; that is, a connection consists of an unsplittable flow. Finally, no two links with the same label (even/odd) that interfere with one another can be on paths in the routing. A *feasible routing* is a routing satisfying all these conditions.

We show that determining if a feasible routing exists is NP-complete via a reduction from the NP-complete problem of finding a 3-Partition. The proof is included in the appendix.

Theorem 4.1: Given the link capacities and the connection bit rates, determining if there is a feasible routing is NP-complete.

A. ILP Formulation

Since finding a feasible routing is NP-complete, we consider the following method for constructing a feasible routing given the link capacities $C(e)$ and connection rates. We wish to determine the largest α such that if all demands are scaled by an α factor, a feasible routing exists. We show that we can formulate this problem as an Integer Linear Program (ILP) as follows.

Let V be the set of nodes, E be the set of directed links, Ω be the set of connections, and for $i \in \Omega$ let f_i be the rate of connection i. $E_{in}(v)$ and $E_{out}(v)$ are as defined in Section III for each node. Also, let $s(i)$ and $t(i)$ denote the source and sink of connection i. For each directed link (v, w) we define a variable $F_i(v, w)$ to represent the bit rate of connection i assigned to travel on the link from v to w. Then $F_i(v, w)$ will be either 0 or $\alpha * f_i$. Let $\mathcal{I}$ be the set of link pairs that interfere with each other; we only need to consider pairs of links that do not share a node in common. For each vertex v we define a binary variable $\pi(v)$ which is meant to represent the parity of v (i.e., $\pi(v) = 0$ if v is chosen to be an even node and $\pi(v) = 1$ if v is chosen to be an odd node). For each connection i and each directed link e we define a binary variable $x_i(e)$ so that $x_i(e) = 1$ represents the fact that e is on the path that connection i gets routed along. In this case, we say that e has been *selected* for i. We define the binary variable $x(e)$ for each link e, which is set to 1 when e is selected by at least one connection. Here parameter $C(e)$ is the capacity of the directed link e. The resulting ILP, which we call the PathsOpt ILP, is shown in Figure 3.

Constraint set (i) just contains the standard conservation of flow constraints for each connection. Constraint set (ii) guarantees that $x(e) = 1$ if link e is selected for at least one connection. Constraint set (iii) guarantees that the bit rate of connection i on links that are not selected for i is 0. Constraint sets (iv) and (v) guarantee that selected links are between nodes with different parities. Constraint set (vi) guarantees that each connection remains on a single path. Constraint sets (vii) and $(viii)$ ensure that any pair of interfering links that are both selected must be assigned different parities. Constraint sets (ix) and (x) ensure that the admissibility conditions (3) are satisfied.

B. Heuristics for Routing

Computing a feasible routing using the ILP formulation described above can sometimes be time consuming even for a relatively small number of nodes. Therefore we now describe three heuristic methods we used to construct routings whose performance will be shown in Section VI.

One heuristic uses a modified Dijkstra's algorithm to compute a shortest path tree of V rooted at R, that is, a tree T such that for

Maximize α;

Subject to:

(i) For $w \in V$, $i \in \Omega$,
$$\sum_{v \in V} (F_i(v,w) - F_i(w,v)) = \begin{cases} -\alpha f_i & \text{if } w = s(i) \\ \alpha f_i & \text{if } w = t(i) \\ 0 & \text{if } w \neq s(i), t(i) \end{cases}$$

(ii) For $e \in E$, $i \in \Omega$, $x_i(e) \leq x(e)$

(iii) For $e \in E$, $i \in \Omega$, $F_i(e) \leq x_i(e) \cdot C(e)/2$

(iv) For $v, w \in V$, $x(v,w) + \pi(v) + \pi(w) \leq 2$

(v) For $v, w \in V$, $\pi(v) + \pi(w) \geq x(v,w)$

(vi) For $v \in V$, $i \in \Omega$, $\sum_{w \in V} x_i(v,w) \leq 1$

(vii) For $((u_1, v_1), (u_2, v_2)) \in \mathcal{I}$,
$$\pi(u_1) + \pi(u_2) \geq x(u_1, v_1) + x(u_2, v_2) - 1$$

$(viii)$ For $((u_1, v_1), (u_2, v_2)) \in \mathcal{I}$,
$$\pi(u_1) + \pi(u_2) \leq 3 - x(u_1, v_1) - x(u_2, v_2)$$

(ix) For $v \in V$, $\sum_{e \in E_{in}(v), i \in \Omega} \frac{F_i(e)}{C(e)} \leq \frac{1}{2}$

(x) For $v \in V$, $\sum_{e \in E_{out}(v), i \in \Omega} \frac{F_i(e)}{C(e)} \leq \frac{1}{2}$

Fig. 3. The PathsOpt ILP formulation.

any node $u \in V$, the path in T between R and u is a shortest path [10]. We use $1/C(e)$ as the distance along link e. Dijkstra's algorithm iteratively constructs a shortest path tree from the root R. At any stage of the algorithm there is a partial tree T' and we find the shortest edge from T' to some node not in T'. However we modify this step to only consider such edges so that directed versions of the edge will not interfere with any directed version of any edge already in T'. The routing of each connection is the unique path in the tree from the source to sink of that connection. Note that the even/odd labeling of a node is taken as the parity of the depth of the node in the final tree.

The next two heuristics are based on the following. Suppose we are given a complete directed graph $G = (V, E)$ and a set of connections Ω between various node pairs as well as a routing P of these connections. For each directed link in E, let $F_P(e)$ be the resulting total bit rate routed over e. Then for each $v \in V$, define $L_P(v)$, the *node load* of v with respect to P, as $L_P(v) = \max(\sum_{e \in E_{in}(v)} F_P(e)/C(e), \sum_{e \in E_{out}(v)} F_P(e)/C(e))$.

We now describe two heuristics based on the node load. Each heuristic adds connections one at a time in some greedy manner. We assume that for each node v there is a unit rate connection from v to R and a unit rate connection from R to v.

The first heuristic is as follows. Order the nodes by increasing length of their shortest path to R where as before, link lengths are taken to be $1/C(e)$. Start with all nodes unlabeled, all connections unrouted and so all node loads are 0. Then for each node v in order do the following. First route the connection from v to R along the path using no directed links between nodes of the same parities, and whose resulting maximum node load is minimum among all possible paths. When choosing the route we also ensure that no two interfering link pairs in $\mathcal{I}$ get the same label. Such a path is easily computed using a simple modification of Dijkstra's shortest path algorithm. Next, update the node loads accordingly. Finally, in a similar manner, route the connection from R to v. We call this method "MinMax+SP".

The second heuristic is similar to the first except that the order in which connections are routed is not fixed from the start. Inductively, suppose some number of connections have been routed and hence they induce node loads. Given these node loads, for each unrouted connection i find the routing P_i of i that minimizes the maximum node load along P_i and let m_i be the resulting maximum node load along P_i. Then route the connection that minimizes this quantity m_i and update the node loads accordingly. Continue until all connections are routed. We call this method "MinMax".

