This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2007 proceedings.

Link Scheduling for Minimum Delay in Spatial Re-use TDMA

Petar Djukic and Shahrokh Valaee
The Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto, 10 King's College Road, Toronto, ON, M5S 3G4, Canada
e-mail:{djukic,valaee}@comm.utoronto.ca

***Abstract*— Time division multiple access (TDMA) based medium access control (MAC) protocols provide QoS with guaranteed access to wireless channel. However, in multihop wireless networks, these protocols may introduce delay when packets are forwarded from an inbound link to an outbound link on a node. Delay occurs if the outbound link is scheduled to transmit before the inbound link. The total round trip delay can be quite large since it accumulates at every hop in the path. This paper presents a method that finds schedules with minimum round trip scheduling delay.**

We show that the scheduling delay can be interpreted as a cost collected over a cycle on the conflict graph. We use this observation to formulate a min-max program for the delay across a set of multiple paths. The min-max delay program is NP-complete since the transmission order of links is a vector of binary integer variables. We design heuristics to select appropriate transmission orders. Once the transmission orders are known, a modified Bellman-Ford algorithm is used to find the schedules. The simulation results confirm that the proposed algorithm can find effective min-max delay schedules.

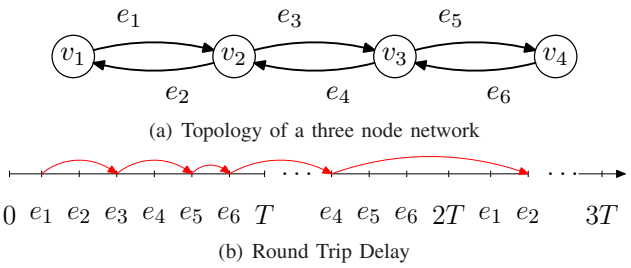
***Index Terms*— TDMA Scheduling, Network Flows, Cycles in Graphs**

I. Introduction

New applications of wireless multihop networks, such as commercial mesh networks, require guaranteed Quality-of-Service (QoS) in the MAC layer. This has prompted development of new multihop MAC protocols based on Time Division Multiple Access (TDMA), such as 802.11s and 802.16 [**?**], [1], [2]. These new protocols provide guaranteed link bandwidth with scheduled access to 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 non-conflicting links can transmit together taking advantage of spatial reuse. The bandwidth of each link is given by the number of slots it is assigned in the frame.

Our paper answers the following important question: *Given an assignment of link bandwidths in the network, is there a minimum delay schedule that can realize the assignment?* The delay in the network consists of queueing delay due to traffic variations and TDMA scheduling delay accumulated on missed inbound to outbound link packet handoffs. In this paper, we minimize the TDMA scheduling delay.

The TDMA scheduling delay occurs when packets are forwarded from an inbound link to the outbound link. The delay can be large if the outbound link is scheduled to transmit before the inbound link in the same frame. In such cases, arriving packets on an inbound link should wait for the subsequent frame to be transmitted on the outbound link. TDMA scheduling delay accumulates at every hop in the network, so the total delay experienced on the return path can be large. Consider the network in Fig. 1(a) and the transmission order $e_1 \rightsquigarrow e_2 \rightsquigarrow e_3 \rightsquigarrow e_4 \rightsquigarrow e_5 \rightsquigarrow e_6$.[1] With this transmission order, round trip time (RTT) for a packet traveling between v_1 and v_4 is more than two frames (Fig. 1(b)). Note that at both node v_3 and node v_2, the packet is delayed one frame while respectively waiting for links e_4 and e_2 to transmit in the next frame. On the other hand, if the order of transmissions is $e_1 \rightsquigarrow e_3 \rightsquigarrow e_5 \rightsquigarrow e_6 \rightsquigarrow e_4 \rightsquigarrow e_2$, RTT for all nodes is one frame.



(a) Topology of a three node network

(b) Round Trip Delay

Fig. 1. TDMA Scheduling Delay

When the link bandwidths are known in advance, the goal is to design schedules with minimum TDMA delay. Previously, this was addressed by finding schedules with minimum frame size [3]–[7], by reducing the problem to designing a minimum edge colouring. However, this is not a practical solution in networks using a protocol such as 802.16, where the whole network has to be rebooted to change the frame length [?]. We will show later that minimum length scheduling does not automatically guarantee minimum TDMA delay, so our algorithm improves the performance in the network over that achieved with minimum frame scheduling. This approach may also be ineffective, especially if there are many link-to-link handoffs. For example, in a mesh network all paths form a routing tree with the *point-of-presence* (or base-station) as the

This work was sponsored in part by the LG Electronics Corporation.

[1] We use $e_i \rightsquigarrow e_j$ to mean link e_i transmits before link e_j.

0743-166X/07/$25.00 ©2007 IEEE

root. The point-of-presence is connected to the Internet, so all packets in the network must cross the point-of-presence. In this case, the schedule should ensure that the delay from every node to the point-of-presence is small. In the present paper, we find an ordering of link transmissions resulting in TDMA schedules were the delay is minimized over a set of paths. To the best of our knowledge, our paper is the first to address the TDMA scheduling delay in the context discussed above.

In this paper, we solve the minimum TDMA scheduling problem in two parts. First, we formulate the scheduling problem as a network flow problem in the conflict graph. This formulation allows us to state the conditions on the existence of a feasible schedule as a set of linear constraints, one for each conflict in the network. The inequalities in the constraints are defined in relation to the transmission duration of each link in the conflict and the transmission order of the links. We then use the constraints to formulate a $\{0,1\}$-integer linear program that finds a min-max delay for a subset of paths in the network. The binary variables correspond to the transmission order of links in the frame. The number of binary variables in the linear program is equal to the number of active links in the network.

Second, since the $\{0,1\}$-integer linear program is hard to solve in on-line situations, we separate the TDMA scheduling into finding transmission orders and finding a schedule with a given transmission order. We show that if the transmission order is fixed, the schedule can be found in polynomial time by applying the Bellman-Ford algorithm on an augmented version of the conflict graph. We then show that delay is minimized if all links transmit in a sequence corresponding to their return path, and extend this to tree topologies. However, this transmission order ignores spatial reuse in the network, so it may not result in a feasible schedule. We propose a heuristic that adds spatial reuse to the transmission order so that the maximum delay on any path bound by a parameter is passed to the heuristic. We examine the performance of the heuristics compared to the performance of the full scheduling algorithm with numerical simulations.

