Delay Aware Link Scheduling for Multi-Hop TDMA Wireless Networks

Petar Djukic, *Member, IEEE*, and Shahrokh Valaee, *Senior Member, IEEE*

***Abstract*—Time division multiple access (TDMA) based medium access control (MAC) protocols can provide QoS with guaranteed access to the wireless channel. However, in multi-hop wireless networks, these protocols may introduce scheduling delay if, on the same path, an outbound link on a router is scheduled to transmit before an inbound link on that router. The total scheduling delay can be quite large since it accumulates at every hop on a path. This paper presents a method that finds conflict-free TDMA schedules with minimum scheduling delay.**

We show that the scheduling delay can be interpreted as a cost, in terms of transmission order of the links, collected over a cycle in the conflict graph. We use this observation to formulate an optimization, which finds a transmission order with the min-max delay across a set of multiple paths. The min-max delay optimization is NP-complete since the transmission order of links is a vector of binary integer variables. We devise an algorithm that finds the transmission order with the minimum delay on overlay tree topologies and use it with a modified Bellman–Ford algorithm, to find minimum delay schedules in polynomial time. The simulation results in 802.16 mesh networks confirm that the proposed algorithm can find effective min-max delay schedules.

***Index Terms*—Scheduling delay, stop-and-go queueing, TDMA scheduling algorithms.**

I. Introduction

NEW applications of wireless multi-hop networks, such as commercial mesh networks, require guaranteed quality-of-service (QoS) in the MAC layer. This has prompted development of new multi-hop MAC protocols based on time division multiple access (TDMA), such as the 802.16 mesh protocol [2] and the 802.11s mesh deterministic access (MDA) protocol [3]. The new protocols provide guaranteed link bandwidth with scheduled access to the wireless channel. The link bandwidth is allocated over frames with a fixed number of slots. A schedule assigns slots to links and during each slot a number of nonconflicting links can transmit simultaneously, taking advantage of spatial reuse. The bandwidth of each link is given by the number of slots it is assigned in the frame.

An important goal for TDMA scheduling algorithms is to find the minimum number of slots required to schedule requested end-to-end rates. If TDMA frame length is variable, minimizing the number of slots in the frame maximizes the concurrent throughput of end-to-end flows. On the other hand, if the frame length is fixed, minimum length TDMA scheduling maximizes the portion of the frame available for best-effort wireless access. In this case, the minimum length TDMA schedule also has minimum utilization, where utilization is defined as the portion of active slots in the frame, used for TDMA scheduled traffic.

Although previous TDMA scheduling approaches [5]–[16] can find minimum length schedules, they do not account for TDMA scheduling delay. Scheduling delay occurs when packets arriving on an inbound link must wait for the subsequent frame to be transmitted on the outbound link. Since TDMA wireless networks are stop-and-go queueing systems [17], there is no queueing delay and packets experience delay due scheduling delay alone. Scheduling delay accumulates at every hop in the network, so end-to-end delay experienced on a path can be large. In this paper, we address the following important problem: *Given an assignment of link bandwidths, what is the minimum length TDMA schedule that also minimizes end-to-end scheduling delay?* We call this problem the delay aware link scheduling problem.

We solve the delay aware link scheduling problem in two parts. First, we develop a new class of TDMA algorithms that find conflict-free TDMA schedules. A TDMA schedule is conflict-free if all links whose packets collide in simultaneous transmissions transmit at non-overlapping times. We derive the necessary and sufficient conditions that a TDMA schedule is conflict-free as a set of linear inequalities. The conflict-free linear inequalities correspond to pairwise conflicts in the network and each inequality is defined by the transmission duration of the links in the conflict, the activation times of the links in the conflict and the transmission order of the links. We show that for fixed transmission orders, the inequalities can be solved in polynomial time with the Bellman–Ford algorithm, so minimum length TDMA schedules can be found in polynomial time.

Second, we show that the transmission order defines the end-to-end scheduling delay. This allows us to formulate a $\{0, 1\}$-integer linear program that finds a min-max delay for a subset of paths in the network. The number of binary variables in the linear program is equal to the number of conflicts in the network and corresponds to the link transmission order. We use the optimization in an iterative procedure to find minimum length schedules, which also have the min-max scheduling delay property. We devise a polynomial algorithm for overlay tree topologies that finds transmission orders whose scheduling delay is at most one frame. We use this algorithm in an iterative procedure to find minimum length schedules with the minimum, one frame, scheduling delay. Since the two most important and common examples of wireless multi-hop—sensor

Manuscript received January 27, 2008; revised February 20, 2008 and July 02, 2008; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Das. First published October 03, 2008; current version published June 17, 2009. This work was supported in part by the LG Electronics Corporation. A preliminary version of this work was presented at the IEEE INFOCOM 2007, Anchorage, AK.

The authors are with The Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, M5S 3G4, Canada (e-mail: djukic@comm.utoronto.ca; valaee@comm.utoronto.ca).

Digital Object Identifier 10.1109/TNET.2008.2005219

networks and mesh networks—use overlay tree topologies, the iterative algorithm is applicable in a wide array of scenarios.

We examine the applicability of the scheduling algorithms to 802.16 mesh networks with numerical simulations. We compare the performance of our algorithm to algorithms proposed for 802.16 networks [15], [16], which are based on graph coloring.

A. Related Work

The TDMA scheduling problem is to find a conflict-free TDMA schedule for a given set of link rates. A related problem is to find a restriction on a set of link rates, which ensures these link rates are admissible—they can be scheduled. If the set of admissible link rates is known, it can be used to formulate and solve cross-layer design problems such as joint end-to-end rate control and scheduling optimizations [4], [9] and joint routing and scheduling optimizations [8], [10]. We first review related work in scheduling and then the related work defining the set of admissible link rates.

The scheduling problem can be formulated as a graph coloring problem, since wireless conflicts can be modeled with interference (conflict) graphs [5], [6]. Conflict graphs consists of links as vertexes and link conflicts as arcs. Arcs in the conflict graph connect links, which cannot transmit simultaneously due to mutual interference. A graph coloring on the conflict graph is a conflict-free schedule [6]–[13], as is a set of independent sets with appropriate cardinality [5]. It is also possible to solve the scheduling problem by finding cliques in the complement of the conflict graph—the compatibility graph [18], [19].

In the special case, where only the conflicts between links sharing a neighbor exist (primary conflicts), a vertex coloring on the conflict graph is equivalent to an edge coloring on the topology graph [8]–[10]. An edge coloring is also called a *matching* for the graph. This special case of conflicts is based on the assumption that the conflicts between links not sharing a neighbor (secondary conflicts) can be removed by careful assignment of links to orthogonal channels, in frequency or in code [8]. Conflicts can also be removed with directional antennas [14]. The technological complexities of channel assignment can be mitigated with the use of asynchronous TDMA [9]. With these assumptions, it is possible to find TDMA schedules in polynomial time [8], or with edge-coloring heuristics with well-defined lower bounds on performance [20]. It is also possible to devise distributed algorithms, which approximate centralized matching algorithm within a constant factor [13].

Although the removal of secondary conflicts simplifies the scheduling problem, it is still important to consider scheduling with all conflicts. Finding a channel assignment to remove secondary conflicts is still a vertex coloring problem, which is NP-complete [21]. Without the orthogonal channel assignment, the secondary conflicts can be removed from schedules based on matching by reversing the direction of transmissions on the links [12]. The edge-coloring approach can also be extended by restricting the coloring to ensure that links that are two or more hops away have different colors—K–hop coloring [11]. For special graph topologies, it is possible to develop a polynomial-time approximation scheme (PTAS) algorithm to find schedules for K-hop coloring problems [11]. Graph heuristics can also be used [14]–[16].

In this work, we introduce the delay aware link scheduling problem, which significantly extends the previous related works [5]–[16]. Our scheduling approach allows us to separate the complexity of finding optimal schedules from scheduling. We show that by fixing a single network parameter—end-to-end scheduling delay—the scheduling problem becomes that of finding minimum paths in the conflict graph. Since these paths can be found with the Bellman–Ford algorithm, schedules can even be found with the *distributed* Bellman–Ford algorithm that does not require the direct knowledge of the full network topology [22]. The distributed and centralized scheduling algorithms compute identical schedules. Our scheduling approach also allows us to schedule links for transmissions once per frame, or to schedule links multiple times per frame. The limitation on the number of transmissions significantly reduces overhead.

An algorithm to solve a different delay aware scheduling problem is proposed in [23]. That paper proposes an "odd-even" TDMA regime, which groups links in two categories and schedules the links from the two categories at alternate times. Conflicts are resolved with a routing heuristic and with an additional operation that assigns orthogonal frequencies to links. In the odd-even TDMA regime, the delay comes from queueing, while in regular TDMA networks, considered in this paper, delay is due to the TDMA scheduling delay.

With only the primary conflicts, the set of achievable rates can be defined with the linear inequalities associated with each odd subset of the links [8]. Even though there are an exponential number of these subsets, they can be used to show, in polynomial time, if a set of link rates is admissible [8]. It is also possible to define a necessary set of conditions for achievable rates [10], which can be achieved using an edge-coloring heuristic [20]. For general conflict graphs, the set of achievable rates can also be defined with the independent vertex set polytope of the conflict graph [5]. However, since there are an exponential number of independent sets in the graph, a heuristic is needed to iteratively expand the feasibility conditions by adding independent sets [5].

We derive a set of linear constraints, which specify necessary and sufficient conditions for the existence of a TDMA transmission schedule. The number of these constraints is polynomial in the number links and when the end-to-end scheduling delay in the network is fixed, they can be used to verify if link rates are admissible in polynomial time. If link rates are admissible a schedule can be found for them in polynomial time with the Bellman–Ford algorithm.

II. Network and Transmission Model

We model the network with a topology graph connecting the nodes in the wireless range of each other. We assume that if two nodes are in the range of each other, they establish links in the MAC layer, so the TDMA network can be represented with a *directed* connectivity graph $G(V, E)$, where $V = \{v_1, \ldots, v_n\}$ is the set of nodes and $E = \{e_1, \ldots, e_m\}$ is the set of directed links.

We assume that a routing protocol establishes routes between nodes. In the rest of the paper, we assume that paths form an *overlay* tree using a subset of the links, E, since this is the topology commonly used to manage mesh networks [2], [3].

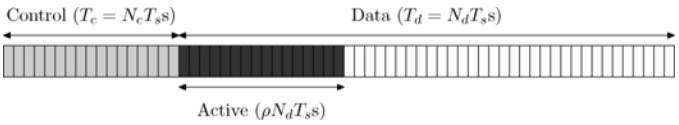

Fig. 1. TDMA framing.

Nevertheless, the scheduling algorithms presented in this paper do not depend on the network topology defined by link connectivity, E. Only the minimum delay scheduling algorithm (Section V-C) depends on the overlay tree topology.