Note that due to interfere constraints, a feasible routing may not always exist. Further, even when a feasible routing exists (and can be found using the ILP) the heuristics may not find one.

V. Scheduling Policy and Delay Analysis

Now that we have a set of admissible connections each with a given route, we are ready to describe our scheduling policy. We treat all links as directed. Each link has a buffer at the head of the link. Packets queue in these buffers when waiting to be transmitted. The scheduling policy determines the order in which packets leave each buffer. We assume that the scheduler makes scheduling decisions continuously. Whenever a link finishes servicing a packet, the scheduler decides which packet to service next. A link with capacity c takes time L/c to service the entire packet of size L. A packet is eligible for scheduling if the entire packet has arrived in the buffer. If a link becomes inactive before an entire packet is serviced, the remainder of the packet waits in the queue and is serviced once the link is active again. We use τ to denote the duration of a timeslot. We also assume zero propagation delay on each link since it is negligibly small.

Recall under our Even-Odd link activation scheme, odd (resp. even) links are active during odd (resp. even) time slots, where odd (resp. even) links are defined to be outgoing links of odd (resp. even) nodes. Thus each link is active half the time and idle during the other half of the time and so we refer to this type of system as a *half-idle system*. We define the *effective capacity* $c(e)$ for each link e to be $C(e) \cdot w(e)$, where $w(e)$ is defined in (2).

Many scheduling policies are studied for the traditional wireline setting in which links are active all the time. In order to apply these policies to our Even-Odd framework we introduce an *imaginary* wireline network N_I with the same topology and effective link capacities as the wireless network N. We apply *any* scheduling policy to the imaginary system. The times at which packets leave a buffer in the imaginary system dictate the times at which packets leave a buffer in our real wireless system. At a high level, our mapping works as follows. The arrivals during two consecutive timeslots to the real system are mapped to one timeslot to the imaginary system. In particular, the arrivals during timeslots $2t$ and $2t+1$ for the real system are mapped to timeslot t for the imaginary system. This "squeezes" the half-idle wireless system into a fully-loaded imaginary system. We then apply a scheduling policy to the latter. Suppose a packet leaves a link in N_I during a timeslot t', then the packet leaves the same link in N during timeslot $2t'+2$ or $2t'+3$ depending on whether or not the link is active during an even timeslot. We show that this actually "spreads" out the departure times of the packets in a feasible way, i.e. the links in the real system can indeed service all the packets according to these departure times. Therefore, as long as the scheduling policy has an end-to-end delay guarantee in the imaginary system, it also has a delay guarantee in the real system which roughly doubles the bound.

A. Construction

We begin with a description of the imaginary system. The imaginary system N_I consists of the real network N with a *propagation delay* of τ added to every odd link and zero propagation delay added to every even link. See Figure 4. Recall τ is the duration of a timeslot. If a link has propagation delay g, then after the link completely services a packet p it takes time g for p to traverse the link. Once the link completes servicing p it can start servicing the next packet without waiting for the propagation time of g. (Recall also our real system has zero propagation delay on all links.) Each link e in N_I is assigned a link capacity equal to its effective capacity $c(e)$ in the real network N.

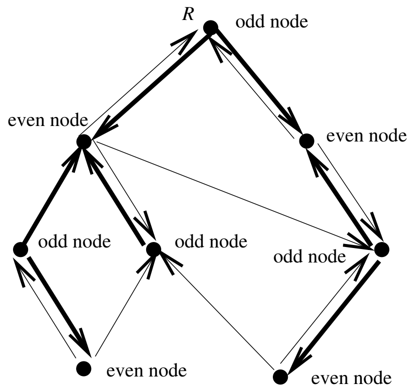


Fig. 4. Imaginary network N_I for the backhaul network from Figure 2. Thick links are odd links with propagation delay τ and thin links are even links with propagation delay 0.

The clock of the imaginary system relative to the clock of the real system is depicted in Figure 5. The imaginary clock starts at real time τ and is turned off every other timeslot of duration τ. More precisely, during real time intervals $[\tau, 2\tau)$, $[3\tau, 4\tau)$, etc. the imaginary clock is turned on and during real time intervals $[0, \tau)$, $[2\tau, 3\tau)$, etc. the imaginary clock is turned off.

To complete the definition of the imaginary system let us define the connections. Each connection in N_I follows a path identical to that in N. Suppose a packet arrives at its source in N at real time $a = n \cdot \tau + \alpha$ where $\alpha = a (\text{mod } \tau)$. Let the packet arrive at N_I at imaginary time

$$a_I = \lfloor n/2 \rfloor \cdot \tau + \alpha. \tag{4}$$

This definition of a_I is depicted by the two downward arrows in Figure 5. Note that at any imaginary time N_I all packets with arrival times $a_I \leq t_I$ have arrived in the real system already. Therefore, we can use any existing scheduling policy such as WFQ, EDF to schedule the imaginary system. Let us call this policy S_I.

We now define a scheduling policy S for the real system using the scheduling decisions made by S_I for the imaginary system. Suppose a packet starts to leave the buffer at a link e of N_I at time $d_I = n \cdot \tau + \delta$ where $\delta = d_I (\text{mod } \tau)$, then this packet starts to leave the buffer of link e in N at time d where

$$d = \begin{cases} (2n+2) \cdot \tau + \delta & \text{if } e \text{ active at even timeslots} \\ (2n+3) \cdot \tau + \delta & \text{if } e \text{ active at odd timeslots} \end{cases} \tag{5}$$

This definition of d is depicted by the two upward arrows in Figure 5. As said before, if the link becomes inactive before an entire packet is serviced, the remainder of the packet stays in the buffer and is serviced once the link becomes active again. This defines the scheduling policy S for our real system.

B. Analysis

We first show that S is well defined for our real system by verifying the following.

1) Each packet departs from a buffer only when the link is active.
2) Each packet starts to depart from a buffer after the complete packet has arrived.
3) Each link services one packet at a time.
4) No packet misses its departure time. A packet *misses* its departure time d if the packet is assigned its departure time at time $t > d$.

We use the following notation. The departure time $d(p, e)$ (resp. $d_I(p, e)$) is the moment when packet p starts to leave the buffer along link e in N (resp. in N_I). The arrival time $a(p, e)$ (resp. $a_I(p, e)$) is the moment when packet p has entirely arrived at the buffer along link e in N (resp. in N_I).

Lemma 5.1: The resulting scheduling policy S for the real system is well defined.

Proof: Item 1 holds trivially from the definition of departure time d in (5).