We now review the related work. In [3] the authors provide a polynomial time algorithm that finds a minimum length TDMA schedule that can carry a given bandwidth allocation. The authors assume the only conflicts in the network are primary conflicts between links sharing a neighbour. In [4] the authors also use the assumption that there are no secondary conflicts in the network and find a TDMA schedule that supports a set of bandwidths through an edge colouring on a multi-edged version of the topology graph. Edge colouring is used in [5], but with additional operation after the colouring, which finds transmission directions with no secondary conflicts. In [6] the authors include secondary conflicts by adding interference edges to the topology graph and provide heuristic edge colouring algorithms for the enhanced graph. In [7] scheduling is reduced to finding a minimum node colouring of the chain obtained by flattening the network topology graph. [8] presents a distributed TDMA scheduling algorithm that converges to the bandwidth requested by higher layers. The algorithm controls the feasibility of link bandwidths by enforcing a set of local conditions on a tree topology, and assumes usage of an asynchronous TDMA scheme. In [9] authors use the independent set polytope of the conflict graph to state the existence of a schedule. We also use the conflict graph, with the difference that in this paper a schedule is explicitly stated in our feasibility conditions, and that we do not require the set of all independent sets in the conflict graph, which may be exponentially large.

II. Network and Transmission Model

We assume that the network is using Time Division Multiple Access (TDMA) MAC protocol [?], [1], [2]. The time is divided into slots of fixed duration, which are grouped into frames. A fixed portion of the frame is dedicated to control traffic, while the other slots are reserved for data traffic. The slots in the data frame are assigned through the exchange of messages in the control part of the frame, which establish a common transmission schedule. The schedule repeats in every frame until traffic demands in the network change.

The TDMA network can be modelled 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 symmetrical links in the MAC layer, so the TDMA network can be represented with a connectivity graph $G(V, E, f_t)$, where $V = \{v_1, \ldots, v_n\}$ is the set of nodes, $E = \{e_1, \ldots, e_m\}$ are directional links between neighbouring nodes, and $f_t : E \to V \times V$ assigns links to pairs of nodes. The connectivity map f_t is used to enforce the fact that all links are directional, so for a link $e_k \in E$, $f_t(e_k) = (v_i, v_j)$ means that the traffic on the link is transmitted from v_i to v_j.

Links are established between neighbours if they are in the range of each other, however since all transmissions are over the wireless channel, we associate a link bitrate to every link to model channel quality. The bitrate depends on the modulation, which is chosen based on signal-to-noise ratio for the link. The signal-to-noise ratio is divided into several discrete levels and each is associated with its maximum bitrate. We define the link bitrate as the number of bits transmitted in a TDMA slot, which is represented with the mapping $b : E \to \{B_1, B_2, \ldots, B_{\max}\}$, where B_1 is the number of bits carried in a slot with the minimum modulation and $B_{\max}$ is the number of bits carried in a slot with the maximum modulation.

We model the connections in the network with a multicommodity bandwidth assignment, so for a source node v_i, $g_i^\phi > 0$ is the number of bits arriving for connection (commodity) ϕ in one frame. On the other hand for the destination node v_j, $g_j^\phi < 0$ is the number of bits leaving the network on connection ϕ in one frame. For example, in a mesh network each node has an uplink and a downlink connection to the point-of-presence. The point-of-presence allows the nodes to reach the Internet, so each mesh node establishes two connections to it, one for the uplink and one for the downlink traffic.

Link bandwidth is defined as the number of bits the link transmits in a frame. The link bandwidth is shared between

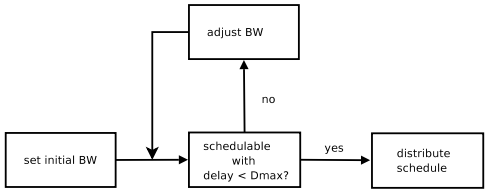


Fig. 2. Joint Routing and Scheduling Algorithm

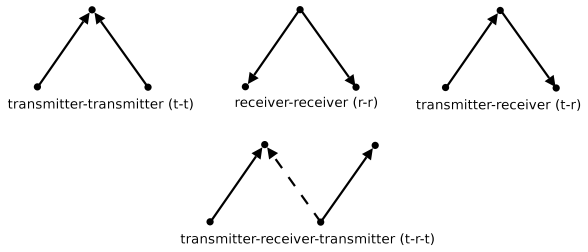


Fig. 3. Conflicts in TDMA Wireless Networks

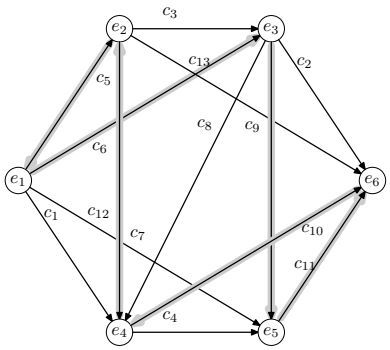


Fig. 4. Conflict Graph

different commodities, so

$$\sum_{e_j \in \{v_i\}^+} f_j^\phi = g_i^\phi + \sum_{e_k \in \{v_i\}^-} f_k^\phi, \tag{1}$$

where f_j^ϕ is the bandwidth allocated to commodity ϕ on link e_j, and $\{v_i\}^+$ and $\{v_i\}^-$ are respectively the set of outgoing and incoming links of the node.

The assignment of link bandwidths and node bandwidths so that flow conservation, (1), is satisfied is not the topic of this paper. We assume that the links are assigned bandwidth as a part of a joint routing and scheduling algorithm (Fig. 2). The joint routing and scheduling algorithm finds an initial set of link bandwidths satisfying a certain QoS and then checks if there is a TDMA schedule that supports the bandwidth. If a schedule exists and it's delay is less than the maximum delay tolerated in the network $D_{\max}$, the algorithm is done, otherwise the algorithm iteratively adjusts the bandwidths until a schedule is found.

Given the assignment of link bandwidths for the commodities, the number of timeslots the link should be active in the frame can be found from:

$$d_j = \left\lceil \frac{\sum_\phi f_j^\phi}{b_j} \right\rceil = \left\lceil \frac{f_j}{b_j} \right\rceil, \tag{2}$$

where $f_j = \sum_\phi f_j^\phi$ is the total number of bits transmitted by all commodities using the link, b_j is the modulation on the link and d_j is the number of slots link e_j transmits in a frame.

A link may interfere (conflict) with other links, so they should not transmit at the same time. There are four types of transmission conflicts that need to be considered in TDMA networks (Fig. 3). The first three types are between the connections that share a neighbour. In the case of the transmitter-transmitter (t-t) conflict the parallel transmissions garble each other at the common receiver, and in the case of the receiver-receiver (r-r) conflict a single transmitter cannot separate packets for the two receivers. The transmitter-receiver (t-r) conflict happens because the nodes cannot transmit and receive at the same time.