We assume a standard TDMA model, which also corresponds to the operation of 802.16 mesh networks [2] and 802.11s mesh networks [3]. Time is divided into slots of fixed duration, which are then grouped into frames. The duration of each slot is T_s seconds and there are a total of N_f slots in each frame making the frame duration $T_f = N_f T_s$ seconds. There are N_c slots reserved for the control traffic and N_d slots reserved for data traffic (Fig. 1). In the rest of the paper, we assume that the control slots are grouped together at the beginning of the frame, as in the 802.16 mesh protocol. However, we show that our scheduling algorithms work even without such grouping.

Unless otherwise stated, in the rest of the paper we assume that links are scheduled to transmit once per frame (slots are allocated contiguously). Contiguous slot allocation is needed for TDMA standards such as 802.16 mesh networks [2] and to decreases the overhead of TDMA MAC protocols, which space-out link transmissions to handle imperfect synchronization in wireless networks. Nevertheless, we also show how to modify our scheduling algorithms to produce schedules with non-contiguous slot allocations.

One of the goals of this paper is to find minimum length schedules. If the frame size is fixed, minimum length TDMA scheduling also minimizes utilization.

Definition 1 (Frame Utilization): The ratio of slots carrying traffic to the total number of slots in the data sub-frame is called frame utilization, ρ.

A. Stop-and-Go Queueing in TDMA Networks

TDMA wireless networks are stop-and-go queueing systems. Each link is equivalent to a server in stop-and-go queueing, where the servers are active when TDMA links are transmitting. For no queueing delays anywhere in the network, it is sufficient for each link rate to satisfy

$$r_j = \sum_{P_l \in \mathcal{P}} g_l I(e_j \in P_l) \tag{1}$$

where $\mathcal{P}$ is the set of all paths found by the routing algorithm, $I(\cdot)$ is the indicator function, which is 1 when its argument is true and 0 when its argument is false and g_l is the requested end-to-end rate of connection l, using the network path P_l, in bits-per-second. We assume that the network provides a mechanism (e.g., TDMA MAC protocol) for wireless nodes to request end-to-end bandwidths and establish network wide schedules.

For link e_j, we use Δ_j to indicate the number of slots allocated to the link in each frame and $d_j = T_s \Delta_j$ to indicate the duration of the link's transmission. The number of slots required by the link can be found from the link rate with

$$\Delta_j = \left\lceil \frac{r_j T_f}{c_j T_s} \right\rceil + h \tag{2}$$

where $\lceil \cdot \rceil$ is the ceiling function, r_j is the link rate, c_j is the bit-rate on the link and h is the spacing (in slots) between transmissions of different links. The numerator is the number of bits the link transmits in each frame and the denominator is the number of bits the link can transmit in each slot. The overhead of h slots is needed to space-out TDMA transmissions and compensate for synchronization errors between the wireless nodes.

A scheduling algorithm may not be able to allocate the number of slots requested on the link, if too many slots are requested by all links. In such a case, the scheduler adjusts the link slot allocations. If link e_j's slot allocation Δ_j is adjusted to $\hat{\Delta}_j$, the actual rate on link e_j is obtained with

$$\hat{r}_j = \left(\hat{\Delta}_j - n_j h\right) c_j \frac{T_s}{T_f} \tag{3}$$

where n_j is the number of times the link transmits in the frame.

Ideally, the requested link rates, (1), match the link rates provided by the schedule, (3). If the two sets of rates do not match, we can achieve Generalized Processor Sharing (GPS) fairness [24] among end-to-end connections, by setting the end-to-end rate for connection l to

$$\hat{g}_l = \min_{e_j \in P_l} g_l \frac{\hat{r}_j}{r_j} \tag{4}$$

where P_l is the path traversed by the connection [25]. In addition to giving each connection a proportional share of the link bandwidth, the assignment of link rates with (4) also ensures that

$$\hat{r}_j \geq \sum_{P_l \in \mathcal{P}} \hat{g}_l I(e_j \in P_l) \tag{5}$$

so data arriving on e_j is always transmitted on the next link in the path by the end of next frame, consistent with stop-and-go queueing.

B. Wireless Interference Model

Wireless links interfere (conflict) with other links, if their packets collide at a receiver in simultaneous transmissions. We first examine conflicts between individual links and then show how to construct the conflict graph for the network, which keeps track of conflicts between all links. The conflicts are based on the distance model of interference, which assumes that two links interfere with each other at a receiver, if the receiver cannot decode packets from either link. Shortly, we show how this model can be improved to include other interference models.

We show five types of transmission conflicts between isolated pairs of links in Fig. 2. The first three types of conflicts are between the links that share a neighbor—*primary conflicts*. In the case of the transmitter-transmitter (t-t) conflict, the parallel transmissions garble each other at the common receiver.

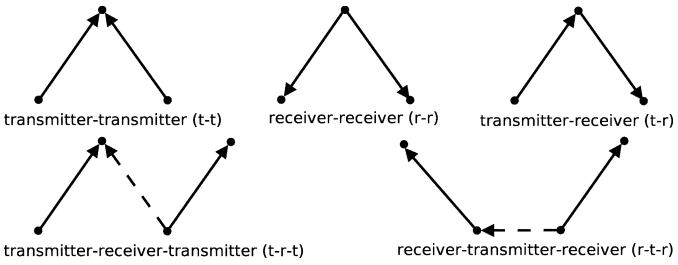

Fig. 2. Conflicts in TDMA wireless networks.

In the case of the receiver-receiver (r-r) conflict, a single transmitter cannot separate packets intended for two different receivers. The transmitter-receiver (t-r) conflict happens because the nodes cannot transmit and receive at the same time. It is possible to remove this conflict, if nodes can transmit in full duplex. However, in the rest of the paper, we assume that nodes transmit in half-duplex, so we take this conflict into account.

In addition to the three direct neighbor conflicts, TDMA networks also have a restriction on their second hop neighbor links – *secondary conflicts*. We show this as the transmitter-receiver-transmitter (t-r-t) conflict. In the t-r-t conflict, the two conflicting links are shown with a solid line. They cannot transmit at the same time because the two transmitters share a neighbor, which hears both transmissions, shown with the dashed line for the overheard transmission.

Only the first four conflicts need to be considered in TDMA networks [6]. However, a fifth type of conflict also exists in wireless networks. We show this as the receiver-transmitter-receiver (r-t-r) conflict. In the r-t-r conflict, the two conflicting links cannot transmit at the same time because the two transmitters can overhear each other. Normally, this is not a problem since a sender can transmit while other transmissions are ongoing. However, in 802.11 networks [26] this conflict exists because during request-to-send (RTS)/clear-to-send (CTS) handshake between a transmitter and a receiver, the transmitter cannot send an RTS packet if it overhears another transmission [6]. In this paper, we assume that the network is using TDMA, so the r-t-r conflict does not exist in any of our conflict graphs.

We keep track of conflicts between links with conflict graphs. We define conflict graphs with $G_c(E, C)$, where E is the set of links and C is the set of directed arcs, representing link conflicts, one for each of the r conflicting pairs of links. The orientation of the conflict arcs does not cause any loss of generality, however it simplifies the derivation of the results in this paper. In the sequel, we use c_{ij} to denote the conflict between links e_i and e_j.

We use a six node topology [Fig. 3(a)] to demonstrate how conflict graphs are created. The vertexes in the conflict graph are the 10 links from the connectivity graph. The links that conflict with each other (e.g., e_1 and e_3) are connected with an arc in the conflict graph [$c_{1,3}$, Fig. 3(b)]. On the other hand, the links that do not conflict with each other (e.g., e_2 and e_6) are not connected with an arc in the conflict graph. The orientation of links corresponds to link identifiers, so if for two conflicting links e_i and e_j, $i < j$, their conflict arc is c_{ij}, originating at e_i and terminating at e_j.

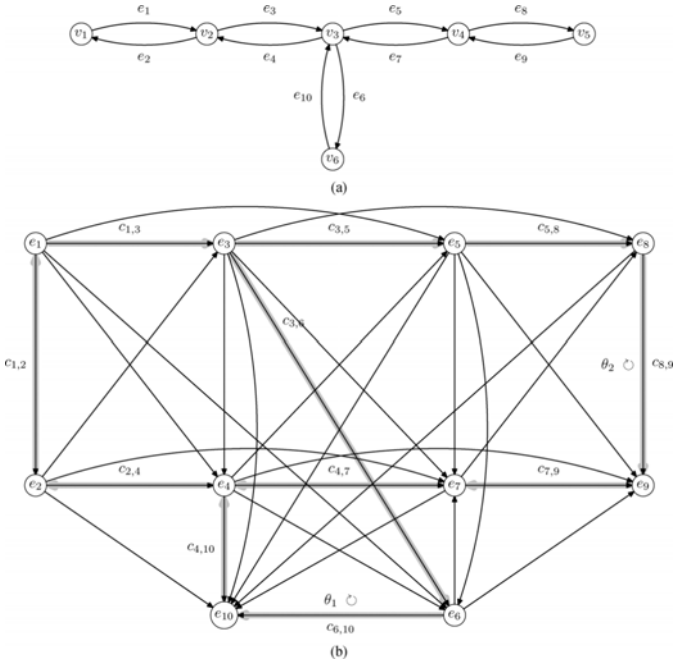

Fig. 3. Topology and conflict graphs for a simple topology. (a) Topology graph. (b) Conflict graph.

We note that the conflict-graph can allow us to use other interference models. For example, if we consider node scheduling, where instead of links we schedule nodes, the conflict graph can be used to track conflicts between nodes, for the protocol model of interference [27]. In the protocol model, two nodes are allowed to transmit simultaneously if their transmissions can be decoded, separately, by the same receiver. Since the algorithms presented in the next section can only depend on the knowledge of the conflict graph, they can be used to solve scheduling problems under the protocol model. The conflict graph can also be used to track conflicts if links have different transmission and interference ranges and the information distinguishing the two is available. We do not explore these models of interference further in this paper.

In this paper, we do not take the effect of cumulative interference into account – the physical interference model [27]. In the physical interference model, the reception of a packet depends on the difference between the received signal strength of its sender and the accumulated interference at the receiver, which is the total received signal strength of other concurrent transmitters plus noise. However, a simple iterative procedure can be used to approximate the effect of accumulative interference with conflict graphs [28]. The iterative procedure [28], starts with a conflict graph, which only includes links whose mutual interference exceeds a required threshold. In each step, the procedure finds a schedule and measures the accumulated interference for each slot, at each receiver. If for some slot the interference at a receiver exceeds the required threshold, all links active in that slot are connected with each other in the conflict graph and scheduling is repeated again. The procedure stops when the interference at each receiver is below the required threshold for all slots.