To show item 2 we verify that $a(p, e) \leq d(p, e)$ for every packet p at every link e along the path in N. Suppose e is the first link along the path. Since S_I is a well-defined schedule, $d_I(p, e) > a_I(p, e)$. Suppose $d_I(p, e) = n \cdot \tau + \delta$, then we have $d(p, e) \geq (2n+2) \cdot \tau$ by definition (5) and $a(p, e) < (2n+2) \cdot \tau$ by definition (4). Therefore, $d(p, e) > a(p, e)$ if e is the first link.

The case where e is not the first link requires tighter analysis. Let e' be the link immediately before e along the path in N. There are two cases to consider depending on whether e' is an odd link or an even link.

- If e' is an even link, then e' has zero propagation delay in N_I. Since S_I is a valid schedule, we have,

$$d_I(p, e) \geq a_I(p, e) = d_I(p, e') + L/c(e') \tag{6}$$

where L is the packet size of p. Suppose $d_I(p, e') = n' \cdot \tau + \delta'$ and $d_I(p, e) = n \cdot \tau + \delta$. This implies $n \geq n' + z - 1$ where $z = \lceil \frac{L/c(e')}{\tau} \rceil$. Note that a packet of size L is served in either z timeslots in which case the packet experiences $z - 1$ entire idle timeslots; or the packet is served in $z + 1$ active timeslots in which case it experiences z entire idle timeslot. An even link e' is active at even timeslots, which implies $d(p, e') = (2n' + 2) \cdot \tau + \delta'$ and $d(p, e) = (2n + 3) \cdot \tau + \delta$. Hence,

$$\begin{aligned} d(p, e) &= (n+2) \cdot \tau + (n \cdot \tau + \delta) + \tau \\ &\geq (n' + z + 1) \cdot \tau + (d_I(p, e') + L/c(e')) + \tau \\ &= d(p, e') + L/c(e') + z\tau. \end{aligned}$$

Note that $a(p, e) = d(p, e') + L/c(e') + (z-1)\tau$ if e' services p within z active timeslots, and $a(p, e) = d(p, e') + L/c(e') + z\tau$ if e' services p within $z+1$ active timeslots. As a result, $d(p, e) \geq a(p, e)$.

- If e' is an odd link, then e' has propagation delay of τ in N_I. We have,

$$d_I(p, e) \geq a_I(p, e) = d_I(p, e') + L/c(e') + \tau \tag{7}$$

Since e' is active at odd timeslots, $d(p, e') = (2n'+3) \cdot \tau + \delta'$ and $d(p, e) = (2n+2) \cdot \tau + \delta$. Since $\delta < \tau$, (7) implies $n \geq n' + z$. Hence,

$$\begin{aligned} d(p, e) &= (n+2) \cdot \tau + (n \cdot \tau + \delta) \\ &\geq (n' + 2 + z) \cdot \tau + (d_I(p, e') + L/c(e') + \tau) \\ &= d(p, e') + L/c(e') + z\tau. \end{aligned}$$

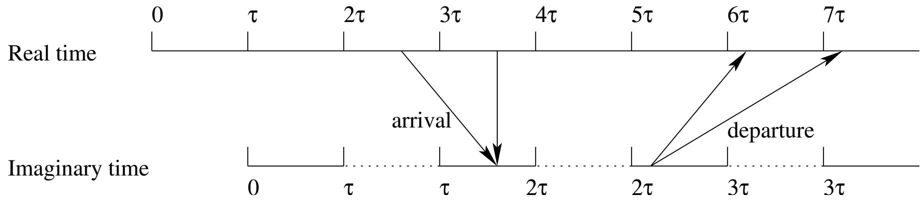


Fig. 5. The clocks for the real and imaginary system. The dotted intervals for the imaginary time line is when the imaginary clock is turned off. The arrows indicate the mapping of the arrival and departure times of the two systems.

As argued before, $a(p,e) \leq d(p,e') + L/c(e') + z\tau$. Hence, $d(p,e) \geq a(p,e)$.

We now consider item 3. We inductively assume that before servicing packet p, link e in N services packets one at a time under the schedule S. Let p' be any packet whose departure time is after that of p, i.e. $d(p',e) > d(p,e)$. Let their corresponding departure times under S_I in the imaginary system be $d_I(p,e) = n \cdot \tau + \delta$ and $d_I(p',e) = n' \cdot \tau + \delta'$. Since $d(p',e) > d(p,e)$ and $\delta < \tau$, we have $d_I(p',e) > d_I(p,e)$. Since e services one packet at a time under S_I, $d_I(p',e) \geq d_I(p,e) + L/c(e)$. There are two cases to consider depending on whether e services p in z or $z+1$ timeslots under S_I.

- If e services p within z timeslots under S_I, then e also services p within z active timeslots under S. In this case, $n' \geq n+z-1$. If e is active during even timeslots, then $d_I(p',e) \geq d_I(p,e) + L/c(e)$ implies

$$\begin{aligned} d(p',e) &= (2n'+2)\cdot\tau + \delta' \\ &\geq (n'+2)\cdot\tau + d_I(p,e) + L/c(e) \\ &\geq (n+z+1)\cdot\tau + d_I(p,e) + L/c(e) \\ &= d(p,e) + L/c(e) + (z-1)\tau. \end{aligned}$$

  If e is active during odd timeslots, similar analysis also shows $d(p',e) \geq d(p,e) + L/c(e) + (z-1)\tau$.
- If e services p in $z+1$ timeslots under S_I, then e also services p in $z+1$ active timeslots under S. In this case, $n' \geq n+z$. If e is active during even timeslots we have,

$$\begin{aligned} d(p',e) &= (2n'+2)\cdot\tau + \delta' \\ &\geq (n'+2)\cdot\tau + d_I(p,e) + L/c(e) \\ &\geq (n+z+2)\cdot\tau + d_I(p,e) + L/c(e) \\ &= d(p,e) + L/c(e) + z\tau. \end{aligned}$$

  If e is active during odd timeslots, similar analysis also shows $d(p',e) \geq d(p,e) + L/c(e) + z\tau$.

Therefore, in both cases e services p' after it completes its service of p.

Item 4 is best illustrated by the two upward arrows in Figure 5. Packet p is assigned departure time $d(p,e)$ at a time no later than when p starts to depart from e in the imaginary system, which is the imaginary time $d_I(p,e)$. The real time that corresponds to $d_I(p,e) = n\cdot\tau + \delta$ is at most $(2n+2)\cdot\tau$. By definition (5), $d(p,e)$ is at least $(2n+2)\cdot\tau$. Hence, item 4 holds. ■

Before bounding the end-to-end delay in the real system, we remark that the arrival to our imaginary system does not overload any links. Roughly speaking, packet arrival to the imaginary system doubles that of the real system. Since the effective capacity $c(e) = C(e)\cdot w(e)$ is half utilized by (2) in the real system, the arrival rate to link e is lower than $c(e)$ in the imaginary system in the long run. Therefore,

Lemma 5.2: The packet arrival to the imaginary system does not overload any link.