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

We keep track of conflicts between the links with conflict graphs. Conflict graphs can be defined with a triplet $G_c(E, C, f_c)$, where E is the set of links, $C = \{c_1, \ldots c_r\}$ is the set of TDMA conflicts, one for each of the r conflicting pairs of links, and $f_c : C \rightarrow \{\{e_i, e_j\}, \text{for all } e_i, e_j \in E\}$ associates the conflicts with pairs of links. We use the notation $\{\cdot\}$ for unordered sets and $(\cdot)$ for ordered sets, so f_c defines an undirected graph. The graph is undirected since conflicts are symmetrical.

In this paper, we use a conflict graph with an arbitrary assignment of directions to the arcs, $\overrightarrow{G}_c(E, C, \overrightarrow{f}_c)$, where $\overrightarrow{f}_c : C \rightarrow E \times E$. The directed conflict graph simplifies the derivation of formulas, however the arbitrary orientation of arcs does not cause any loss of generality. We use the four node example from Fig. 1 to demonstrate how the arcs in the conflict graph are created. The vertices in the conflict graph are the six links from the connectivity graph. All of the links conflict with each other, except for pairs e_1 and e_6 and e_2 and e_5, so they are not connected (Fig. 4). The graph in the figure also has an arbitrary orientation.

III. TDMA Scheduling Problem

A TDMA schedule assigns each slot in a frame to a link. Since the schedule repeats from frame to frame, it is sufficient to represent this assignment with a map $I : E \times M \rightarrow \{0, 1\}$, where E is the set of links and $M = \{0, \ldots, T-1\}$ is the index of T slots in the frame [3], [10]. Once a schedule is decided, the map is repeated in every frame until a new

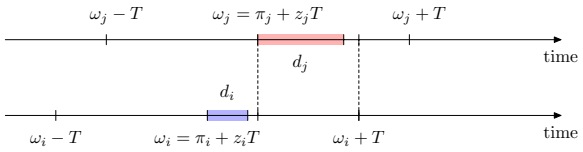


Fig. 5. Series of Activation Times

schedule is determined. The meaning of the scheduling map is that a link e_i transmits in slot t if $I(e_i, t) = 1$. Since the schedule repeats in every frame, if $I(e_i, t) = 1$, link e_j is active at all times in the set $\{t + z_i T, z_i \in \mathbb{Z}\}$.

A map I defines a *valid* schedule if every link is allocated the number of slots required to transmit all of its data,

$$\sum_{t=0}^{T-1} I(e_i, t) = d_i, \quad \forall e_i \in E \tag{3}$$

where d_i is given by (2). A map I defines a *conflict-free* schedule if conflicting links do not transmit at the same time,

$$I(e_i, t) + I(e_j, t) \leq 1, \quad \forall t \in M, \forall c_k \in C, \tag{4}$$

where $f_c(c_k) = \{e_i, e_j\}$. In general, the TDMA scheduling problem is to find a scheduling map I that represents a valid and a conflict free schedule.

The problem with using the assignment matrix I to represent TDMA schedules is that it does not allow an easy way to limit the number of times a link transmits in a frame. The importance of limiting the number of link transmissions is twofold. First, this is consistent with the 802.11s and 802.16 mesh protocols, i.e. links are only allowed to transmit once in a frame. Second, limiting the number of times a link transmits in a frame decreases the total network overhead. For example, in 802.16 every transmission needs a guard time of three TDMA slots, which at the highest modulation means an overhead of 324 bytes per transmission. In this paper, we will assume that the number of transmissions is limited to one per frame. Later we will show how we can implement schedulers with an arbitrary number of transmissions per frame.

Limiting each link to transmit at most once in a frame, is equivalent to limiting the set of possible scheduling maps I to matrices with rows of consecutive ones [11]. The first column where the sequence of ones starts is the link activation time, and the length of the sequence is the duration of the link transmission.

The consecutive one property of the scheduling matrix allows us compress it into a pair of maps $S(\boldsymbol{\pi}, \boldsymbol{d})$, where $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_m]^T$ is the vector of start time offsets corresponding to each link and $\boldsymbol{d} = [d_1, \ldots, d_m]^T$ is the vector corresponding to the duration of link transmissions, found in (2). The vector of start times $\boldsymbol{\pi}$ is restricted to be positive and less than T, $\boldsymbol{\pi} \in \mathbb{Z}^m_{[0,T)}$. The schedule $S(\boldsymbol{\pi}, \boldsymbol{d})$ corresponds to a scheduling matrix I_S in which a row corresponding to link e_i has the first π_i columns set to zero, and the columns $\{\pi_i, \ldots, \pi_i + d_i - 1\}$ set to one. The TDMA scheduling problem, in this paper, is to find the schedule $S(\boldsymbol{\pi}, \boldsymbol{d})$ whose corresponding scheduling matrix I_S is valid and conflict free and also results in minimum TDMA delay.

Since the activation times are periodic, the start time π_i for link e_i actually represents a series of activation times, which can be derived from π_i by adding multiples of T slots. So, $\Pi_i = \{\pi_i + z_i T, z_i \in \mathbb{Z}\}$ is the series of activation times for link e_i, generated with π_i (Fig. 5). The normalized activation time π_i can be found from any activation time $\omega_i \in \Pi_i$ with the modulo operator:

$$\pi_i = \omega_i \pmod{T}. \tag{5}$$

The schedules defined by $S(\boldsymbol{\pi}, \boldsymbol{d})$ are valid by construction since every link e_i is allocated d_i columns in the scheduling matrix I_S. In order for the schedule to be conflict free, it must ensure that if a link starts transmitting, all of its conflicting links remain silent until the link's transmission is over. For two conflicting links e_i and e_j, this conflict free condition should be true for all time points $\omega_i \in \Pi_i$ and $\omega_j \in \Pi_j$ that occur in the same one frame interval.

Take any activation time $\omega_i \in \Pi_i$ for link e_i and choose the next activation point for a conflicting link e_j, $\omega_j = \min\{\omega \in \Pi_j : \omega \geq \omega_i\}$ (Fig. 5). In this case, e_j should not transmit before e_i finishes its transmission

$$\omega_j \geq \omega_i + d_i \Leftrightarrow \quad \omega_j - \omega_i \geq d_i, \tag{6}$$

and e_j should stop transmitting before e_i transmits again

$$\omega_j + d_j \leq \omega_i + T \Leftrightarrow \quad \omega_j - \omega_i \leq T - d_j. \tag{7}$$

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

$$d_i \leq \omega_j - \omega_i \leq T - d_j, \quad \text{and} \quad 0 < \omega_j - \omega_i < T. \tag{8}$$

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

$$d_j \leq \omega_i - \omega_j \leq T - d_i, \quad \text{and} \quad 0 < \omega_i - \omega_j < T. \tag{9}$$

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

$$d_i - p_k T \leq \omega_j - \omega_i \leq T - d_j - p_k T, \tag{10}$$

where $p_k = 0$ if $\omega_j - \omega_i > 0$ and $p_k = 1$ if $\omega_j - \omega_i < 0$. The extra variable p_k specifies a relative order of transmissions, which prompts us to refer to it as the "transmission order" in the rest of the paper.