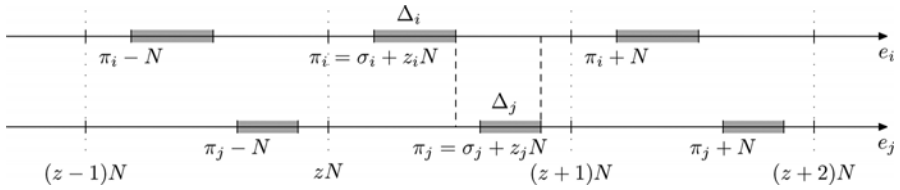

Fig. 4. Generalized activation times for conflicting links e_i and e_j.

III. TDMA Scheduling

In this section, we discuss the TDMA scheduling problem when there are no control slots and no restrictions on scheduling delay. We show how the algorithm can be changed to allow for control traffic, corresponding to TDMA protocols such as 802.16 [2] and 802.11s [3] in the next section. We show how to account for TDMA scheduling delay in Section V.

Initially, we assume that slots are allocated contiguously and derive the necessary and sufficient conditions that make a schedule conflict-free over an arbitrary number of slots. Then, we show that for a fixed transmission order we can use these conflict-free conditions to derive a polynomial time scheduling algorithm. Finally, we discuss how the algorithm can be modified to accommodate multiple link transmissions in the frame.

A. Conflict-Free Schedules Over N Slots

We develop a set of necessary and sufficient conditions that ensure that a set of link rates can be scheduled with a conflict-free TDMA schedule over frames N slots long. We assume that slots are allocated contiguously and use the notation that σ_j is the activation slot of link e_j in the frame and that it transmits continuously for Δ_j subsequent slots. We also call these activation times, *normalized* activation times since for each link e_j, $0 \leq \sigma_j \leq N$. We call the set of activation times for the whole network, $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_m]$, a (normalized) schedule.

We start by observing that the repetitive nature of TDMA schedules means that an activation time σ_i for link e_i actually represents a series of activation times (Fig. 4). The series of activation times can be derived from σ_i by adding multiples of N slots, $\Pi_i = \{\sigma_i + z_i N, z_i \in \mathbb{Z}\}$. Conversely, σ_i can be found from any activation time $\pi_i \in \Pi_i$ with the modulo operator:

$$\sigma_i = \pi_i (\mathrm{mod}\ N). \tag{6}$$

Since π_i's are not, in general, restricted to the range $[0, N]$, we call π_j a *generalized* activation time of link e_j and we call a subset of generalized activation times $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_m]$, where for a link j, $\pi_j \in \Pi_j$, is a *generalized* schedule. A (normalized) schedule, $\boldsymbol{\sigma}$, can always be obtained from a generalized schedule $\boldsymbol{\pi}$ by applying the modulo operator, (6), to each member of $\boldsymbol{\pi}$.

We establish the conflict-free conditions using the generalized schedules. We consider a pair of links e_i and e_j, whose conflict is c_{ij}. Take any generalized activation time $\pi_i \in \Pi_i$ for link e_i and choose the next generalized activation time for a conflicting link e_j, $\pi_j = \min\{\pi \in \Pi_j : \pi \geq \pi_i\}$ (Fig. 4). In this case, e_j should not transmit before e_i finishes its transmission

$$\pi_j \geq \pi_i + \Delta_i \Leftrightarrow \quad \pi_j - \pi_i \geq \Delta_i \tag{7}$$

and e_j should stop transmitting before e_i transmits again

$$\pi_j + \Delta_j \leq \pi_i + N \Leftrightarrow \quad \pi_j - \pi_i \leq N - \Delta_j. \tag{8}$$

The equations can be combined to arrive at the following conflict-free condition:

$$\Delta_i \leq \pi_j - \pi_i \leq N - \Delta_j. \tag{9}$$

In the above example, we assume that e_i transmits first in each frame. If we change the order of transmissions, we have

$$\Delta_j \leq \pi_i - \pi_j \leq N - \Delta_i. \tag{10}$$

We can combine the two conflict-free conditions further since their ranges of $\pi_j - \pi_i$ are mutually exclusive:

$$\Delta_i \leq \pi_j - \pi_i + o_{ij} N \leq N - \Delta_j \tag{11}$$

where $o_{ij} = 0$ if $\pi_j - \pi_i > 0$ and $o_{ij} = 1$ if $\pi_j - \pi_i < 0$. The extra variable o_{ij} specifies a relative order of transmissions, which prompts us to refer to it as the *transmission order* in the rest of the paper.

We "stack" the inequalities on top of each other, to define a $\{0, 1\}$-integer linear program, which finds generalized feasible conflict-free schedules for N slots:

$$\begin{aligned} \text{Find} \quad & \boldsymbol{\pi}, \boldsymbol{o} && \text{(12a)} \\ \text{s.t} \quad & \boldsymbol{l} \leq \boldsymbol{C}^T \boldsymbol{\pi} - \boldsymbol{o} N \leq N\mathbf{1} - \boldsymbol{u} && \text{(12b)} \\ & \boldsymbol{\pi} \in \mathbb{Z}^m, \boldsymbol{o} \in \{0,1\}^r && \text{(12c)} \end{aligned}$$

where $\boldsymbol{C}$ is the $m \times r$ incidence matrix of the conflict graph where the rows correspond to the vertexes and columns correspond to the arcs of the conflict graph and entries in the matrix are defined as $\boldsymbol{C}_{ik} = 1$ if k-th column corresponds to arc c_{ij}, $\boldsymbol{C}_{ik} = -1$ if k-th column corresponds to arc c_{ji} and $\boldsymbol{C}_{ik} = 0$ otherwise, $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_m]^T$, $\boldsymbol{o} = [\ldots, o_{ij}, \ldots]^T$ corresponding to columns of $\boldsymbol{C}$, $\boldsymbol{l} = [\ldots, l_{ij}, \ldots]^T$ and $\boldsymbol{u} = [\ldots, u_{ij}, \ldots]^T$, where each entry corresponds to a column of $\boldsymbol{C}$, $u_{ij} = \Delta_j$ and $l_{ij} = \Delta_i$, and $\mathbf{1}$ is a column vector of m 1's. The problem in this form is a special instance of the Periodic Event Scheduling Problem (PESP), which was shown to be NP-complete by reduction from Hamiltonian Path [29].

For the special case of $N = N_d$, search problem (12) can be used to find out if a set of link rates is admissible. Since we assumed that link rates are given, they may not be admissible for N_d slots, in which case they should be modified to find a schedule. We address one way of modifying the link rates to make them admissible in the next section.

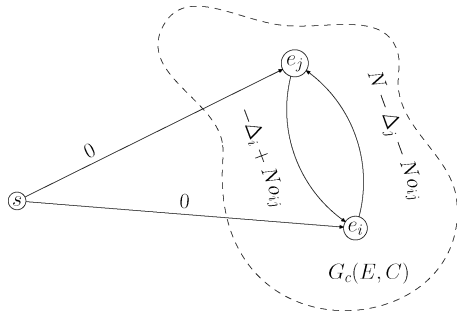

Fig. 5. Scheduling graph.

B. Conflict-Free Scheduling for Fixed Transmission Orders

Even though the general scheduling problem is NP-complete, if the transmission order $\boldsymbol{o}$ is fixed, the scheduling problem has polynomial complexity. In the next section, we show that the transmission order is directly related to scheduling delay and propose methods to find transmission orders with minimum delay. For now, we assume that the transmission order is known prior to solving (12).

The linear inequalities can be solved efficiently using the Bellman–Ford shortest path algorithm [30, pp. 532–535] or Dijkstra's shortest path algorithm [31, pp. 194–200]. In addition to the centralized TDMA scheduling algorithm, which we present in this section, we have also proposed a decentralized TDMA scheduling algorithm, which is also based on the Bellman–Ford shortest path algorithm [22]. Due to page restrictions we only briefly outline the centralized version of the TDMA scheduling algorithm; full description of the algorithm can be found in [32, pp. 50–54].

The algorithm starts by adding an extra vertex s to the conflict graph and then connects the new vertex to each of the original vertexes in the graph with an arc of cost 0 (Fig. 5). Each arc in the original conflict-graph is replaced with two new arcs, which get their costs from the upper and lower bounds in (11). For example, for conflict c_{ij} we can rewrite (11) as

$$\Delta_i - o_{ij}N \leq \pi_j - \pi_i \leq N - \Delta_j - o_{ij}N$$

so the new arc, which connects (e_i, e_j) has the cost of $N-\Delta_j-o_{ij}N$, and the new arc, which connects (e_j, e_i) has the cost of $\Delta_i - o_{ij}N$ (Fig. 5).

Next, the algorithm finds the shortest distance from s to each $e_j \in E$, with the Bellman–Ford shortest path algorithm.

Proposition 1 (Shortest Path Tree is a Schedule): The distance from s to each e_j in the scheduling graph is a conflict-free generalized schedule.

Proof: This is easily seen by the optimality of the shortest paths. For each $e_i \in E$, and all conflicts arcs originating at e_i, c_{ij} we have

$$\pi_j \leq \pi_i + N - \Delta_j - o_{ij}N \tag{13a}$$

$$\pi_i \leq \pi_j - \Delta_i + o_{ij}N \tag{13b}$$

where π_i is the cost of reaching e_i from s and π_j is the cost of reaching e_j from s. Combined, these equations correspond to (11). ■

The final step of the algorithm is to use the modulo operator to find normalized schedule $\boldsymbol{\sigma}$ from $\boldsymbol{\pi}$.

In the case that the Bellman–Ford algorithm cannot find minimum paths, the link durations cannot be supported by the given transmission order, over N slots. At this point either the transmission order should be changed, or N should be increased to find a schedule that supports the link durations.

C. Multiple Link Transmissions in a Frame

We now show how the scheduling algorithm can be modified to allow for multiple transmissions in the frame, showing that our scheduling approach is more general than graph coloring approaches since it *can* limit the number of link transmissions in the frame. Previous approaches to TDMA scheduling use graph coloring to produce conflict-free schedules [5]–[13].

We create a new conflict graph from the original conflict graph by "splitting" the vertexes associated with the links. We split each vertex in the original conflict graph the number of times equal to the duration of the link associated with the vertex. The vertexes are split until the new conflict graph has $N_{\max} = \sum_{i=1}^{m} \Delta_i$ vertexes, and the duration of each vertex is 1 slot. In the new graph, all vertexes whose links conflicted also conflict. In addition, vertexes derived from the same link conflict with each other. A schedule in the new graph is also a graph coloring and corresponds to a conflict-free TDMA schedule, which allows links to transmit multiple times in the same frame.