Let us state the relationship between the end-to-end delays that a packet experiences in the two systems.

Theorem 5.3: For any scheduling policy and any packet p, we have

$$D(p) \leq 2D_I(p) + 4\tau,$$

where $D(p)$ and $D_I(p)$ are the actual end-to-end delays that p experiences in the real and imaginary systems respectively.

Proof: Let e_1 be the first link and e_k be the last link along the path for packet p, and let L be the packet size of p. We have

$$\begin{aligned} D(p) &= d(p,e_k) + L/c(e_k) - a(p,e_1), \\ D_I(p) &= d_I(p,e_k) + L/c(e_k) - a_I(p,e_1). \end{aligned}$$

Let $d_I(p,e_k) = n_d\cdot\tau + \delta$ and $a_I(p,e_1) = n_a\cdot\tau + \alpha$. By (5) and (4), we have

$$\begin{aligned} D(p) &\leq (2n_d+3)\cdot\tau + \delta + L/c(e_k) - (2n_a\cdot\tau + \alpha) \\ &\leq 2(d_I(p,e_k) + L/c(e_k) - a_I(p,e_1)) + 4\tau \\ &\leq 2D_I(p) + 4\tau. \end{aligned}$$

■

The above theorem immediately implies the following.

Corollary 5.4: For any scheduling policy and connection i, we have

$$\Delta(i) \leq 2\Delta_I(i) + 4\tau,$$

where $\Delta(i)$ and $\Delta_I(i)$ are the worst-case end-to-end delay bounds for connection i in the real and imaginary systems respectively.

To determine how end-to-end delays are affected by our Even-Odd scheme, we need to relate, $\Delta(i)$, the delay bound in our wireless setup where links have alternating active periods to the delay bound in the traditional wireline setup where links are active all the time and have no propagation delays. We denote the latter by $\Delta_I^0(i)$ since it is equivalent to the delay in the imaginary system if we did not add propagation delay τ to every other level of the links. We would like to show

$$\Delta_I(i) \leq \Delta_I^0(i) + \lceil K_i/2 \rceil \cdot \tau. \qquad (8)$$

However, the above relationship does not hold for all scheduling policies indiscriminately. For example, if the scheduling policy makes decisions based on the delay a packet has experienced so far, the link propagation delay could alter the priority among the packets. Fortunately, (8) does hold for two well-studied scheduling policies WFQ and CEDF. More strongly, we have the following for any arbitrary network.

Lemma 5.5: Consider the scheduling policies of WFQ and CEDF in an arbitrary network N where links are active all the time. If connection i has a total propagation delay of $g(i)$, then the end-to-end delay is bounded by $\Delta_N^0(i) + g(i)$ where $\Delta_N^0(i)$ is the bound when there is no propagation delay.

We omit the detailed proof of the above lemma for space consideration. At a high level, the proof for WFQ uses the concept of rate-latency server (see for example [4]). The key is the following: $i)$ a WFQ server is a rate-latency server; $ii)$ propagation delay g on a server shifts the latency by an amount of g; and $iii)$ the concatenation of several rate-latency servers is a rate-latency server. For WFQ, $\Delta_N^0(i) = \frac{\sigma_i}{\rho_i} + (K_i - 1) \cdot \frac{L_i}{\rho_i} + \sum_{1 \le k \le K_i} \frac{L_{\max}}{c_k}$ (See [28].) To incorporate non-zero propagation delay to the CEDF protocol we redefine the deadline at each server by adding the amount equal to the propagation delay of the server. For CEDF, $\Delta_N^0(i) = \frac{\sigma_i + 4L_i/\varepsilon}{\rho_i} + \sum_{1 \le k \le K_i} \frac{L_{\max}}{c_k} \log(\cdot)$, with high probability. (See [2], [24].)

Combining Corollary 5.4 and Lemma 5.5 we have the following.

Theorem 5.6: Under WFQ and CEDF, the end-to-end delay of each connection i is bounded by

$$\Delta(i) \le 2\Delta_I^0(i) + (K_i + 1)\tau + 4\tau.$$

We remark that given the Even-Odd framework, $K_i\tau$ is a lower bound on the delay for connection i. This is because two consecutive links are active in alternating timeslots and therefore no packet can traverse two consecutive links in a single timeslot. The factor of 2 in the term $2\Delta_I^0(i)$ comes naturally as a link in the real wireless system has the same capacity as in the imaginary wireline system but is only activated half of the time. We also remark that although Theorem 5.6 does not hold for all scheduling policies, Theorem 5.3 and Corollary 5.4 do.

VI. Simulation Results

We now describe the simulations we conducted to test our routing algorithms (Section VI-B) and our scheduling framework (Section VI-C).

A. Methodology

We generate random locations for the access points in a given area, and select one access point at random to act as the gateway. Since 802.16d supports a tree-based topology for centralized scheduling over meshes [16], we modified our PathsOpt ILP (Section IV) to create routes on a single tree. The combination of all uplink and downlink routes now form the optimal tree topology rooted at the gateway. All nodes are assigned equal uplink and downlink demands in our experiments. The modified ILP is then solved to construct the backhaul tree topology with the maximum total throughput (sum of all demands); this is feasible only for inputs with a limited number of access points. Here we define the *capacity* of a backhaul network as the total per-node connection rate (uplink + downlink) that is admissible within our Even-Odd framework. Thus if every node can send f bps in the uplink direction and receive f bps in the downlink direction (as in our test cases), the capacity of the wireless backhaul is $2f$. We also modified the node load-based routing heuristics described in Section IV to generate interference-free routes over a tree topology (the shortest path heuristic already generates a tree).

To evaluate the Even-Odd activation framework we have implemented a simulator for the backhaul network that takes as input the routes (tree topology in our test cases), the capacities for each link, and the scheduling policy at the access points. The simulator computes the capacity of the backhaul network, along with the subchannel assignment for each directional link to achieve this capacity. It then simulates the backhaul network for several timeslots and measures the resulting end-to-end delays experienced by the packets in the uplink and downlink directions.

We make a few minor, practical modifications to the Even-Odd framework so that it can be easily implemented in a real wireless backhaul. While our model allows access points to make scheduling decisions continuously on a packet-by-packet basis, our simulator restricts the scheduling decisions as well as the packet arrivals to occur on timeslot boundaries. Thus, if an access point can send n packets over a link in one timeslot, it selects those n packets at the start of every timeslot. The scheduling policy at each node has been implemented using both of the following methods: (a) simulating the imaginary wireline network where every links is activated every alternate timeslot (Section V), which dictates all the scheduling decisions in the real wireless network, or (b) simply implementing the specified scheduling policy at each node in the real network by activating each link every alternate timeslot, with no imaginary network. Method (b) is a more practical approximation than Method (a). We saw very little difference in the delay results using the two methods, and we only report results using the more practical Method (b) for all scheduling policies.