A subset of activation times $\omega_i \in \Pi_i$, $\forall e_i \in E$, can be interpreted as a "potential" on the conflict graph, if we take $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_m]$ to be a function on the vertices of the conflict graph: $\omega : E \rightarrow \mathbb{Z}$. In this case, the conflict-free conditions for a schedule can be stated in terms of potentials in the conflict graph, thus formulating the scheduling problem as a network flow problem:

Proposition 1 (Conflict-Free Schedules): Schedule $S(\boldsymbol{\pi}, \boldsymbol{d})$ is conflict-free if and only if there exists a potential $\omega : E \rightarrow \mathbb{Z}$, such that for every arc $c_k \in C$ and its corresponding pair of

conflicting links $\overrightarrow{f}_c(c_k) = (e_i, e_j)$ the conflict-free condition, (10), is satisfied.

The importance of potentials becomes clear from the "only if" part of the proposition, which gives a strategy to find feasible conflict-free schedules. The strategy is to simultaneously look for an ordering vector $\boldsymbol{p}$ and a potential $\boldsymbol{\omega}$, satisfying the conflict free conditions (10) for every pair of conflicting links. This search problem was shown to be NP-complete in [12], by reduction to Graph K-Colourability. A more general version of the problem, where p is allowed to take any integer value, is known as the Periodic Event Scheduling Problem (PESP) [13].

The proposition defines a polyhedron of feasible schedules defined by $\boldsymbol{p}$ and $\boldsymbol{\omega}$. We will use this fact in the next section to define constraints in the optimization of minimum delay. However, the proposition also gives a natural way to split the TDMA scheduling problem into two parts: finding $\boldsymbol{p}$ and then finding $\boldsymbol{\omega}$. We will show that $\boldsymbol{\omega}$ can be found with the Bellman-Ford algorithm if $\boldsymbol{p}$ is fixed. So, we will tackle the scheduling problem by finding good heuristics for $\boldsymbol{p}$.

IV. Minimum Delay TDMA Scheduling

In this section we formulate an optimization that the joint routing and scheduling algorithm can use to find schedules with maximum delay less than the QoS constraint $D_{\max}$. The optimization minimizes the maximum delay among a set of paths. The optimization can result in one of two outcomes. First, there may not be a schedule that support the link bandwidths in which case the joint routing and scheduling algorithm needs to adjust the link bandwidths. Second, there may be a schedule whose maximum delay on any path is given by the objective function of the optimization. In this case, the joint routing and scheduling algorithm can decide if the delay is larger than $D_{\max}$ and adjust the link bandwidths if needed.

We consider a scenario in which the TDMA delay should be minimized on some set of paths. For example, in mesh networks, there are $q = n-1$ paths forming a tree rooted at the point-of-presence and leading to each one of the $n-1$ nodes in the mesh. In this case, a schedule should ensure that none of the paths have a very large delay. We first find the delay on return paths and then use this delay as an objective function to a linear program that minimizes the maximum among the path delays in a set of paths.

A. TDMA Delay

TDMA delay may occur during any link-by-link packet forwarding along a path between the source and the destination. A path $\mathcal{P} = (e_i, \ldots, e_j)$ is an ordered sequence of links connecting two vertices in the topology graph. Links adjacent to each other in the path share a common node, which forwards packets. For example, if link e_j follows another link e_i in the path, their common node forwards packets received on link e_i to link e_j. TDMA delay occurs if, in the same frame, link e_j is scheduled before link e_i.

Each path $\mathcal{P}$ in the topology graph corresponds to a path $\theta_{\mathcal{P}}$ in the conflict graph. The path in the conflict graph can be obtained by finding the conflicts needed to visit the vertices of the conflict graph $\overrightarrow{G}_c(V, E, \overrightarrow{f}_c)$ listed in $\mathcal{P}$. For example, the return path $\mathcal{P} = (e_1, e_3, e_5, e_6, e_4, e_2)$ in the four node topology shown in Fig. 1, corresponds to the path $\theta_{\mathcal{P}} = \{c_6, c_9, c_{11}, c_{10}, c_7\}$ in Fig. 4.

We find the total TDMA delay for a path $\mathcal{P}$ by finding the delay incurred while traversing the corresponding path $\theta_{\mathcal{P}}$ in the scheduling graph. For each packet forwarded from an inbound link to an outbound link on a node, there is a conflict connecting the two links in the conflict graph. Therefore, the total delay for traversing a path $\mathcal{P}$ in the topology graph is identical to the total delay for the corresponding path $\theta_{\mathcal{P}}$ in the conflict graph. So, we first we find the hop-by-hop delay by examining how packets are delayed at each conflict and then we sum them up over $\theta_{\mathcal{P}}$.

Consider a fixed schedule $S(\boldsymbol{\pi}, \boldsymbol{d})$ with a corresponding potential $\boldsymbol{\omega}$. The potential also has an ordering vector $\boldsymbol{p}$, such that the conflict free conditions (10) are true. Suppose that link e_i transmits at time ω_i and the next link on the path is e_j, that is $e_i \rightsquigarrow e_j$, and we have $\overrightarrow{f}_c(c_k) = (e_i, e_j)$. If $p_k = 0$, the delay on the conflict is $\Delta_k = \omega_j - \omega_i$ since $\omega_j > \omega_i$ by the conflict-free conditions. However, if $p_k = 1$, $\Delta_k = \omega_j - \omega_i + T$ since $\omega_j < \omega_i$ and the next transmission of e_j is at time $\omega_j + T$. So, for a conflict c_k, with the corresponding vertices $\overrightarrow{f}_c(c_k) = (e_i, e_j)$, time between successive transmissions is:

$$\Delta_k = \omega_j - \omega_i + p_k T = \tau_k + p_k T, \tag{11}$$

where $\tau_k = \omega_j - \omega_i$ is the tension for the conflict c_k with the end vertices e_j and e_i in the conflict graph. On the other hand if $\overrightarrow{f}_c(c_k) = (e_j, e_i)$, and the link is traversed in the opposite direction (i.e. $e_i \rightsquigarrow e_j$) it can be shown by a similar argument that:

$$\Delta_k = \omega_j - \omega_i + (1 - p_k)T = -\tau_k + (1 - p_k)T, \tag{12}$$

where the tension for this conflict is defined as $\tau_k = \omega_i - \omega_j$.