One way to derive arc orientation in the new graph is to use the orientation from the original graph. For a conflict c_{ij} in the original graph, in the new graph, we direct the arcs from each copy of e_i to each copy of e_j. This arc orientation does not introduce any loss of generality, since the transmission order, and not the arc orientation, determines the precedence of transmissions in the new graph.

In the case of the new conflict graph, the transmission order and the arc orientation are a vertex ordering. Given the vertex ordering, we find a graph coloring in the new graph with the Bellman–Ford algorithm. While it may seem suspect that a shortest path algorithm can find a coloring in a graph, since graph coloring is an NP-complete problem [21], this is not a new result [33], [34]. Fixing the vertex ordering in the graph removes the complexity from the problem.

In addition to significantly increasing the size of the scheduling problem, splitting the link transmissions in this way does not allow an easy way to limit the number of times a link transmits in a frame. Recall that transmissions of different links must be spaced-out to account for synchronization errors. We show the impact of multiple link transmissions with simulations.

IV. Minimum Length TDMA Scheduling

The algorithm we devised in the previous section assumes that there are N slots in the frame and only has two possible outcomes: either the schedule exists, in which case a schedule is provided, or no schedule exists. If no schedule exists, a feasible schedule can be found if the number of slots in the frame is increased. Increasing the frame size decreases all link rates proportionally to make them admissible. However, there are two reasons not to increase the frame size. First, the number of slots in the frame cannot be increased in protocols such as 802.16

[2] and 802.11s [3], which have fixed frame sizes ($N = N_d$). Second, as we show in the next section, scheduling delay is directly proportional to the frame size, so increasing the frame size also increases the delay.

In this section, we propose an algorithm that first finds a minimum number of slots required to schedule all links, $N_{\min}$, and then scales down the link rates with

$$\alpha = \frac{N_d}{\max\{N_{\min}, N_d\}} \tag{14}$$

so that it fits into a fixed frame with N_d data slots. Without the scaling, this algorithm finds the minimum length TDMA schedule, consistent with the previous literature in TDMA scheduling [8], [10]. Scaling down the link rates has the same effect as increasing the frame size, without the impact on delay or applicability of the algorithm to TDMA MAC protocols with fixed frame lengths such as 802.16 and 802.11s.

The scaled down schedule is also the schedule with minimum utilization, if the requested link rates are admissible ($N_{\min} < N_d$). In this case, the requested link rates can be scheduled without the scaling, frame utilization, $\rho = N_{\min}/N_d < 1$, is minimum and the maximum $(1 - \rho)N_d$ slots are available for best-effort traffic. If the link rates are not admissible, $N_{\min} > N_d$, the frame is fully utilized ($\rho = 1$) and no slots are available for best-effort traffic. In this case, the resulting end-to-end rates maximize the concurrent throughput in the network [10].

The scaling keeps the GPS-like fairness among the end-to-end connections that was first established with (1). By substituting (3) into (4) and observing that due to the scaling $\hat{\Delta}_j = \alpha\Delta_j$, we get

$$\hat{g}_l = \min_{e_j \in P_l} g_l \frac{\hat{r}_j}{r_j} = \min_{e_j \in P_l} g_l \frac{\hat{\Delta}_j}{\Delta_j} = \alpha g_l \tag{15}$$

where we assume that $h = 0$. So, each end-to-end connection gets a portion of the total bandwidth available for all end-to-end connections, which is proportional to its requested bandwidth.

Another way to modify link rates is by considering local link fairness [35], where instead of considering the end-to-end fairness of (15) over all paths, each link can be allocated slots with

$$\hat{\Delta}_j = \frac{r_j}{k \sum\limits_{e_j \in I_j} r_j} \Delta_j \tag{16}$$

where I_j is the set of links interfering with e_j and k should be minimized. This assignment achieves local, topology, dependent fairness [35].

This fairness approach is more appropriate for mobile networks, where it is difficult to have the global information of network topology and end-to-end requests. It can also be used with the scheduling algorithms in this paper, however in the rest of the paper, we assume that fairness is kept among end-to-end flows.

A. Linear Search Algorithm

We use an iterative procedure to find the minimum length schedule (MINIMIZE-SCHEDULE-LENGTH). The algorithm takes the link durations as input and has access to globally known information such as the conflict graph $G_c(E, C)$, slot duration T_s and the number of slots in the data sub-frame N_d.

Algorithm 1 MINIMIZE-SCHEDULE-LENGTH ($\boldsymbol{\Delta}$)

1: $N \leftarrow 0$
2: **while** $\nexists \boldsymbol{\sigma}$ for $\boldsymbol{\Delta}$ over N slots **do**
3: $\quad N \leftarrow N + 1$
4: **end while**
5: $\alpha \leftarrow N_d / \max\{N, N_d\}$ // this $N = N_{\min}$
6: $\forall e_j \in E: \quad \hat{\Delta}_i \leftarrow \lfloor \alpha \Delta_i \rfloor$
7: $\forall c_{ij} \in C : o_{ij} \leftarrow 1$ if $\sigma_i < \sigma_j$, otherwise $o_{ij} \leftarrow 0$
8: Use $\boldsymbol{o}$ to find $\hat{\boldsymbol{\sigma}}$ for $\hat{\boldsymbol{\Delta}}$ over N_d slots
9: $\forall e_j \in E : s_i \leftarrow T_s \sigma_i$, $\hat{d}_i \leftarrow T_s \hat{\Delta}_i$
10: **return** $S(\boldsymbol{s}, \hat{\boldsymbol{d}})$

The algorithm first searches for the minimum number of slots required to schedule all links (steps 1–4). In each iteration, the algorithm tries to find a schedule $\boldsymbol{\sigma}$ over N slots for link durations $\boldsymbol{\Delta}$ (step 2). If the schedule that fits into N slots cannot be found, N is incremented. The scheduling algorithms used in step 2 can either be the delay unaware algorithms from the last section or the delay aware algorithms shown in the next section. After finding the minimum N, the algorithm scales the link durations (step 6), resulting in scaled down link durations $\hat{\boldsymbol{\Delta}} = [\hat{\Delta}_1, \ldots, \hat{\Delta}_m]^T$.

Finally, the algorithm uses the scaled-down link durations, $\hat{\boldsymbol{\Delta}}$, to find a schedule that fits into N_d slots (steps 7–9). First, the algorithm uses the schedule $\boldsymbol{\sigma}$ to find the transmission order (step 7). Then, it uses the transmission order to find a scaled down schedule $\hat{\boldsymbol{\sigma}} = [\hat{\sigma}_1, \ldots, \hat{\sigma}_m]^T$ with the Bellman–Ford scheduler (step 8) and produces an actual schedule by multiplying each start time with the slot duration T_s (step 9). The algorithm returns a conflict-free TDMA schedule $S(\boldsymbol{s}, \hat{\boldsymbol{d}})$, where $\boldsymbol{s} = [s_1, \ldots, s_m]^T$ is the vector of activation times, corresponding to how long (in seconds) after the beginning of the data sub-frame links start their transmissions and $\hat{\boldsymbol{d}} = [\hat{d}_1, \ldots, \hat{d}_m]^T$ is the vector of link durations (in seconds), corresponding to the amount of time each link transmits when active.

Since the scaled down schedule, $S(\boldsymbol{s}, \hat{\boldsymbol{d}})$, fits into $T_d = N_d T_s$ seconds, it can be directly mapped into the frame. The full frame can be constructed from $S(\boldsymbol{s}, \hat{\boldsymbol{d}})$ by repeating the schedule in each frame, after the control sub-frame. We note that if control traffic is not grouped at the beginning of the frame, but individual messages are periodic with the period T_f as in 802.11s [3], there are N_d data slots in the frame with fixed position in each frame, so schedule $S(\boldsymbol{s}, \hat{\boldsymbol{d}})$ can also be mapped into the frame.

We show correctness of the algorithm in two steps. First, we show that N in step 5 is minimum. If a schedule exists for N slots, it also exists for $N + 1$ slots since we can always leave the last slot empty. Since we are incrementing N in steps of 1, N in step 5 is minimum. Second, we show that since we round down the link durations in step 6, step 8 always finds a schedule. By previous argument it is always possible to schedule links over $N_d \geq N_{\min}$ slots. We show the case of $N_{\min} > N_d$ by contradiction. Assume that link durations $\hat{\boldsymbol{\Delta}}$ cannot be scheduled over N_d slots, and so link durations $1/\alpha\hat{\boldsymbol{\Delta}}$ cannot

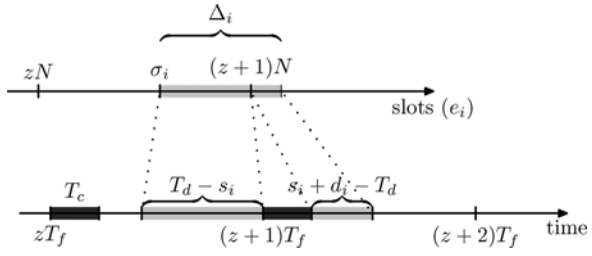

Fig. 6. Split transmissions.

be scheduled over $1/\alpha N_d = N_{\min}$ slots. Since for $\forall e_i \in E$: $\Delta_i \geq 1/\alpha\hat{\Delta}_i$ and $\boldsymbol{\Delta}$ can be scheduled over $N_{\min}$ slots we have a contradiction.

B. Minimizing Split Transmissions

In schedules found with the MINIMIZE-SCHEDULE-LENGTH algorithm, links transmit once before they are mapped into the frame. However, link transmissions may be split over two frames after the mapping, if all control slots are grouped at the beginning of the frame, as in the 802.16 mesh protocol [2]. Link e_i will be scheduled twice for transmission in the frame if $\sigma_i + \Delta_i > N$ (Fig. 6). The first transmission starts at time $T_c + s_i$ with the duration of $T_d - s_i$ seconds and the second transmission starts at the beginning of the next data sub-frame with the duration of $s_i + d_i - T_d$ seconds. So, our scheduling algorithm limits the number of transmissions by any link to at most two in a frame.

The number of split transmissions can be minimized in at most N steps, after step 8 of the algorithm. The algorithm that minimizes the number of split transmissions shifts the generalized schedule $\boldsymbol{\pi}^*$ in iterations. In each iteration a new generalized schedule $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_m]^T$ is obtained from the activation time in the previous iteration by increasing each π_i by 1. Then, the algorithm normalizes the shifted generalized schedule with the modulo operation and counts the number of links with split transmissions. The algorithm picks the normalized schedule, which has the fewest links with split transmissions. This algorithm also works when control slots are not grouped together.