Physical layer. To compute the link capacity between two access points, we use the fixed wireless pathloss model [11] proposed as part of the IEEE 802.16 task group. The terrain is assumed to be of type “A” (hilly/moderate-to-heavy tree density); wireless backhaul links between access points experience shadow fading [12]. All access point antennas are assumed to be at a height of 6 meters (20ft), with a transmit power of 36dBm (including the transmitter gain). The receive antennas have a gain of 18dBi. Assuming a 20MHz bandwidth and no interference across links, the pathloss allows us to compute the resulting signal-to-noise ratio (SNR) at the receiver. We then use the results of link-level simulations (which account for coding and modulation penalties) to translate this SNR into a bit rate for links between each pair of access points. Timeslots have a length of $\tau = 1$ millisecond, and all packet sizes in the system are set to 1Kbit.

We assume directional antennae at the transmitters as well as the receivers to minimize interference between different links in the backhaul. This is possible either by installing an adaptive antenna array for the backhaul transmissions at each access point, or with the use of fixed directional antennae with one antenna pointing to each neighbor in the backhaul topology. We assume a simple beam forming model for directional transmissions, ignoring side lobes similar to previous work [3]. Each transmit (receive) beam is 20 degrees wide, centered on the receiver (transmitter). We also assume that there is no interference at angles beyond this beamwidth, or at distances beyond 5% of the transmitter-receiver link length (see Figure 6). Directional transmissions over two different links that use the same subchannels will interfere at the two receiving access points if the access points are located close to each other, *and* if they both receive the signal from a similar direction. Cross-link interference is computed accordingly as input for the routing phase.

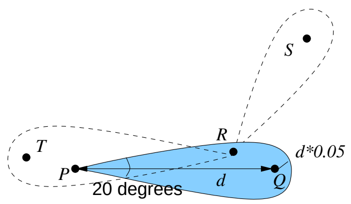


Fig. 6. Region of interference (shaded region) when P transmits to Q. The dotted regions denote the receive beams for access point R. Thus when P transmits to Q, R sees interference if it were to receive from T on the same subchannel, but not if it were receiving from S.

B. Routing

We generated seven input scenarios, which we call S1–S7. Each input has 15 access points located at random within a 5km x 5km area. Recall that all traffic must pass through the single root (gateway). Since with a 20MHz bandwidth we assume a maximum link capacity of 75Mbps, in the best scenario (where all links in the backhaul network are of high capacity), we can expect to have a tree of capacity at most 75/15 = 5Mbps. In reality, however, shadow fades and pathloss reduce the link capacities to some extent.

Figure 7 compares the capacity of the optimal backhaul tree constructed using the ILP, to the capacity of the tree obtained using the heuristics. Since the heuristic solutions run very fast, we can run all three and select the topology that results in the highest capacity. As shown in the figure, the capacity of such a heuristic-based tree was within 75–100% of the optimal.

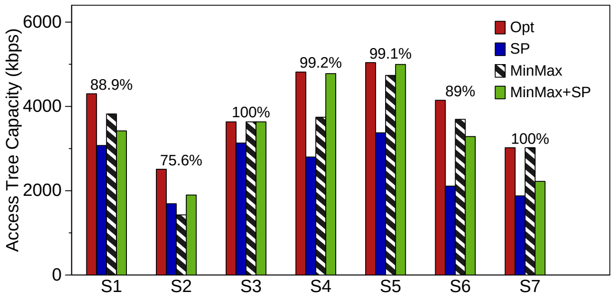


Fig. 7. Total backhaul network capacity for seven sample inputs using four different routing algorithms. "Opt" is the optimal solution from the optimal ILP; "SP" is the shortest path heuristic, "MinMax" is the heuristic that tries to minimize the maximum node load, and "MinMax+SP" is MinMax using the node ordering from the SP algorithm. The number at the top of each group denotes how close the best of the three heuristics comes to the optimal solution.

C. Scheduling for Low Delays

Henceforth we refer to the best heuristic-based backhaul trees for the input scenarios S1–S7 from Figure 7 as the topologies T1–T7. We now examine the delays experienced by the packets when different scheduling policies and link activation frameworks are used. To evaluate the effectiveness of the Even-Odd framework, we compare it with a more traditional link activation framework which we call *Periodic*. In the Periodic framework, a link uses all the available subchannels when it is activated, that is, there is no subchannelization. A periodic schedule is generated for each link, with a period of Γ timeslots; the period length Γ is chosen as the minimum length that can generate a schedule with the required capacity for each tree. The number of times a link e is activated during a period of Γ slots is $\lceil \Gamma \cdot F(e)/C(e) \rceil$ where $F(e)$ is the total bit rate on the link e and $C(e)$ is its capacity. For each link we use several heuristics to spread out the active slots as evenly as possible over the period of Γ timeslots. The Periodic framework appears to be one of the best link activation strategies in the absence of subchannelization, and fits well into the link activation frameworks described in previous work [21].

Within both the Periodic and the Even-Odd frameworks, we have implemented the following scheduling policies at the access points: first-in-first-out (FIFO), oldest first (OF), Weighted Fair Queuing (WFQ), and Coordinated EDF (CEDF). In the Periodic framework we henceforth refer to them as P-FIFO, P-OF, P-WFQ, and P-CEDF, respectively, while in the Even-Odd framework they will be called E-FIFO, E-OF, E-WFQ, and E-CEDF, respectively. With OF, an access point selects those packets to send over an active link that are the oldest in the system amongst all the packets that currently need to be routed over that link.

Subchannelization Penalty. Since each link needs to utilize an integral number of subchannels in the Even-Odd framework, subchannelization may cause the maximum capacity determined by the constraints in Section III-B to be reduced. The fewer the number of subchannels, the larger this reduction. Figure 8 shows this reduction in capacity in the Even-Odd framework as a function of the total number of subchannels for tree topologies T1–T7. With 32 or more subchannels, we are able to attain over 95% of the maximum capacity.