Using the single hop delay, we can find the delay on path $\theta_{\mathcal{P}}$ as:

$$\begin{aligned} D(\theta_{\mathcal{P}}) &= \sum_{c_k \in \{\theta_{\mathcal{P}}\}} \Delta_k \\ &= \sum_{c_k \in \{\theta_{\mathcal{P}}\}^+} \Big(\tau_k + p_k T\Big) - \sum_{c_k \in \{\theta_{\mathcal{P}}\}^-} \Big(\tau_k + p_k T - T\Big), \end{aligned} \tag{13}$$

where $\{\theta_{\mathcal{P}}\}^+$ is the set of conflicts where the packets are forwarded in the direction of the conflict, and $\{\theta_{\mathcal{P}}\}^-$ is the set of conflicts passed in their opposite directions.

We can also represent the delay as a linear combination of vectors, if we define a vector for a path in the conflict graph with $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_r]^T$, where:

$$\forall c_k \in C, \quad \theta_k = \begin{cases} 1, & \text{if } c_k \in \{\theta\}^+ \\ -1, & \text{if } c_k \in \{\theta\}^- \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

$\{\theta\}^+$ is the set of arcs in the positive direction of θ and $\{\theta\}^-$ is the set of arcs in the negative direction of θ. For example, the

cycle (path) emphasized in Fig. 4, corresponds to the vector $\boldsymbol{\theta} = [0,0,0,0,-1,1,-1,0,1,-1,1,0,0]^T$.

With the vector notation, the delay is a linear combination of the path $\boldsymbol{\theta}_\mathcal{P}$, tension and the transmission order:

$$D(\boldsymbol{\theta}_\mathcal{P}) = \boldsymbol{\theta}_\mathcal{P}^T[\boldsymbol{\tau} + \boldsymbol{p}T] + D_\mathcal{P} \tag{15}$$

where $D_\mathcal{P} = \sum_{c_k \in \{\theta_\mathcal{P}\}^-} T$ is a constant for the path and $\boldsymbol{\tau} = [\tau_1, \ldots, \tau_r]^T$, and $\boldsymbol{p} = [p_1, \ldots, p_r]^T$. Note that if path $\mathcal{P}$ starts with link e_i and ends with link e_j, $\boldsymbol{\theta}_\mathcal{P}^T\boldsymbol{\tau} = \omega_j - \omega_i$ since the terms in the middle cancel out.

In the rest of this section, we will concentrate on return paths in the topology graph. The return path delay is important for applications that use TCP as the transport protocol since the throughput of TCP is inversely proportional to the return path delay [14]. A return path in the topology graph corresponds to a cycle in the conflict graph. For example, the return path $\mathcal{P} = (e_1, e_3, e_5, e_6, e_4, e_2, e_1)$ in the four node topology shown in Fig. 1, corresponds to the path $\theta_\mathcal{P} = \{c_6, c_9, c_{11}, c_{10}, c_7, c_5\}$, marked in Fig. 4. Note that for return paths, $\boldsymbol{\theta}_\mathcal{P}^T\boldsymbol{\tau} = 0$, so the delay does not include the tension variables:

$$D(\boldsymbol{\theta}_\mathcal{P}) = \boldsymbol{\theta}_\mathcal{P}^T\boldsymbol{p}T + D_\mathcal{P}. \tag{16}$$

B. Min-Max TDMA Delay

We design our scheduler to minimize the total delay on some set of paths $\theta_1, \ldots, \theta_q$. An objective that achieves this goal is the min-max delay defined as $\min \max_{\mathcal{P}=1\ldots q} D(\theta_\mathcal{P})$. The constraints in the optimization can be obtained by the application of the conflict-free proposition. Note that the inequalities in the proposition can be stacked horizontally, so that we can express them as:

$$\boldsymbol{l} \le \boldsymbol{C}^T\boldsymbol{\omega} - \boldsymbol{p}T \le \boldsymbol{u}, \tag{17}$$

where $\boldsymbol{l} = [l_1, \ldots, l_m]^T$, $\boldsymbol{u} = [u_1, \ldots, u_m]^T$, and for an arc c_k with corresponding vertices $\overrightarrow{f}_c(c_k) = (e_i, e_j)$, $u_k = T - d_j$ and $l_k = d_i$ and $\boldsymbol{C}$ is the $m \times r$ incidence matrix of the conflict graph defined as

$$\boldsymbol{C}_{ik} = \begin{cases} 1, & \text{if } \overrightarrow{f}_c(c_k) = (e_i, e_j) \\ -1, & \text{if } \overrightarrow{f}_c(c_k) = (e_j, e_i) \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

We combine the polyhedron of feasible transmission orders with the formula for the delay to formulate an integer program that finds a schedule with the min-max delay:

$$\min_{\boldsymbol{\omega}, \boldsymbol{p}, t} \quad t \tag{19a}$$

$$\text{s.t} \quad \boldsymbol{\theta}_\mathcal{P}^T\boldsymbol{p}T + D_\mathcal{P} \le t, \quad \mathcal{P} = 1, \ldots, q \tag{19b}$$

$$\boldsymbol{l} \le \boldsymbol{C}^T\boldsymbol{\omega} - \boldsymbol{p}T \le \boldsymbol{u} \tag{19c}$$

$$\boldsymbol{\omega} \in \mathbb{Z}^m, \boldsymbol{p} \in \{0,1\}^r, t \in \mathbb{R}, \tag{19d}$$

The first q constraints (19b) 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 (19c) define the polyhedron of conflict-free schedules.

The optimum solution is a vector of feasible potentials $\boldsymbol{\omega}^*$ and a vector of optimum transmission order $\boldsymbol{p}^*$. The optimum schedule can be obtained from $\boldsymbol{\omega}^*$ with the modulo operation, so $\pi_i^* = \omega_i^* \pmod T$. The potential $\boldsymbol{\pi}$ has its own transmission order $\boldsymbol{p}^{(\pi)}$, which may be different from $\boldsymbol{p}$. The two orders may be different because $\boldsymbol{\omega}$ is related to the normalized activation time with $\omega_i = \pi_i + z_i T$ and for an arc c_k, with the endpoints $\overrightarrow{f}_c(c_k) = (e_i, e_j)$, z_i may be different from z_j while the feasibility conditions, (10), are still satisfied. If we substitute $\omega_i = \pi_i + z_i T$ into (10), we get the following:

$$d_i - (z_j - z_i + p_k)T \le \pi_j - \pi_i \le T - d_j - (z_j - z_i + p_k)T, \tag{20}$$

which shows that the change in the relative transmission order may happen if $p_k = 0$ and $z_j - z_i = 1$, or if $p_k = 1$ and $z_j - z_i = -1$. The inequality also shows that the absolute transmission order in the frame, $\boldsymbol{p}^{(\pi)}$, is related to the relative transmission order with:

$$p_k^{(\pi)} = z_j - z_i + p_k. \tag{21}$$

Even though the order of transmissions is changed, the delay remains the same because the single hop delay is unchanged. The single hop delay in the positive direction of a conflict c_k, with $\overrightarrow{f}_c = (e_i, e_j)$ is:

$$\begin{aligned} \Delta_k^{(\pi)} &= \pi_j^* - \pi_i^* + p_k^{(\pi)*}T \\ &= \pi_j^* - \pi_i^* + (z_j - z_i + p_k^*)T = \omega_j^* - \omega_i^* + p_k^*T, \end{aligned} \tag{22}$$

which is the same as (11). Similarly the single hop delay in the opposite direction of the conflict is also valid:

$$\Delta_k^{(\pi)} = \pi_i^* - \pi_j^* + (1 - p_k^{(\pi*)})T = \omega_i^* - \omega_j^* + (1 - p_k^*)T \tag{23}$$

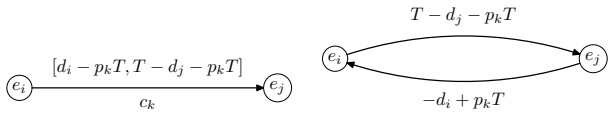
the same as (12).

V. Heuristics for Minimum Delay Scheduling

The min-max formulation (19) is hard to solve due to the transmission ordering variables $\boldsymbol{p}$. The optimization requires a search for $\boldsymbol{p}$ over the space of all $\{0,1\}^r$ vectors of $\boldsymbol{p}$. This search may be done with a standard branch-and-bound technique [15] or with a customized cutting plane algorithm [16]. We are interested in finding a method that allows us to use heuristics to find the schedule.

We split the scheduling problem into two parts. First, a search for a transmission order $\boldsymbol{p}$ is performed and then for the fixed $\boldsymbol{p}$ the corresponding potential $\boldsymbol{\omega}$ is sought. If the transmission order $\boldsymbol{p}$ is fixed, the potential $\boldsymbol{\omega}$ can be found with a modified Bellman-Ford algorithm. We therefore limit heuristics to finding good transmission orders.

Separating the search for the transmission order from finding a feasible schedule is also well suited for use with the 802.16 centralized scheduling protocol. In the 802.16 centralized scheduling protocol, the base-station finds a routing tree and calculates the link assignments, and disseminates both throughout the network. The mesh nodes use the routing tree and the assignments to find schedules. The routing tree can also identify the transmission order, and therefore the proposed solution can indeed be applied in the 802.16 centralized protocol.



(a) Single Edge with Double Costs (b) Double Edges with Single Costs

Fig. 6. Interpretation of Costs

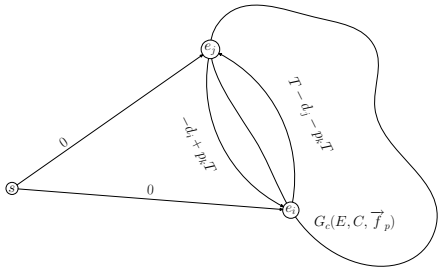


Fig. 7. Augmented Conflict Graph

A. Scheduling with a Fixed Transmission Order

In general, the scheduling problem is NP-complete. However, if the transmission order $\boldsymbol{p}$ is fixed, the scheduling problem becomes that of finding a solution to a set of difference equations. The solution to these difference equations can be found efficiently using the Bellman-Ford algorithm [17]. Here, we describe the scheduling algorithm for a fixed $\boldsymbol{p}$.

First, we transform the conflict graph $G_c(E_c, C_c, \overrightarrow{f}_c)$ so that the cost of traversing the arcs is taken from the upper and lower bounds in the difference equations. The upper bound is treated as the cost of traversing an arc in the positive direction, while the lower bound is treated as the cost of traversing the arc in the negative direction. This situation is illustrated in Fig. 6, where an edge in the scheduling graph is replaced with a set of two edges. If the original edge is traversed in the positive direction, $e_i \rightsquigarrow e_j$ in Fig. 6(a), this is equivalent to traversing the upper edge with cost $T - d_j - p_k T$ in Fig. 6(b). Similarly, if the original edge is traversed in the negative direction, $e_j \rightsquigarrow e_i$ in Fig. 6(a), this is equivalent to traversing the lower edge with cost $-d_i + p_k T$ in Fig. 6(b).

Second, we add a new vertex s to the transformed scheduling graph and connect it to each of the original vertices in the graph with an arc of cost 0 and an arc pointing out of s (Fig. 7). The feasible potential is the minimum distance from s to every node in the graph. This is easily seen by the optimality of the minimum paths from s to each $e_i \in E$, [17], where for all conflicts incident to e_i, $\forall c_k \in C : \overrightarrow{f}_c(c_k) = (e_i, e_j)$:

$$\omega_j \leq \omega_i + T - d_j - p_k T, \tag{24a}$$

$$\omega_i \leq \omega_j - d_i + p_k T, \tag{24b}$$

where ω_j is the total cost of reaching e_i from s. The two conditions correspond to traversing an edge in the positive and negative directions, and when they are combined, we get the feasibility condition (10).

It is possible that the shortest path algorithm may not find distances to all of the vertices in the scheduling graph. This can happen if the conflict graph contains directed cycles with negative cost [17]. In this case, the transmission order specified by $\boldsymbol{p}$ does not have a valid schedule and a new transmission order should be found.

We modify the Bellman-Ford algorithm to find the minimum distance from s to all other vertices. One way to implement the Bellman-Ford algorithm that takes the dual cost into account is to transform the augmented scheduling graph into a symmetrical graph with the costs as shown in Fig. 6b and then directly apply the Bellman-Ford algorithm. However, this may be costly if the scheduling graph has many arcs. An equivalent way of accomplishing the same goal is to change the relaxation in the Bellman-Ford algorithm [17]. The relaxation examines every arc for violations of the optimality conditions and adjusts the cost of reaching a vertex if the optimality is violated. We modify the relaxation in our algorithm to examine both the cost of traversing the edge in the positive direction and cost of traversing the edge in the negative direction, in effect doing the same thing that the Bellman-Ford algorithm would do on the graph with the symmetrical edges.

The last step in the algorithm is to find a schedule in the time period $[0, T)$. Since the minimum distance ω_i is also one of the activation points in the series of activation points Π_i, we can use the modulo rule to find the schedule $\boldsymbol{\pi}$ with $\pi_i = \omega_i \pmod T$, for the link e_i.