V. TDMA DELAY AWARE SCHEDULING

In this section, we devise scheduling algorithms, which account for scheduling delay. Scheduling delay occurs when an outgoing link on a mesh node is scheduled to transmit before an incoming link in the path of a packet. First, we show that the scheduling delay is directly related to the transmission order in the frame. Then, we formulate a {0, 1}-integer program that finds the transmission order for which the maximum scheduling delay among all paths is minimized. With this min-max optimization in step 2 of the MINIMIZE-SCHEDULE-LENGTH algorithm, the algorithm finds the minimum number of slots required to schedule all links, $N_{\min}$, and a transmission order with the min-max scheduling delay property.

Finally, we propose a polynomial time algorithm for networks, which are managed as overlay trees. This algorithm first finds a transmission order, which limits the round-trip delay on all paths to one frame and uses the Bellman–Ford scheduling algorithm to find schedules. With this scheduling algorithm in step 2 of MINIMIZE-SCHEDULE-LENGTH, the algorithm searches for the minimum length schedule over schedules restricted to have the minimum, one frame, scheduling delay.

Even though the two algorithms presented in this section are based on the MINIMIZE-SCHEDULE-LENGTH algorithm, they achieve goals at the opposite sides of the delay/bandwidth spectrum. The first algorithm maximizes bandwidth first and then finds the minimum delay. The second algorithm minimizes delay first and finds the maximum bandwidth subject to the minimum delay constraint.

A. End-to-End Scheduling Delay

Since TDMA networks are stop-and-go queueing systems, there is no queueing delay and end-to-end delay comes exclusively from scheduling delay. Consider a path $P_l = \{\ldots, e_i, e_j, \ldots\}$ with $e_i \in \{v_l\}^-$ and $e_j \in \{v_l\}^+$, where we use the notation that $e_i \in \{v_l\}^-$ are incoming links of v_l and $e_j \in \{v_l\}^+$ are the outgoing links of v_l (Fig. 7). Every frame, link e_i sends $\hat{r}_j T_f$ bits to node v_l, which are transmitted by the next transmission of link e_j. At every router in the path, data also only wait for the difference in time until the transmission of the next link in the path. Since data is only buffered until the next transmission of the outgoing link, there is no queueing delay and the end-to-end delay is determined by the TDMA schedule.

We show how single-hop scheduling delay occurs in Fig. 7 for two different schedules $S_a(\boldsymbol{s}, \boldsymbol{d})$ (Fig. 7(a)) and $S_b(\boldsymbol{s}, \boldsymbol{d})$ (Fig. 7(b)). We align the time axis to the beginning of the data sub-frame to simplify exposition. In schedule $S_a(\boldsymbol{s}, \boldsymbol{d})$, $s_j > s_i$, so the first bit of a packet sent from v_k to v_m experiences the scheduling delay of $s_j - s_i + t_p$, where t_p is the propagation delay. In schedule $S_b(\boldsymbol{s}, \boldsymbol{d})$, $s_i > s_j$, so when the packet arrives at v_l it has to wait for e_j to transmit in the next frame. In this case, the scheduling delay from v_k to v_m is $s_j - s_i + T_f + t_p$, where T_f is frame duration. In the sequel, we drop the propagation delay t_p since it is independent of scheduling.

The single-hop scheduling delay is directly related to the transmission order for the schedule. For TDMA schedule $S_a(\boldsymbol{s}, \boldsymbol{d})$, transmission order $o_{ij} = 0$, while for TDMA schedule $S_b(\boldsymbol{s}, \boldsymbol{d})$, transmission order $o_{ij} = 1$. We conclude that if the path traverses e_i first and e_j second, i.e., conflict c_{ij} is traversed in the positive direction, scheduling delay is

$$t_{ij} = s_j - s_i + o_{ij} T_f. \tag{17}$$

On the other hand, if the conflict is traversed in the opposite direction, i.e., conflict is c_{ji}, it can be shown by a similar argument that the single-hop scheduling scheduling delay is

$$t_{ij} = s_j - s_i + (1 - o_{ji}) T_f. \tag{18}$$

In the rest of the paper, we develop scheduling algorithms that minimize the scheduling delay on round-trip paths. The round-trip delay is important for applications that use TCP as the transport protocol since the throughput of TCP is inversely proportional to the round-trip path delay [36]. Schedulers that minimize scheduling delay of unicast end-to-end connections

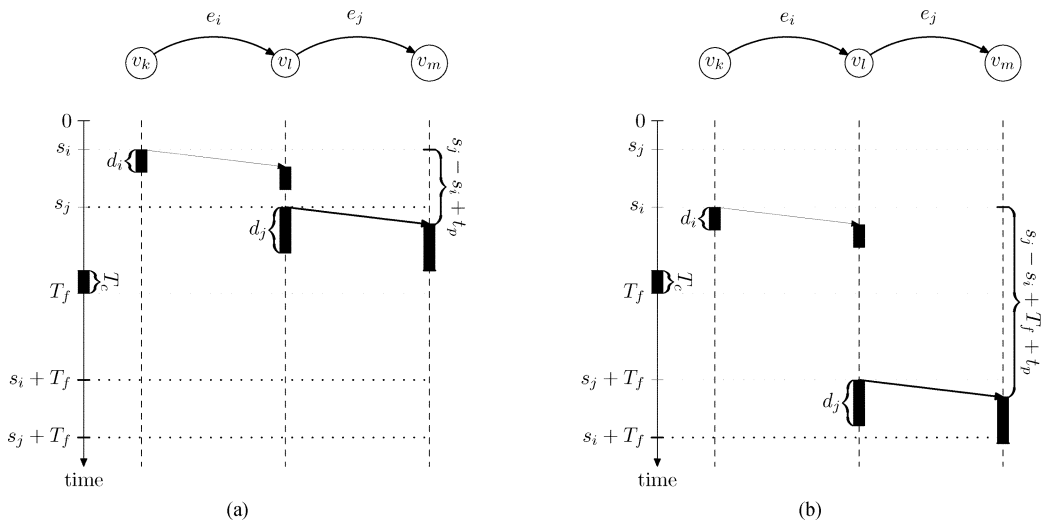

Fig. 7. Single hop delay. (a) $s_j > s_i$. (b) $s_j < s_i$.

such as Voice-over-IP traffic can be derived similarly to the way we derive the scheduler for round-trip path TDMA delay [37].

A round-trip path in the topology graph corresponds to a cycle in the conflict graph. The cycle in the conflict graph can be obtained by finding the conflicts needed to visit the vertexes of the conflict graph $G(E, C)$ listed in a path. For example, the round-trip path $P_1 = (e_1, e_3, e_6, e_{10}, e_4, e_2)$ in Fig. 3(a) corresponds to the cycle $\theta_1 = \{c_{1,3}, c_{3,6}, c_{6,10}, c_{4,10}, c_{2,4}, c_{1,2}\}$, marked in Fig. 3(b). In the same topology, the round-trip path $P_2 = (e_1, e_3, e_5, e_8, e_9, e_7, e_4, e_2)$ in Fig. 3(a) corresponds to the cycle $\theta_2 = \{c_{1,3}, c_{3,5}, c_{5,8}, c_{8,9}, c_{7,9}, c_{4,7}, c_{2,4}, c_{1,2}\}$, also marked in Fig. 3(b).

We find the end-to-end scheduling delay for a path P_l by finding the delay incurred while traversing the corresponding cycle θ_l in the conflict graph:

$$D_l = \sum_{c_{ij} \in \{\theta_l\}^+} (s_j - s_i + o_{ij} T_f) - \sum_{c_{ji} \in \{\theta_l\}^-} (s_i - s_j + o_{ji} T_f - T_f). \tag{19}$$

where $\{\theta_{\mathcal{P}}\}^+$ are the conflicts traversed in their direction and $\{\theta_{\mathcal{P}}\}^-$ are the conflicts traversed in their opposite direction.

Roundtrip scheduling delay depends on the transmission order only:

$$D_l = \sum_{c_{ij} \in \{\theta_l\}^+} o_{ij} T_f + \sum_{c_{ji} \in \{\theta_l\}^-} \Big(1 - o_{ji}\Big) T_f \tag{20}$$

where starting times cancel out. For example, the summation over difference terms for cycle θ_1 in Fig. 3(b) is

$$(s_3 - s_1) + (s_6 - s_3) + \cdots - (s_2 - s_1) = 0. \tag{21}$$

This cancellation in is a well known graph property [31, pp. 174].

The delay calculated by (20) is the worst case round-trip delay, since it measures the delay from the time the first link on the path transmits to the time the first link transmits again. It is also possible to calculate delay more precisely, by accounting for the delay from beginning of the first transmission on the path and to end of the last transmission on the path. This delay can be calculated using the unicast delay formula [37], which is very similar to (20). However, in this paper we restrict ourselves to the worst case round-trip delay.

In the rest of the paper, we use vector notation to represent cycles to make formulae more compact. A cycle θ_l in the conflict graph is defined by the r-dimensional vector $\boldsymbol{\theta}_l = [\ldots, \theta_{ij}^{(l)}, \ldots]^T$, where $\theta_{ij}^{(l)} = 1$ if $c_{ij} \in \{\theta_l\}^+$, $\theta_{ij}^{(l)} = -1$ if $c_{ij} \in \{\theta_l\}^-$ and $\theta_{ij}^{(l)} = 0$ if $c_{ij} \notin \theta_l$. For example, the cycle θ_1 in Fig. 3, corresponds to the vector $\boldsymbol{\theta}_1 \in \{0, 1\}^{33}$ where $\boldsymbol{\theta}_{1,3}^{(1)} = \boldsymbol{\theta}_{3,8}^{(1)} = \boldsymbol{\theta}_{6,10}^{(1)} = 1$, $\boldsymbol{\theta}_{4,10}^{(1)} = \boldsymbol{\theta}_{2,4}^{(1)} = \boldsymbol{\theta}_{1,2}^{(1)} = -1$ and all other entries in the vector are 0. In vector notation, scheduling delay on a round-trip path is

$$D_l = \left[\boldsymbol{\theta}_l^T \boldsymbol{o} + K_l\right] T_f \tag{22}$$

where $K_l = \sum_{c_{ji} \in \{\theta_l\}^-} 1$ is a constant for the cycle.

B. Min-Max TDMA Delay Scheduling

We formulate a {0, 1}-integer program that finds a transmission order that minimizes the end-to-end scheduling delay on the set of round-trip paths, while ensuring that a feasible virtual schedule exists over N slots. An objective function that finds such schedules is the min-max delay defined as $\min \max_{P_l \in \mathcal{P}_R} D_l$. We combine the formula for the scheduling delay (20) with the polyhedron of feasible transmission orders to formulate a {0, 1}-integer program that finds a schedule with the min-max delay for a fixed N:

$$\min_{\boldsymbol{\pi}, \boldsymbol{O}, t} \quad t \tag{23a}$$

$$\text{s.t} \quad \Big[\boldsymbol{\theta}_l^T \boldsymbol{o} + K_l\Big] T_f \le t, \quad P_l \in \mathcal{P}_R \tag{23b}$$

$$\boldsymbol{l} \le \boldsymbol{C}^T \boldsymbol{\pi} - N\boldsymbol{o} \le N\mathbf{1} - \boldsymbol{u} \tag{23c}$$

$$\boldsymbol{\pi} \in \mathbb{Z}^m, \boldsymbol{o} \in \{0, 1\}^r, t \ge 0 \tag{23d}$$

Constraints (23b) ensure that no path has a delay larger than t, while t is minimized; this achieves the goal of finding the

min-max optimum. The other constraints (23c) define the polyhedron of conflict-free schedules for a fixed N.