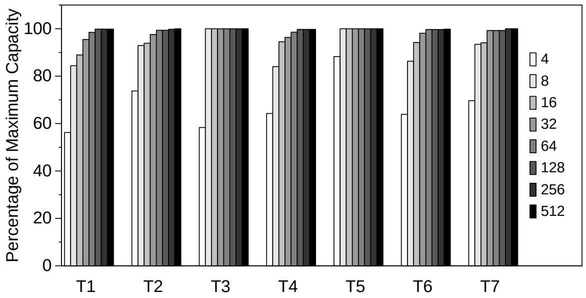
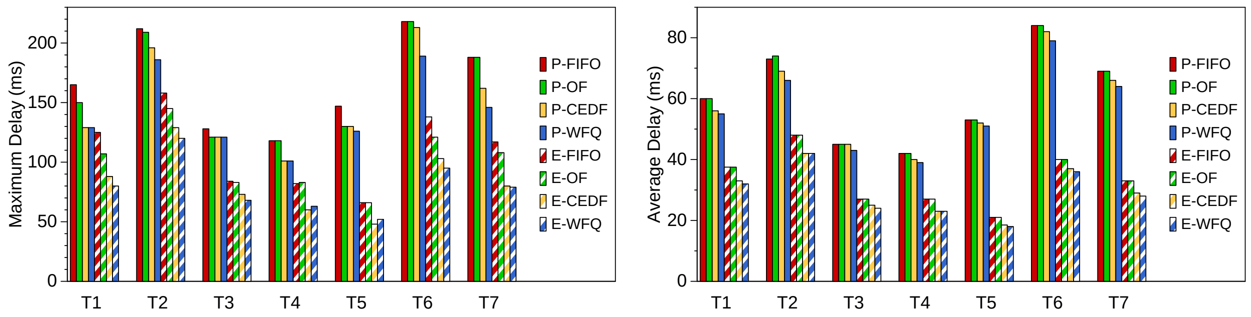


Fig. 8. Total backhaul network capacity for seven sample inputs for different numbers of subchannels. Capacity is shown as a percentage of the maximum capacity without subchannelization.

In the remainder of the experiments we assume there are a total of 64 subchannels, and use the accordingly reduced capacity as input to evaluate delay in all the schedulers for both the Periodic and Even-Odd frameworks. The length of the periodic schedule is set to the minimum length possible (typically 100-400 msec) to support this capacity for each tree topology.

End-to-end Delays. We now measure the end-to-end packet delays when each node in the backhaul network has an equal-rate connection in the uplink and downlink (of combined rates as shown in Figure 8). The arrival process for each connection uses a burst size σ_i of at most 100 packets. Figure 9 shows the average and maximum delays incurred by packets for each backhaul topology T1–T7. As we can see from the figure, the subchannelization and alternate link activation of the Even-Odd framework provides a significant improvement in end-to-end delays. Further, within either framework, the bounded delay schedulers (WFQ and CEDF) result in lower maximum delays (up to 45% lower compared to, say, FIFO).

Burstiness of Input Traffic. Next, we examine the end-to-end delays with variations in the amount of burstiness in packet arrivals. For ease of presentation, we compare one representative scheduler from each framework, namely, P-WFQ and E-WFQ, for the backhaul tree T1. Figure 10 shows how the delays for the P-WFQ and the E-WFQ schedulers increase as the burst size is increased: larger burst sizes represent more bursty traffic. For both schedulers, the delay increases as burst size increases, although E-WFQ consistently provides lower delays compared to P-WFQ. In particular, for non-bursty (uniform) traffic E-WFQ can provide average (maximum) delays of under 5ms (20ms), while the P-WFQ average (maximum) delay remains above 40ms (150ms). We also plot here the upper bound on end-to-end packet delay for E-WFQ computed using Theorem 5.6. We see that the measured maximum E-WFQ delay is close to the analytical bound for traffic with low burstiness; this indicates that the analytical delay



(a) (b)

Fig. 9. (a) Maximum and (b) average packet delays over all uplink and downlink connections using different link activation schemes and scheduling policies. The striped bars represent delays in the Even-Odd framework.

bound may be useful when designing backhaul networks or admitting delay-sensitive connections.

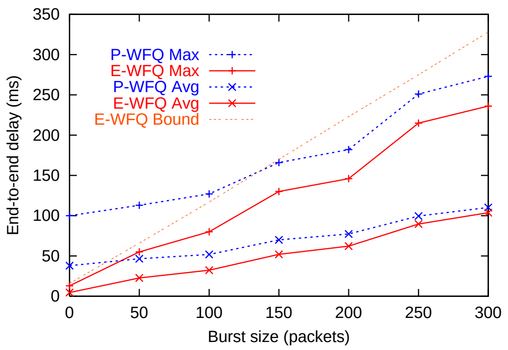


Fig. 10. Maximum and average end-to-end packet delays as the burst size of the input traffic for all connections is increased.

Effect of Traffic Load. We define load as the rate of each connection as a percentage of the maximum admissible rate. Figure 11 shows the effect of increasing load on end-to-end delays for topology T1. Here the link activation schedule is determined according to the maximum admissible rates, so at low rates far fewer packets are sent over each link compared to its maximum capacity. Therefore, at very low loads, the difference between the schedulers is less pronounced, although the Even-Odd framework consistently yields lower delays.

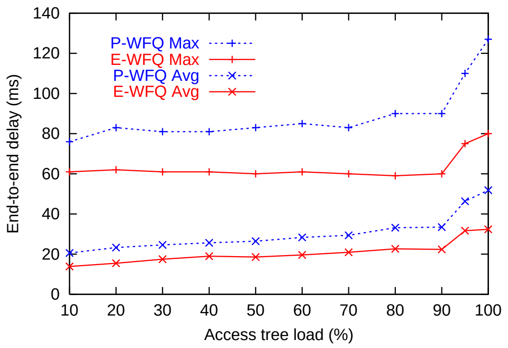


Fig. 11. Maximum and average end-to-end packet delays as the data rates of all the connections is increased, shown here as a percentage of the maximum admissible connection rate.

Number of Access Points. Next we examine the effect of the number

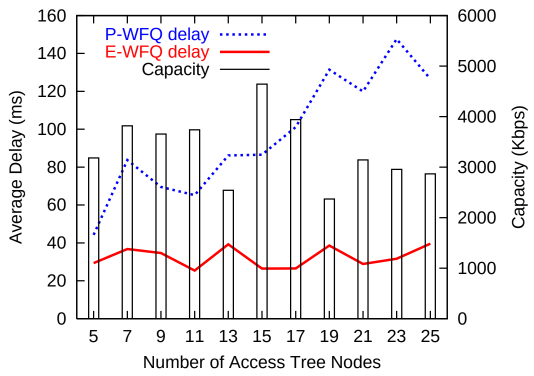


Fig. 12. Total backhaul network capacity (as a histogram) and the average packet delays as more nodes are added to a 5km x 5km space. The backhaul tree is reconstructed at each stage.

of nodes in a backhaul network on the network's capacity and the end-to-end delays. Starting with 5 nodes, we incrementally add nodes in a 5km x 5km space until we have a total of 25 nodes. At each step, we constructed from scratch the best heuristic-based backhaul tree topology from the given set of nodes. Note that when a node is added, the capacity of the resulting backhaul network may rise or fall depending on the location of the node. Figure 12 shows the capacity and the average end-to-end delay at each step as we added nodes. We found that the Even-Odd activation framework remains significantly more stable in its delays as the number of nodes increases.