The algorithm can easily be extended to allow links to transmit more than once in a frame. In general, a link has a number of vertices in the conflict graph that is equal to the number of times it transmits in the frame. For example, if a link transmits twice in the frame, we add an extra vertex for it to the conflict graph. The new vertex conflicts with the same links as the old vertex associated with the link. In addition, we also add a conflict between the new vertex and the original vertex and take into account the restriction on their transmissions with (10). The solution for the scheduling problem identified with the new graph will schedule the node with two vertices in the conflicting graph to transmit twice in the frame.

B. Transmission Order Heuristic

In this subsection, we propose an algorithm that can be used to find a transmission order whose maximum delay on any path is T. The algorithm defines a ranking function $R : E \rightarrow \mathbb{Z}$, which indicates the preferred order of transmissions of the links, and then uses the ranking function to find the transmission order $\boldsymbol{p}$.

Initially, the algorithm sets the rank of all the nodes to zero. The algorithm then examines each of the q return paths to the point-of-presence, 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 point-of-presence is $v_1 \in V$. For links in a return path $\mathcal{P} = \{e_i, \ldots, e_k, e_l, \ldots e_j\}$, where $e_i \in \{v_1\}^+$ and $e_j \in \{v_1\}^-$, the rank is assigned as follows:

$$R_k = \max\{R_k, R_l + 1\}, \forall e_k, e_l \in \mathcal{P} : e_l \rightsquigarrow e_k. \tag{25}$$

We note that the distance of the link is defined as it's placement on the return path and not it's topological distance from the root of the tree. For the example in Fig. 1(a), we have $R_1 = 0, R_3 = 1, R_5 = 2, R_6 = 3, R_4 = 4$, and $R_2 = 5$.

Given the ranking, the transmission order $\boldsymbol{p}$ is assigned with the following rule: $\forall c_k \in C$, with $\overrightarrow{f}_c(c_k) = (e_i, e_j)$,

$$p_k = \begin{cases} 0, & \text{if } R_j \geq R_i \\ 1, & \text{otherwise} \end{cases} \tag{26}$$

Proposition 2: If the ordering $\boldsymbol{p}$, derived from the ranking $R : E \to \mathbb{Z}$, has a feasible schedule, then for any path $\mathcal{P}$,

$$D(\theta_{\mathcal{P}}) = T. \tag{27}$$

Proof: Consider a return path $\mathcal{P} = \{e_i, \dots e_j\}$, where $e_i \in \{v_1\}^+$ and $e_j \in \{v_1\}^-$, and its corresponding cycle $\theta_{\mathcal{P}}$ in the conflict graph. Assume that c_l is the last conflict in $\theta_{\mathcal{P}}$, connecting e_j and e_i in the conflict graph. By construction of $\boldsymbol{p}$, $p_k = 0$ if $c_k \in \{\theta_{\mathcal{P}}\}^+ \setminus \{c_l\}$ and $p_k = 1$ if $c_k \in \{\theta_{\mathcal{P}}\}^- \setminus \{c_l\}$. For conflict c_l, $p_l = 1$ if $c_l \in \{\theta_{\mathcal{P}}\}^+$ and $p_l = 0$ if $c_k \in \{\theta_{\mathcal{P}}\}^-$. The delay can be found as:

$$\begin{aligned} D_{\mathcal{P}} &= \boldsymbol{\theta}_{\mathcal{P}} \boldsymbol{p} T + \sum_{c_k \in \{\theta_{\mathcal{P}}\}^-} T \\ &= -\left|\{\theta_{\mathcal{P}}\}^- \setminus \{c_l\}\right| T + p_l T + \left|\{\theta_{\mathcal{P}}\}^-\right| T = T, \end{aligned} \tag{28}$$

where $|\cdot|$ is the cardinality of a set. ■

The problem solved above cannot handle spatial reuse since all links will be scheduled sequentially. In the sequel, we propose a heuristic that modifies the ranking to introduce spatial reuse on the paths. The proposed solution allocates new ranks found from

$$R_i^H = R_i \pmod H, \tag{29}$$

where H is a constant usually larger than three.[2]

The new ranking function introduces spatial reuse because it allows links, far enough on the same path, to transmit at the same time. The new ranking function R_i^H is reset every H hops. Suppose link e_j follows link e_i on the path, such that $R_i < R_j$. Then, in the proposed transmission ordering, e_i will transmit before e_j. However, if in the modified scheme $R_i^H > R_j^H$, then e_j will be scheduled to transmit before e_i, possibly at the same time as some of the other links preceding e_i in the path, thus increasing spatial reuse. This scheme is sub-optimal because it does not maximize spatial reuse in the network.

The transmission ordering function found with R_i^H increases the set of links in the opposite direction of a path. It reverses every Hth link on a path. So, on the longest path with $n_{\max}$ hops, the new ranking function introduces the delay:

$$D_{\max} = \left\lfloor \frac{n_{\max}}{H} \right\rfloor T. \tag{30}$$

It can be easily seen that the delay is less than $D_{\max}$ on all paths with less than $n_{\max}$ hops as well, i.e.

$$\max_{\mathcal{P}=1,\dots,q} D(\boldsymbol{\theta}_{\mathcal{P}}) \leq \left\lfloor \frac{n_{\max}}{H} \right\rfloor T. \tag{31}$$

[2]On a long chain, links that are at least 2 hops apart can transmit simultaneously.

VI. Numerical Results

We perform two types of simulations. First, we examine TDMA delay scheduling algorithms on a chain topology. Second, we compare the minimum delay integer program with the heuristic on a mesh topology.

We create a scenario where the network has a chain topology with n nodes. The two nodes at the opposite sides of the chain establish an uplink and a downlink connection with symmetrical bandwidths $g_n^{\text{up}} = g_n^{\text{dn}} = g_n$. We call g_n the load. We fix the frame length to $T = 100$ slots and slot size to 1ms, making the frame duration 100ms. We assume that each link transmits for 10 slots in the frame and that each slot carries 10 bits, making the bitrate on all links, as well as the load, 1000bps.

Using the chain topology, we first compare the minimum frame length scheduling with minimum delay scheduling. The minimum frame length, $T_{\min}$, is found as the minimum number of colours needed to schedule the links without conflicts. We found that since every link is on for 10 slots in the frame, $T_{\min} = 40$ slots are needed to schedule all links for chain lengths longer than three nodes. The reason for this frame size is that the chain topology allows spatial reuse every four hops. Since we are comparing the TDMA delay, we keep slot sizes the same, making the minimum frame duration 40ms and we set the number of bits in every slot to 4 to maintain the same load.