We use the max-min optimization (23) in step 2 of the MINIMIZE-SCHEDULE-LENGTH algorithm. In the sequel, we refer to this refinement of the MINIMIZE-SCHEDULE-LENGTH algorithm as Algorithm-MM. Since Algorithm-MM finds the minimum length schedule, the final schedule found by the algorithm is min-max delay optimum schedule with the minimum length.

C. Single Frame TDMA Scheduling for Tree Overlays

The min-max formulation is hard to solve because it requires a search for $\boldsymbol{o}$ over the space of all $\{0,1\}^r$ vectors, which can be performed with the standard branch-and-bound technique [38]. In this section, we take advantage of the fact that many multi-hop wireless networks are managed as overlay tree topologies and propose a polynomial algorithm for overlay tree topologies that produces schedules with the maximum scheduling delay of one frame on all round-trip paths. Since the two most important and common examples of wireless multi-hop networks, managed as *overlay* tree topologies, are sensor networks and mesh networks, this algorithm is applicable in a wide array of scenarios. This algorithm is used in step 2 of the MINIMIZE-SCHEDULE-LENGTH algorithm. In the sequel, we refer to this refinement of MINIMIZE-SCHEDULE-LENGTH as Algorithm-TH.

The algorithm first finds a transmission order, which produces round-trip scheduling delay of one frame on all paths, by giving each link e_i rank R_i, which indicates the preferred order of transmissions. Initially, the algorithm sets the rank of all the links to zero. The algorithm then examines each round-trip path, link-by-link, and assigns a rank to each link as a function of the distance from the root of the routing tree. We assume that the base-station is $v_1 \in V$. For links in a round-trip path $P_l = \{e_i, \ldots, e_k, e_l, \ldots e_j\}$, where $e_i \in \{v_1\}^+$ and $e_j \in \{v_1\}^-$, the rank is assigned as follows:

$$R_k = \max\{R_k, R_l + 1\}, \text{ if } e_k \text{ follows } e_l \text{ on the path.} \quad (24)$$

We note that the distance of the link is defined as its placement on the round-trip path and not its topological distance from the root of the tree. For the example in Fig. 3(a), we have $R_1 = 0, R_3 = 1, R_5 = R_6 = 2, R_8 = R_{10} = 3, R_9 = 4, R_7 = 5, R_4 = 6$ and $R_2 = 7$.

Given the ranking, the transmission order $\boldsymbol{o}$ is assigned with

$$o_{ij} = \begin{cases} 0, & \text{if } R_j \geq R_i \\ 1, & \text{otherwise.} \end{cases} \quad (25)$$

Finally, the algorithm finds a schedule with the Bellman–Ford algorithm.

Proposition 2: If the schedule is derived with Algorithm-TH, then for all round-trip paths P_l:

$$D_l = T_f. \quad (26)$$

Proof: Consider a round-trip path $P_l = \{e_i, \ldots e_j\}$, where $e_i \in \{v_1\}^+$ and $e_j \in \{v_1\}^-$, and its corresponding cycle θ_l in the conflict graph. If we consider the delay, (20), without the last conflict, which connects e_i and e_j, we observe that both sums add up to 0. If the last conflict is c_{ji}, then, since $R_j > R_i$, $D_l = o_{ji} = 1$. On the other hand, if the last conflict is c_{ij}, then $D_l = 1 - o_{ij} = 1$. ■

We note that the ranking function does not allow spatial re-use between links on the same path since it orders all links on the path to transmit in a sequence. However, the ranking function still allows spatial re-use for links on different paths.

This algorithm can also be used as a heuristic for networks, which are not managed as tree overlays. However, in that case the delay may be larger than one frame. In the rest of the paper, we only consider networks, which are managed as overlay trees.

D. Modulo Operation Preserves Delay Properties

Both algorithm Algorithm-MM and Algorithm-TH, find a generalized schedule in step 2, which has the transmission order with some delay properties. However, these schedules are also normalized with the modulo operation, and during this process the transmission order may change. We now show that despite the change, the final normalized schedule has the same scheduling delay as the generalized schedule.

Suppose that Algorithm-MM or Algorithm-TH find a generalized schedule in their step 2, with a transmission order $\boldsymbol{o}$ and a generalized schedule $\boldsymbol{\pi}$ corresponding to a virtual schedule. When this schedule is normalized with the modulo operation to obtain $\boldsymbol{\sigma}$, the final transmission order $\boldsymbol{o}^*$ may be different from $\boldsymbol{o}$. The two orders may be different since each generalized activation time π_i is related to the normalized activation time with $\pi_i = \sigma_i + z_i N$ and for a conflict c_{ij}, z_i may be different from z_j, while the feasibility conditions (11) are still true for σ_i and σ_j. If we substitute $\pi_i = \sigma_i + z_i N$ into (11), we get

$$\Delta_i - o^*_{ij} N \leq \sigma_j - \sigma_i \leq N - \Delta_j - o^*_{ij} N \quad (27)$$

where

$$o^*_{ij} = z_j - z_i + o_{ij}, \quad \forall c_{ij} \in C. \quad (28)$$

So, the change in the relative transmission order may happen if $o_{ij} = 0$ and $z_j - z_i = 1$, or if $o_{ij} = 1$ and $z_j - z_i = -1$.

Even though the order of transmissions is changed, the delay remains the same. Using (28), the single-hop delay in the positive direction of c_{ij} is

$$t_{ij} = s_j - s_i + o^*_{ij} T_f = s_j - s_i + (z_j - z_i) T_f + o_{ij} T_f. \quad (29)$$

The delay in the opposite direction has a similar difference term with the z variables. Since the difference terms cancel out, (19), the delay stays the same even if the transmission order is changed with the modulo operation.

VI. SIMULATION RESULTS

In this section we compare the performance of scheduling algorithms from the previous section when they are applied to 802.16 mesh networks [2]. We do not get into the details of the 802.16 protocol in this paper; more information about the protocol and 802.16 scheduling algorithms are available in the standard [2] and our survey of scheduling algorithms for 802.16 mesh networks [39].

The 802.16 mesh protocol specifies two scheduling *protocols* for assignment of link bandwidths (slots): *centralized* and *decentralized* scheduling protocols. In the centralized scheduling protocol, nodes request end-to-end bandwidth from the base-station (BS), which may be multiple hops away. In turn, the BS responds with a network wide schedule, which is determined based on the requests. In the decentralized scheduling protocol, nodes negotiate pairwise bandwidth assignments. The centralized scheduling protocol can be used to establish network-wide end-to-end QoS, while the decentralized scheduling protocol is not expected to establish end-to-end QoS [40].

In all our examples, we set the 802.16 frame duration to 10ms giving a total of 800 Orthogonal Frequency Division Multiplexing (OFDM) symbols in each frame. We set the size of the control sub-frame to 70 OFDM symbols, leaving 730 symbols for the data sub-frame. The 802.16 mesh protocol mandates the assignment of data sub-frame symbols to links in terms of *transmission opportunities*. With our parameters transmission opportunities are 3 OFDM symbols long, for a total of 243 transmission opportunities in the data sub-frame. Each link transmission has an overhead of $h = 3\,\text{slots}$. In order to make our scheduling approaches consistent with 802.16 mesh protocol, we equate transmission opportunities to TDMA slots. In the sequel, we fix the percentage of the frame used for centralized scheduling to 80%, so $N_d = 194\,\text{slots}$.

Our TDMA scheduling approach fits into 802.16 centralized scheduling, without the need to change any protocol packets. First, the base-station uses Algorithm-MM or Algorithm-TH to find the transmission order (link rankings) and the scaling factor α that maximizes the concurrent end-to-end throughput of mesh nodes. Then, the base-station multicasts the ranking and the scaled-down link bandwidths. Second, the nodes use the ranking and the link bandwidths they receive from the base-station and their knowledge of the network topology to calculate the network wide TDMA schedule with the Bellman–Ford algorithm. We assume that the network wide topology is known to all nodes. If this assumption is not true, the nodes can use the distributed version of the scheduling algorithm [22], which relies only on local topology information to find network wide TDMA schedules.

We use two performance indices to measure the performance of the algorithms. The first performance index shows the ratio between the requested end-to-end rates and the end-to-end rates achieved with a particular 802.16 schedule. For this measure, we calculate the average percentage of achieved bandwidth over the active sources in the network:

$$\bar{\alpha} = \frac{1}{|n|}\sum_{i=1}^{n}\frac{\hat{g}_i^{\text{up}}}{g_i^{\text{up}}} + \frac{1}{|n|}\sum_{i=1}^{n}\frac{\hat{g}_i^{\text{down}}}{g_i^{\text{down}}} \tag{30}$$

where there are n active mesh nodes requesting uplink and downlink bandwidth, g_i^{up} and g_i^{down} are the uplink and downlink bandwidth requested by mesh node i, respectively, $\hat{g}_i^{\text{up}}$ and $\hat{g}_i^{\text{down}}$ are the uplink and downlink bandwidth granted to mesh node i, respectively. The granted end-to-end bandwidth is obtained by first finding the actual rate on each link resulting from the schedule with (3) and then finding the end-to-end rate with (4). The second performance index is the maximum round-trip scheduling delay:

$$D_{\max} = \max_{P_l \in \mathcal{P}} D_l \tag{31}$$

where $\mathcal{P}$ is the set of round-trip paths in the network.