Delay-sensitive applications. Finally, we examine the delays experienced by a delay-sensitive application such as voice-over-IP (VoIP). For each topology T1–T7, we added a VoIP stream in the uplink as well as downlink directions at each node. Each VoIP stream consists of a 1 Kbit packet sent every 20ms (to simulate a 32Kbps voice coder with additional packet headers); the rates for the remainder of the connections were decreased accordingly. We added explicit, fixed deadlines of 20ms for each VoIP packet, and utilized the coordinated EDF (CEDF) scheduler to assign deadlines (and hence priorities) to the remaining packets. Figure 13 shows the maximum and average delays experienced by the VoIP packets over the backhaul. The Even-Odd activation framework with subchannelization provides significantly lower delays; its average delays of 4–9 ms are well within the acceptable range for VoIP traffic over the wireless backhaul.

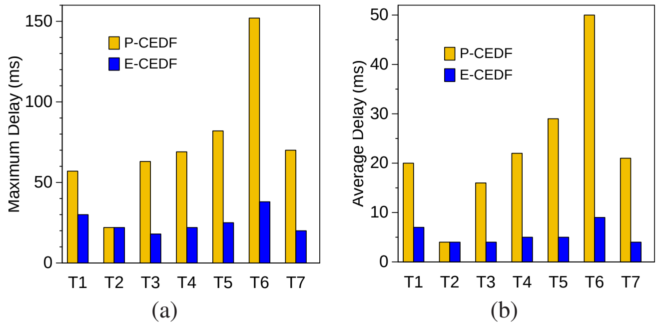


Fig. 13. (a) Maximum and (b) average packet delays over uplink and downlink VoIP packets using different link activation schemes and the CEDF scheduling policy.

VII. Related Work

Several researchers have worked on designing interference-free schedules for multihop wireless networks [30], [7], [18]. The main goal is to increase the throughput of the network by optimizing the length of the computed periodic schedule. We show in Section VI that our scheduling framework allows for better delays in practice compared to the approach of building periodic schedules. Multihopping through the use of relays was found to increase throughput in cellular networks [31]. Routing and schedulability analysis of specific traffic demands between pairs of nodes on a general wireless mesh architecture has been shown to be NP-complete [17], [21]. Our backhaul traffic demand is more restricted (strictly uplink or downlink) in comparison. We show that constructing the optimal routes for such traffic is NP-complete; however, given a backhaul tree topology, finding the maximum throughput and a good delay-sensitive schedule for that throughput is tractable.

Fair queuing has been applied to multihop wireless networks with the goal of balancing fairness with throughput and spatial reuse [26], or applying max-min fairness to an ad-hoc network [15]. Kanodia et al [19] develop a distributed scheduling model for 802.11-based multihop networks, which piggybacks a priority index in RTS/CTS messages to facilitate communication and coordination between nearby nodes. They provide an analysis of probabilities for meeting deadlines set by a coordinated EDF scheduler. Luo et al [25] also present distributed multihop fair queuing algorithms that piggyback flow priorities in control messages; they provide flow fairness guarantees for their algorithms. In contrast, we provide a general link activation framework that mitigates interference given a TDMA physical layer, so that any wireline scheduler can be applied locally at the nodes of a wireless multihop backhaul; the framework efficiently maps end-to-end wireline delay guarantees to the wireless setting.

With the increase of wireless hotspots, wireless backhaul deployments are on the rise, both in the research [5], [20] and commercial communities. Initial experiences indicate that interference and multipath effects can limit performance [1] and fairness [13]. The OFDM physical layer of 802.16d is expected to be more robust to multipath interference for long haul links. Directional antennas, which have become fairly affordable [29], can help reduce cross-link interference and significantly improve delay and throughput [29], [8], [3]. Further, we expect the scheduled, time division multiplexed 802.16d MAC layer to mitigate several of the interference and fairness issues through careful scheduling.

On the commercial front, several companies have been providing single-hop wireless backhauls using proprietary wireless technologies; several more are expected to provide WiMAX-based networks. Multihop, mesh based technologies are now being commercially deployed by companies such as BelAir, Motorola (MeshNetworks), FireTide, Tropos, and SkyPilot; the majority of the mesh offerings provide backhaul for 802.11 hotspots. Some offer bandwidth guarantees similar to DSL or cable providers; however we are not aware of any vendor providing end-to-end delay guarantees on the traffic they admit into their meshes.

VIII. Discussion and Future Work

Suppose instead of a single gateway as described in this paper, we allowed for multiple gateways in our backhaul network. A simple construction shows that we can transform such a network into an equivalent one with only one gateway and so the work in this paper can be easily extended to multiple gateways. The construction adds a dummy gateway D and disjoint length one and length two paths from D to each real gateway. The reason for adding two paths for each gateway is that we do not want to force the parity of all the gateways to be the same. The links on these new paths are all given infinite capacity. Then this new network can be treated exactly as one with just one gateway D.

In this paper, scheduling has been constrained so that each node is in each of transmit mode and receive mode exactly half of the time. We are exploring the benefits of relaxing this constraint while still being able to bound the end-to-end delays.

Thus far, the problems considered have assumed that the gateway(s) are given. But an area of future work is to determine optimal placement of the gateways [6], much like the facilities location problem [27]; our goal is to optimize for both throughput and delay.

IX. Summary

We have presented the Even-Odd link activation framework, which provides a simple mechanism for implementing wireline scheduling policies locally at each access point in a wireless backhaul network. This framework is applicable to any backhaul topology, provided that all routes consist of alternately labelled nodes. The Even-Odd framework allows bounded-delay schedulers such as WFQ and CEDF to be efficiently mapped to our multihop wireless network. We outlined an ILP-based formulation for constructing an optimal routing, a problem that we showed to be NP-complete. Faster, heuristic-based approaches for routing were shown to perform fairly well. Finally, we demonstrated through extensive simulations that the Even-Odd framework allows for lower delays compared to existing scheduling mechanisms.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of ACM SIGCOMM*, August 2004.

[2] M. Andrews and L. Zhang. Minimizing end-to-end delay in high-speed networks with a simple coordinated schedule. In *Proceedings of IEEE INFOCOM*, pages 380 – 388, New York, NY, March 1999.

[3] Lichun Bao and J. J. Garcia-Luna-Aceves. Transmission Scheduling in Ad Hoc Networks with Directional Antennas. In *Proceedings ACM MobiCom*, pages 48–58, New York, September 2002.

[4] J. Y. Le Boudec and P. Thiran. *Network Calculus*. Springer Verlag, http://ica1www.epfl.ch/PS_files/NetCal.htm, 2004.

[5] Benjamin A. Chambers. The Grid Roofnet: A Rooftop Ad Hoc Wireless Network. Master's thesis, Massachusetts Institue of Technology, May 2002. http://www.pdos.lcs.mit.edu/grid/pubs.html.

[6] R. Chandra, L. Qiu, K. Jain, and M. Mahdian. Optimizing the placement of integration points in multi-hop wireless networks. In *Proceedings of IEEE ICNP*, October 2004.