We find the minimum delay schedule for $T_{\min}$ and plot it in Fig. 8 as "Min Len". We note that minimum length scheduling does not always produce minimum delay. In fact, the minimum length schedule has a smaller delay only when the number of hops is small. When the number of hops is large, the feasible set of schedules is restricted so that only schedules with a large delay are possible. Fig. 8 also compares the H-heuristic with the optimum scheduling. The figure shows that, as expected, the delay decreases when H is increased. We also plot the results of the LP-relaxation of the $\{0, 1\}$-integer optimization as "LP-Rel" in Fig. 8. The LP-relaxation finds a minimum delay schedule with a few iterations since it does not do branch-and-bound exhaustive search over the whole space of $\boldsymbol{p}$. The LP-relaxation performs well for a small number of nodes in the chain.

We also performed a Monte-Carlo simulation on a five by five mesh topology. We have repeated the Monte-Carlo simulation 100 times, and each time a set of sources has been chosen at random from the mesh and then connected to the point-of-presence in the corner of the topology. We then find a spanning tree in the network and allocate the bandwidth on the links so that the uplink and the downlink bandwidth of all sources is the same. We set the duration of all links to 10 slots, making the load of end-to-end connections 333bps, 250bps, 200bps for $T = 300, 400, 500$ slots in a frame, respectively.

We compare the performance of the heuristic by finding the probability a schedule exists to the probability that a schedule is found with the heuristic (Fig. 9). The probability of finding a schedule decreases as the frame size is decreased, since the

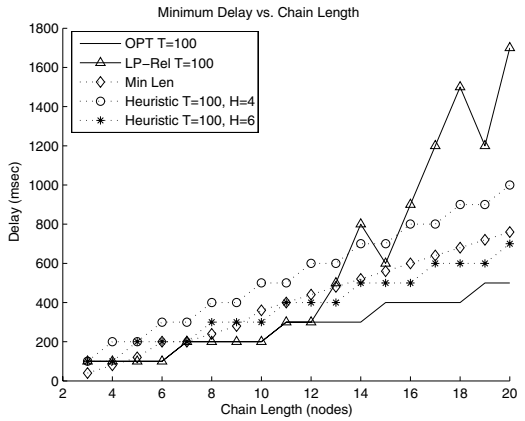


Fig. 8. Delay on the chain topology

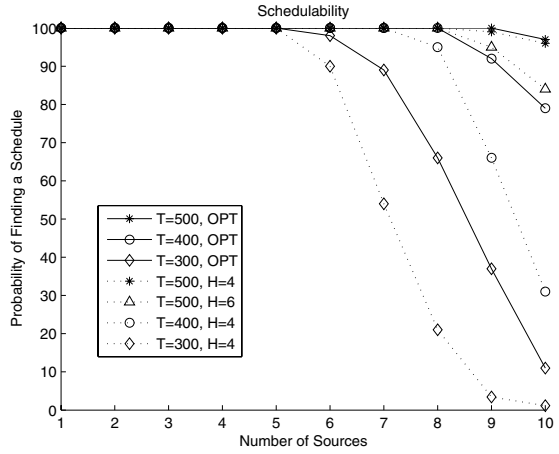


Fig. 9. Delay on the chain topology

load increases. For frame size $T = 500$, the full optimization finds a feasible schedule 96% of the time when there are 10 active sources in the mesh, while when the frame size is $T = 400$, the full optimization finds a feasible schedule 78% of the time. The proposed heuristic performs well when the total load in the network is moderate. For frame size of $T = 500$ the heuristic with $H = 6$ performs almost as well as the full optimization for up to 10 active sources. However, when $T = 400$ the load relative to the frame size increases and the heuristic performs worse than the optimal scheduling.

VII. Conclusion

This paper introduces a TDMA scheduling technique for application in multihop mesh networks. We have shown that the TDMA scheduling problem can be framed as a network flow problem on the conflict graph of the network. This formulation of the TDMA problem allows us to formulate the existence criteria for a feasible schedule as a set of linear constraints with the number of variables and the number of constraints in the order of the number of active conflicts in the network. This is a significant simplification of the TDMA scheduling problem. We use this network flow formulation of the TDMA scheduling problem to formulate a linear min-max delay optimization for TDMA networks. Our technique minimizes the maximum delay on a routing tree rooted at the point-of-presence (base-station).

We have also shown that the TDMA scheduling problem can be decomposed into two parts. In the first part, a relative transmission order of the links (precedence) is found. The relative transmission order is found by assigning ranks to links on each round trip path originated and terminated at the point-of-presence. The links are scheduled to minimize the maximum delay along the longest path in the tree. 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. This separation of the scheduling problem has allowed us to limit the scope of heuristics necessary for min-max schedules to relative transmission orders.

References

[1] R. Nelson and L. Kleinrock, "Spatial TDMA: A collision-free multihop channel access protocol," *IEEE Trans. Commun.*, vol. COM-33, no. 9, pp. 934–944, September 1985.

[2] IEEE, "802.11 TGs MAC enhacement proposal," IEEE, Protocol Proposal IEEE 802.11-05/0575r3, September 2005.

[3] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Trans. Inform. Theory*, vol. 34, no. 5, pp. 910–917, September 1988.

[4] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless networks: The joint routing and scheduling problem," in *MobiCom*, 2003.

[5] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in sensor networks: Distributed edge coloring revisited," in *INFOCOM*, 2005.

[6] S. Ramanathan and E. L. Lloyd, "Scheduling algorithms for multihop radio networks," *IEEE/ACM Trans. Networking*, vol. 1, no. 2, pp. 166–177, April 1993.

[7] S. C. Ergen and P. Varaiya, "TDMA scheduling algorithms for sensor networks," University of California, Berkley, Technical Report, July 2005.

[8] T. Salonidis and L. Tassiulas, "Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks," in *MobiHoc*, 2005, pp. 145–156.

[9] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *MobiCom*, 2003.

[10] E. Arkin, "Some complexity results about packet radio networks," *IEEE Trans. Inform. Theory*, vol. IT-30, no. 4, pp. 681–685, January 1984.

[11] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automat. Contr.*, vol. 37, no. 12, pp. 1936–1948, December 1992.

[12] M. A. Odijk, "Railway timetable generation," Ph. D., Technische Universiteit Delft, January 1998.

[13] P. Serafini and W. Ukovich, "A mathematical model for periodic scheduling problems," *SIAM Journal of Discrete Mathematics*, vol. 2, no. 4, pp. 550–581, November 1989.

[14] 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. Networking*, vol. 8, no. 2, pp. 133–145, April 2000.

[15] L. A. Wolsey, *Integer Programming*, ser. A Wiley-Intersicence Publication. New York: John Wiley & Sons, Inc., 1998.

[16] L. W. Peeters, "Cyclic railway timetable optimization," Ph. D., Erasmus Universiteit Rotterdam, June 2003.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT Press, 2001.