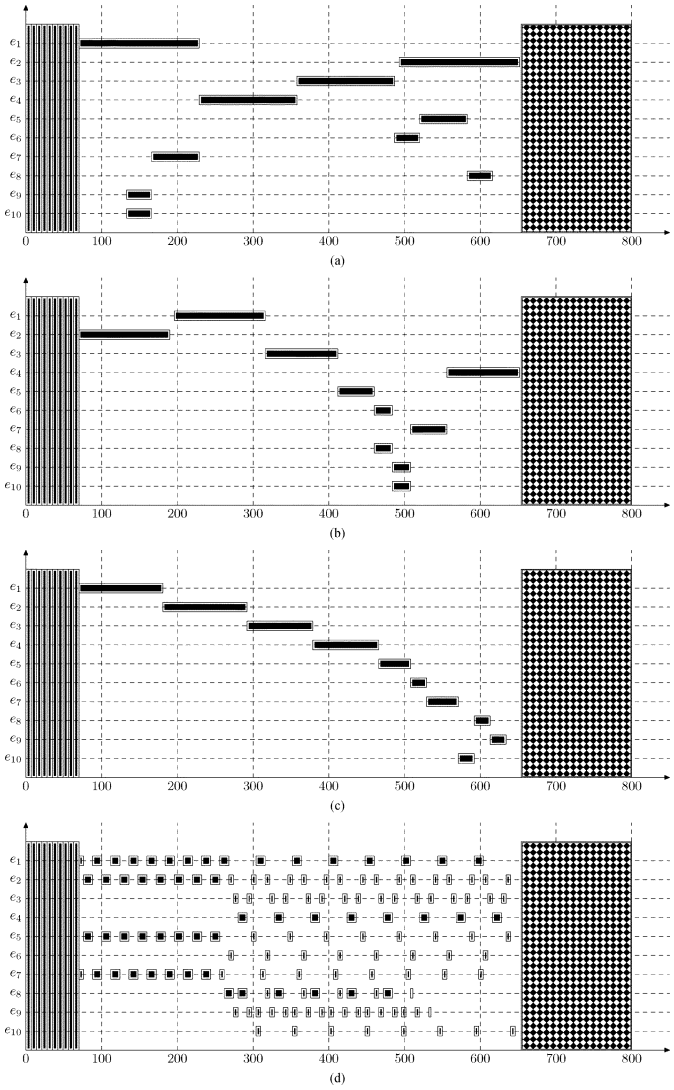

Fig. 8. 802.16 schedules for the small topology (Fig. 3). (a) Algorithm-MM. (b) Algorithm-TH. (c) IEEE 802.16.

A. Example 1: Small Topology

We examine schedules applicable to the topology shown Fig. 3(a). All nodes except the base station (v_1) act as sources with fixed uplink and downlink bandwidth demands of 900kbps. We use Algorithm-MM (Fig. 8(a)), Algorithm-TH (Fig. 8(b)), the 802.16 centralized scheduling algorithm (Fig. 8(c)), and the graph coloring algorithm [15] (Fig. 8(d)).

Fig. 8(c) shows the schedule obtained with the centralized scheduling algorithm proposed in the 802.16 standard [2]. The 802.16 standard proposes this algorithm as an example of how to allocate transmission opportunities, however the standard does not mandate its use. The first step in the 802.16 algorithm is to

TABLE I
ALGORITHM PERFORMANCE ON THE SMALL TOPOLOGY (FIG. 3)

Algorithm	$D_{\max}$	$\bar{\alpha}$
Algorithm-MM	20.0ms	32.5%
Algorithm-TH	10.0ms	20.9%
IEEE 802.16	40.0ms	20.9%
Graph Colouring	40.0ms	18.3%

find a link ranking during a breadth-first traversal of the routing tree. With the breadth first search the ranking on the topology becomes $R_1 = R_2 = 0$, $R_3 = R_4 = 1$, $R_5 = R_6 = R_7 = R_{10} = 2$ and $R_8 = R_9 = 3$. Second, the algorithm assigns transmission opportunities to links in the order of their rank. The link with the lowest rank is assigned transmission opportunities at the beginning of the data sub-frame. The link with the next highest rank is assigned the subsequent transmission opportunities and so on, until all links are scheduled. If the total number of assigned transmission opportunities is larger than the number of transmission opportunities reserved for centralized scheduling, the algorithm scales down the link transmissions to fit in the frame.

Fig. 8(d) shows the schedule obtained with a vertex coloring scheduling algorithm [15]. This algorithm assigns transmission opportunities in rounds. In each round, the algorithm assigns one transmission opportunity to the link with the highest unsatisfied demand and all other links that do not conflict with it. Rounds are repeated until either there are no free transmission opportunities left or demands of all links are satisfied. A similar algorithm is proposed in [16], with the difference in how the nodes are ranked in each iteration. In the rest of the paper, we only show the results for the algorithm proposed in [15]. In order to make sure that schedules assigned with this algorithm do not result in transmissions that carry no data, we make the minimum transmission duration two transmission opportunities.

We summarize the results of the algorithm performance in Table I. The graph coloring algorithm ("Graph Colouring") has the worst performance of all four scheduling approaches because multiple link transmissions in each frame cause a large amount of overhead. The delay of the algorithm is also large since the algorithm ranks links in terms of their required bandwidth, rather than to minimize delay. The 802.16 algorithm ("IEEE 802.16") works better than the graph coloring algorithm, but it achieves lower end-to-end bandwidth than the optimal min-max algorithm or the tree heuristic algorithm since it does not use spatial re-use. The 802.16 algorithm also introduces a large delay because it ranks the links with the breadth-first search of the routing tree.

Algorithm-TH and Algorithm-MM result in the best performance. Algorithm-TH has a bandwidth assignment that is 30% worse than Algorithm-MM, but it also has TDMA delay performance that is half of the delay of Algorithm-MM.

B. Example 2: Chain Topology

In this section, we examine performance of scheduling algorithms on chain topologies. In each scenario, we create a chain of mesh nodes where one of the end nodes is the base-station and all other nodes in the topology request uplink and downlink end-to-end rates of 2.2Mbps. We chose 2.2Mbps as the requested rate because, with our physical layer parameters, this is close to the maximum possible with just two nodes in the chain.

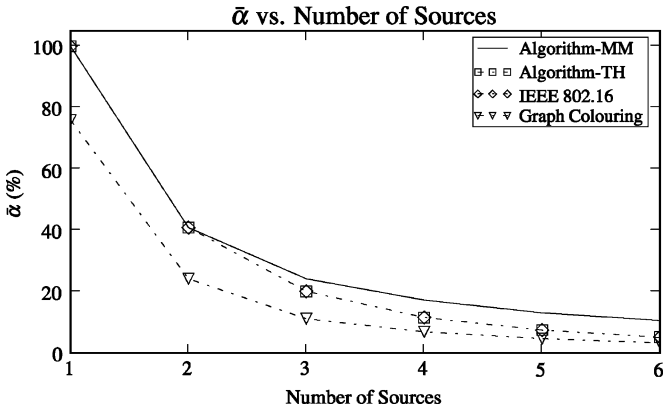

Fig. 9. Percentage of achieved bandwidth for the chain topology.

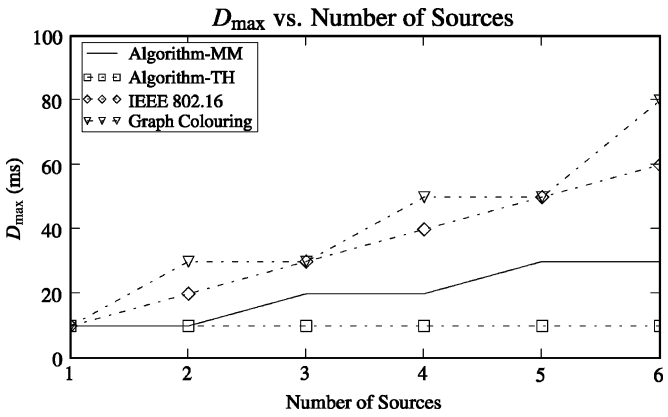

Fig. 10. Maximum delay for the chain topology.

Fig. 9 shows the performance of different algorithms in terms of bandwidth as the number of sources – size of the chain – increases (the naming is the same as in Table I). Algorithm-MM has the best performance in terms of achieveable bandwidth. Algorithm-TH and the 802.16 scheduling algorithm have the same, lower achievable bandwidth. The reason for this is that transmission orders found with the tree heuristic do not allow spatial re-use among links on the same path. However, we have seen in the example of the simple topology (Fig. 8(b)) that in general the transmission orders found with the tree heuristic do allow spatial re-use among links on the different paths. Finally, despite taking advantage of spatial re-use, the graph coloring algorithm has the worst performance due to multiple transmissions in the frame. Even with just one source, when it is possible to schedule all requested end-to-end bandwidths, the coloring scheme only achieves 75% of the requested end-to-end bandwidths.

Fig. 10 shows the maximum delay as the number of sources increases. Algorithm-TH has a constant one frame TDMA delay, consistent with the theory in the last section. As expected, Algorithm-MM has larger delay than Algorithm-TH. The 802.16 algorithm introduces an even larger delay because it ranks the links with the breadth-first search of the routing

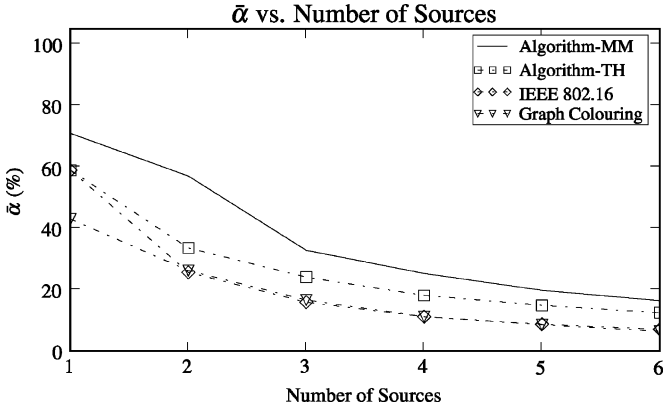

Fig. 11. Percentage of achieved bandwidth for the mesh topology.

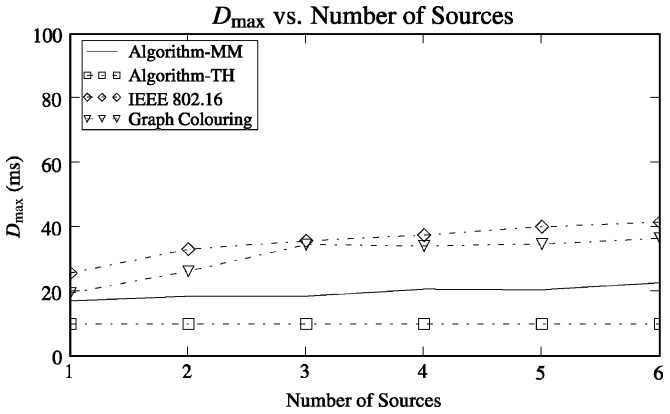

Fig. 12. Maximum delay for the mesh topology.

tree. The graph coloring approach has the maximum delay, approximately twice as large as the delay of Algorithm-MM.