[7] Imrich Chlamtac and András Faragó. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Transactions on Networking*, 2(1):23–29, 1994.
[8] Romit Roy Choudhury, Xue Yang, Nitin H. Vaidya, and Ram Ramanathan. Using directional antennas for medium access control in ad hoc networks. In *Proceedings of ACM MobiCom*, pages 59–70, 2002.
[9] R. Cole, K. Ost, and S. Schirra. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. *Combinatorica*, 21:5 – 12, 2001.
[10] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Math.*, 1:269 – 271, 1959.
[11] V. Erceg and K.V.S. Hari et al. Channel models for fixed wireless applications. *IEEE 802.16 Broadband Wireless Access Working Group*, IEEE 802.16.3c-01/29r4, 2001. http://ieee802.org/16.
[12] V. Erceg and L.J.Greenstein et al. An empirically based path loss model for wireless channels in suburban environments. *IEEE/ACM Journal on Selected Areas in Communications*, 17(7):1205 – 1211, 1999.
[13] Violeta Gambiroza, Bahareh Sadeghi, and Edward W. Knightly. End-to-end performance and fairness in multihop wireless backhaul networks. In *Proceedings of ACM MobiCom*, pages 287–301, 2004.
[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, CA, 1979.
[15] Xiao Long Huang and Brahim Bensaou. On max-min fairness and scheduling in wireless ad-hoc networks: analytical framework and implementation. In *Proceedings of ACM MobiHoc*, pages 221–231, 2001.
[16] IEEE Computer Society and IEEE Microwave Theory and Techniques Society. *802.16-2004: Air Interface for Fixed Broadband Wireless Aceess Systems*, Oct 2004. IEEE Standards.
[17] Kamal Jain, Jitendra Padhye, Venkata N. Padmanabhan, and Lili Qiu. Impact of Interference on Multi-hop Wireless Network Performance. In *Proceedings of ACM MobiCom*, pages 66–80, 2003.
[18] Ji-Her Ju and Victor O. K. Li. An optimal topology-transparent scheduling method in multihop packet radio networks. *IEEE/ACM Transactions on Networking*, 6(3):298–306, 1998.
[19] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly. Distributed multi-hop scheduling and medium access with delay and throughput constraints. In *Proceedings of ACM MobiCom*, 2001.
[20] R. Karrer, A.Sabharwal, and E.Knightly. Enabling Large-scale Wireless Broadband: The Case for TAPs. In *2nd Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge, MA, November 2003.
[21] M. Kodialam and T. Nandagopal. Characterizing the achievable rates in multihop wireless networks. In *Proceedings of ACM MobiCom*, San Diego, CA, August 2003.
[22] I. Koffman and V.Roman. Broadband wireless access solutions based on OFDM access in IEEE 802.16. *IEEE Communications Magazine*, 40(4):96 – 103, 2002.
[23] D. König. Graphok és alkalmazásuk a determinánsok és a halmazok elméletére (in Hungarian). *Mathematikai és Természettudományi Értesítő*, 34:104–119, 1916.
[24] C. Li and E. Knightly. Coordinated network scheduling: A framework for end-to-end services. In *Proceedings of IEEE ICNP*, Osaka, Japan, November 2000.
[25] Haiyun Luo, J. Cheng, and Songwu Lu. Self-coordinating localized fair queueing in wireless ad hoc networks. *IEEE/ACM Transactions on Mobile Computing*, 3(1):86–98, Jan-Feb 2004.
[26] Haiyun Luo, Songwu Lu, and Vaduvur Bharghavan. A new model for packet scheduling in multihop wireless networks. In *Proceedings of ACM MobiCom*, pages 76–86, 2000.
[27] R. Mirchandani and R. Francis, editors. *Discrete Location Theory*. John Wiley & Sons, Inc., New York, NY, 1990.
[28] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case. *IEEE/ACM Transactions on Networking*, 2(2):137 – 150, 1994.
[29] Ram Ramanathan. On The Performance of Ad Hoc Networks Using Beamforming Antennas. In *Proceedings of ACM MobiHoc*, Long Beach, California, USA, Oct. 2001.
[30] S. Ramanathan. A Unified Framework and Algorithm for Channel Assignment in Wireless Networks. In *Proceedings IEEE INFOCOM*, Kobe, Japan, 1997.
[31] H. Viswanathan and S. Mukherjee. Performance of cellular networks with relays and centralized scheduling. In *Vehicular Technology Conference, VTC 2003-Fall*, pages 1923–1928, 2003.
[32] WiMAX Forum. Business case models for fixed broadband wireless access based on WiMAX technology and the 802.16 standard. *Whitepaper*, October 2004. http://www.wimaxforum.org.

APPENDIX

A feasible routing was defined in Section IV.

Theorem 4.1: Given the link capacities and the connection bit rates, determining if there is a feasible routing is NP-complete.

Proof: Clearly checking if a routing is feasible can be done in polynomial time. We show that the problem is NP-hard by a reduction from the known NP-complete problem called 3-PARTITION [14]. An instance of 3-PARTITION consists of a set A of $3m$ elements, a positive integer B and a positive integer size $s(a)$ for each $a \in A$ where $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = mB$. The goal is to find a partition of A into m disjoint sets $A_1, A_2, \ldots, A_m$ such that for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$. Notice that if such a partition exists, then each A_i contains exactly 3 elements of A. This problem is NP-complete in the strong sense.

Consider an instance I of 3-PARTITION. We now describe an instance J of the feasible routing problem based on I that can be constructed in time polynomial in I and such that I has a solution if and only if J does.

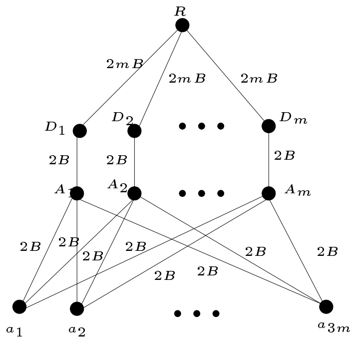


Fig. 14. Illustrating the NP-completeness construction.

The construction is basically shown in Figure 14. There is a root node R, a node labeled a_i for each $a_i \in A$, nodes labeled $A_1, A_2, \ldots, A_m$ representing a partition of A and dummy nodes labeled $D_1, D_2, \ldots, D_m$. Each node a_i has a connection in the up link direction with rate $s(a_i)$. These are the only connections.

Links of the form (a_i, A_j) or (A_j, D_j) have capacity $2B$ while links of the form (D_j, R) have capacity $2mB$. All other links (not shown) have capacity 0.

Suppose more connections with total rate greater than B are routed through A_j. Then the congestion at A_j due to the link (A_j, D_j) is more than $B/2B = 1/2$ and hence this would not be a feasible routing. Thus for any feasible routing each A_j must have connections routed through it with total rate exactly B. It's easy to check that any routing that results in every node A_j have connections with total rate B routed through it is a feasible routing. Hence this feasible routing instance has a solution if and only if the 3-Partition instance has a solution. ∎