C. Example 3: 5 × 3 Mesh Topology With Random Sources

In this section we examine performance of scheduling algorithms on a 5×3 grid. We place the base-station in the centre of the topology and pick sources at random. After picking the sources, we find paths to the base-station with the minimum spanning tree algorithm. Each source requests an end-to-end rate of 2.2Mbps. We repeat the selection of sources 50 times. We plot the results for $\bar{\alpha}$ averaged over all 50 scenarios in Fig. 11 and the results for $D_{\max}$ averaged over all 50 scenarios in Fig. 12.

We note again that despite spatial re-use, the graph coloring algorithm achieves at most as much bandwidth as the 802.16 scheduling algorithm that does not use spatial reuse (Fig. 11). The delay of the graph coloring algorithm is better than the delay of the 802.16 algorithm, however it is still significantly larger than the delay of Algorithm-MM or Algorithm-TH (Fig. 12).

Finally, we note that the bandwidth performance of Algorithm-TH improves, compared to the performance of Algorithm-MM, as the number of sources increases. End-to-end bandwidths allocated with Algorithm-TH come within 5% of the end-to-end bandwidths allocated with Algorithm-MM. The reason for the improvement, is that as the number of sources increases, so does the number of *unique* paths. With the increase in the number of unique paths, Algorithm-TH uses more spatial re-use and comes closer to the performance of Algorithm-MM.

VII. Conclusion

This paper introduces TDMA delay aware scheduling for multi-hop wireless networks. First, we have shown that the delay unaware TDMA scheduling problem can be solved in two parts. In the first part, a relative transmission order of the links (precedence) is found. In the second part, the relative transmission order is used together with the conflict graph as the input to a modified Bellman–Ford algorithm, which can find a feasible schedule in polynomial time. For example, TDMA schedules can be found with the *distributed* Bellman–Ford algorithm that does not require the direct knowledge of the full network topology.

Second, we have shown that the precedence is related directly to scheduling delay in the network, and propose methods to find transmission orders with good TDMA delay properties. We devise a polynomial time algorithm that finds one-frame scheduling delay transmission orders on overlay tree topologies. Since the two most important and common examples of wireless multi-hop networks using overlay tree topologies are sensor networks and mesh networks, this algorithm is applicable in a wide array of scenarios.

We have used simulations to compare the algorithms in this paper applied to 802.16 mesh networks with other algorithms proposed for these networks. We have shown that our approach has better performance for two reasons. First, it allows spatial re-use so total bandwidth provided by the network is increased. Second, it limits the number of transmissions each link makes in the frame, so it increases the efficiency and end-to-end bandwidth.

Acknowledgment

The authors would like to thank the anonymous IEEE INFOCOM and IEEE/ACM TRANSACTIONS ON NETWORKING (ToN) reviewers who gave significant suggestions for improving the paper. The authors are especially grateful to the ToN reviewer who suggested the simple proof of correctness for the MINIMIZE-SCHEDULE-LENGTH algorithm, which is presented in the paper.

References

[1] P. Djukic and S. Valaee, "Link scheduling for minimum delay in spatial re-use TDMA," in *Proc. IEEE INFOCOM*, 2007, pp. 28–36.

[2] *IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems*, 802.16, 2004.

[3] *IEEE P802.11s/D1.01, Draft STANDARD for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks- Specific Requirements- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment: ESS Mesh Networking*, 802.11s, 2006.

[4] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[5] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless Networks*, vol. 11, pp. 4471–487, 2005.

[6] S. Ramanathan, "A unified framework and algorithm for channel assignment in wireless networks," *Wireless Networks*, vol. 5, no. 2, pp. 81–94, Mar. 1999.

[7] S. Ramanathan and E. L. Lloyd, "Scheduling algorithms for multihop radio networks," *IEEE/ACM Trans. Netw.*, vol. 1, no. 2, pp. 166–177, Apr. 1993.
[8] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 910–917, Sep. 1988.
[9] T. Salonidis and L. Tassiulas, "Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks," in *Proc. ACM MobiHoc*, 2005, pp. 145–156.
[10] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless mesh networks with orthogonal channels," *IEEE/ACM Trans. Netw.*, vol. 13, no. 4, pp. 868–880, Aug. 2005.
[11] G. Sharma, R. Mazumdar, and N. Shroff, "On the complexity of scheduling in wireless networks," in *Proc. ACM Mobicom*, 2006, pp. 227–238.
[12] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in sensor networks: Distributed edge coloring revisited," in *Proc. IEEE INFOCOM*, 2005, vol. 4, pp. 2492–2501.
[13] S. Sarkar and K. Kar, "Achieving 2/3 throughput approximation with sequential maximal scheduling under primary interference contraints," in *Proc. 44th Annu. Allerton Conf. Communication, Control and Computing*, 2006, online.
[14] M. Sanchez, J. Zander, and T. Giles, "Combined routing and scheduling for spatial TDMA in multihop ad hoc networks," in *Proc. WPMC*, 2003, vol. 2, pp. 781–785.
[15] H.-Y. Wei, S. Ganguly, R. Izmailov, and Z. Haas, "Interference-aware IEEE 802.16 WiMax mesh networks," in *Proc. IEEE VTC Spring'05*, 2005, vol. 5, pp. 3102–3106.
[16] B. Han, W. Jia, and L. Lin, "Performance evaluation of scheduling in IEEE 802.16 based wireless mesh networks," *Comput. Commun.*, vol. 30, pp. 782–792, 2007.
[17] S. J. Golestani, "A framing strategy for congestion management," *IEEE J. Sel. Areas Commun.*, vol. 9, no. 7, pp. 1064–1077, Sep. 1991.
[18] R. Nelson and L. Kleinrock, "Spatial TDMA: A collision-free multihop channel access protocol," *IEEE Trans. Commun.*, vol. COM-33, pp. 934–944, Sep. 1985.
[19] N. B. Salem and J. Hubaux, "A fair scheduling for wireless mesh networks," in *Proc. Wimesh*, 2005, http://www.cs.ucdavis.edu/~prasant/wimesh.
[20] C. E. Shannon, "A theorem on coloring the lines of a network," *J. Math. Phys.*, vol. 28, pp. 148–151, 1949.
[21] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
[22] P. Djukic and S. Valaee, "Distributed link scheduling for TDMA mesh networks," in *Proc. IEEE ICC*, 2007, pp. 3823–3828.
[23] G. Narlikar, G. Wilfong, and L. Zhang, "Designing multihop wireless backhaul networks with delay guarantees," in *Proc. IEEE INFOCOM*, 2006, pp. 1–12.
[24] A. K. Parekh and R. G. Gallager, "A generalized proessor sharing approach to flow control in integrated services networks: The single node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, Jun. 1993.
[25] L. Georgiadis, R. Guerin, V. Peris, and K. N. Sivarajan, "Efficient network QoS provisioning based on per node traffic shaping," *IEEE/ACM Trans. Netw.*, vol. 4, no. 4, pp. 482–501, Aug. 1996.
[26] *IEEE Standard for Local and Metropolitan Area Networks Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 802.11, 1999.
[27] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.
[28] L. Badia, M. Mastrogiovanni, C. Petrioli, S. Stefanakos, and M. Zorzi, "An optimization framework for joint sensor deployment, link scheduling and routing in underwater sensor networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 11, no. 4, pp. 44–56, Oct. 2007, Special section on ACM WUWNet 2006.
[29] P. Serafini and W. Ukovich, "A mathematical model for periodic scheduling problems," *SIAM J. Discrete Math.*, vol. 2, no. 4, pp. 550–581, Nov. 1989.
[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.
[31] R. T. Rockafellar, *Network Flows and Monotropic Optimization*. New York: Wiley, 1984.
[32] P. Djukic, "Scheduling algorithms for TDMA wireless multihop networks," Ph.D. dissertation, Univ. of Toronto, Toronto, ON, Canada, Mar. 2008 [Online]. Available: http://www.wirlab.utoronto.ca/ref/text/DjukicPHDThesis.pdf
[33] G. J. Minty, "A theorem on n-coloring the points of a linear graph," *Amer. Math. Month.*, vol. 67, pp. 623–624, 1962.
[34] Z. Tuza, "Graph coloring in linear time," *J. Combinator. Theory*, vol. 55-B, pp. 236–243, 1992.
[35] H. Lua, S. Lu, V. Bharghavan, J. Cheng, and G. Zhong, "A packet scheduling approach to QoS support in multihop wireless networks," *ACM J. Mobile Netw. Applicat.*, vol. 9, no. 3, pp. 193–206, Jun. 2004.
[36] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, Apr. 2000.
[37] P. Djukic and S. Valaee, "Quality-of-service provisioning in multiservice TDMA mesh networks," in *Proc. 20th Int. Teletraffic Congr.*, 2007, pp. 841–852.
[38] L. A. Wolsey, *Integer Programming*. New York: Wiley-Interscience, 1998.
[39] P. Djukic and S. Valaee, "Scheduling algorithms for 802.16 mesh networks," in *WiMax/MobileFi: Advanced Research and Technology*, Y. Xiao, Ed. New York: Auerbach, 2007, pp. 267–288.
[40] M. Cao, W. Ma, Q. Zhang, X. Wang, and W. Zhu, "Modeling and performance analysis of the distributed scheduler in IEEE 802.16 mesh mode," in *Proc. ACM MobiHoc*, 2005, pp. 78–89.

Petar Djukic (S'01–M'08) received the B.A.Sc., M.A.Sc., and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada, in 1999, 2002, and 2008, respectively.

His research interests are in wireless multi-hop scheduling and resource allocation, and testbed implementations of new wireless MAC protocols. From 1999 to 2001, he worked as a Software Designer in Ottawa, Canada. From 2007 to 2008, he was a Postdoctoral Researcher at the University of California, Davis.

Shahrokh Valaee (S'88–M'00–SM'02) was born in Tabriz, Iran. He received the B.Sc. and M.Sc. degrees from Tehran University, Tehran, Iran, and the Ph.D. degree from McGill University, Montreal, Canada, all in electrical engineering.

From 1994 to 1995, he was a Research Associate at INRS Telecom, University of Quebec, Montreal, Canada. From 1996 to 2001, he was an Assistant Professor in the Department of Electrical Engineering, Tarbiat Modares University, Tehran, Iran, and in the Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran. During this period, he was also a consultant to Iran Telecommunications Research Center. Since September 2001, he has been an Associate Professor in the Edward S. Rogers Sr. Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, and holds the Nortel Institute Junior Chair of Communication Networks. His current research interests are in wireless ad hoc, sensor, and cellular networks.

Dr. Valaee is the Co-Chair for the Wireless Communications Symposium of IEEE GLOBECOM 2006, the Editor for *IEEE Wireless Communications Magazine* Special Issue on Toward Seamless Internetworking of Wireless LAN and Cellular Networks, and the Editor of a Special Issue of the *Wiley Journal on Wireless Communications and Mobile Computing* on Radio Link and Transport Protocol Engineering for Future Generation Wireless Mobile Data Networks.