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

# Distributed Link Scheduling for TDMA Mesh Networks

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*— We present a distributed scheduling algorithm for provisioning of guaranteed link bandwidths in ad hoc mesh networks. The guaranteed link bandwidths are necessary to provide deterministic end-to-end bandwidth guarantees. Using Time Division Multiple Access (TDMA), links are assigned slots in each frame and during each slot a number of non-conflicting links can transmit simultaneously. The bandwidth of each link is given by the number of slots assigned to it the frame and the modulation used in the slots.**

**Our scheduling algorithm has two parts. The first part of the algorithm is an iterative procedure that finds locally feasible schedules by exchanging link scheduling information between nodes. The iterative procedure is based on the distributed Bellman-Ford algorithm running on the conflict graph, whose partial view is available at every node. The second part of the algorithm is a wave based termination procedure used to detect when all nodes are locally scheduled and a new schedule should be activated. We use analysis to show the worst case convergence time of the algorithm and simulations to show performance of the algorithm in practice.**

***Index Terms*— Quality-of-Service, Distributed TDMA Scheduling, Bellman-Ford**

## I. Introduction

Mesh networks provide a cost effective way to interconnect wireless access points (WAPs) over a large geographical area. The access points are used by wireless terminals as their first-hop connection. In addition to allowing terminals to access the network, each WAP also acts as a mesh router and forwards data packets to the point-of-presence (POP), which is connected to the Internet. Since the POP is the only node connected to the Internet, the cost of providing wireless access over a large area is significantly decreased [1]. Mesh networks are "ad hoc" networks in the sense that once the network is physically installed, the nodes will self-provision and create and maintain a wireless backbone to the POP without any action required by the operator.

Although mesh networks provide cost effective Internet access, they are still expected to provide a high level of quality-of-service (QoS) for applications such as voice over IP (VoIP) [2]. New mesh protocols provide QoS with Time Division Multiple Access (TDMA) technology. For example, IEEE has ratified a TDMA based mesh medium access control (MAC) protocol within the 802.16 MAC protocol, and has also started work on a mesh extension to 802.11, 802.11s, which has a TDMA mode [3], [4].

This work was sponsored by the LG Electronics Corporation.

In TDMA networks, QoS required by terminals is negotiated in terms of end-to-end bandwidth reserved for each terminal through the mesh network and enforced at each hop with scheduled access to the wireless channel. 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 simultaneously. The bandwidth of each link is given by the number of slots assigned to it in the frame and the modulation used in the slots. In this paper we provide a distributed algorithm that finds a common schedule for all links in the network, given their durations.

The distributed TDMA scheduling algorithm consists of two independent procedures. The first procedure is a distributed version of the Bellman-Ford algorithm running on the conflict graph for the network. In this procedure, each link finds locally feasible schedules, by exchanging update messages with its conflicting links, which are known from locally available two-hop neighbourhood information. This extends our work in [5], where we have proposed centralized TDMA scheduling algorithms.

The second procedure, which works independently of the first procedure, is used to find out when the locally feasible schedules converge to a globally feasible schedule. This solves the main difficulty with the distributed scheduling, which is notifying all the nodes of the availability of a new schedule. The procedure works in two phases, both of which are initiated by the POP. The first phase is a version of a wave based termination detection algorithm [6] in which the POP uses the routing tree to query the nodes about their termination status. The nodes respond to the query when they have a locally feasible schedule and all of their children have responded to their query. The second phase is initiated by the POP if it detects a successful termination of the first phase. In this phase, the POP sends a signal to activate the new schedule by applying the last recorded locally feasible schedule.

We show analytically that the scheduling procedure converges in at most $2m$ frames, where $m$ is the total number links in the network. However, we also show that in practice the algorithm converges much more quickly. We show that if scheduling messages are broadcast in the control part of the frame, the worst case difference between the time the last

**1-4244-0353-7/07/$25.00 ©2007 IEEE**

change in link bandwidth happens to the time a new schedule starts is $3n/\kappa T_f + 2mT_f$, where $n$ is the number mesh nodes in the network, $\kappa$ is the number of transmission opportunities in the control part of the frame and $T_f$ is frame duration. We also show that, if the scheduling messages are transmitted in the data part of the frame, the worst case difference between the time a change in link bandwidth happens to the time a new schedule starts is $1.5D_{\max} + 2mT_f$, where $D_{\max}$ is the maximum return path TDMA delay in the network [5].

### A. Related Work

Research in TDMA scheduling was initiated by the seminal paper [7], where the authors provide an algorithm to find a set of link bandwidths and a feasible schedule for that set of bandwidths. However, the results of that paper cannot be applied in mesh networks since the algorithm is centralized and the network model in the paper ignores secondary link conflicts existing in mesh networks. In [8], the authors provide a distributed TDMA scheduling algorithm and they assume that secondary conflicts are removed with the use of multiple orthogonal channels. In order to account for communication difficulties with orthogonal channels, the authors propose an asynchronous TDMA protocol, which is inconsistent with the synchronized TDMA mesh protocols such as 802.16. In [9], the authors propose a scheduling algorithm in which secondary conflicts are resolved by reversing the direction of link transmissions. However, the algorithm only allows for one slot to be allocated to each link, which makes it impractical for mesh networks, where links require multiple time-slots, corresponding to QoS in the network.

In [10], authors provide a scheduling algorithm tightly coupled with a schedule activation algorithm. The algorithm uses a wave to allow an opportunity for every node to negotiate a conflict free schedule with its two hop neighbourhood. Once a new schedule is negotiated in the local neighbourhood, the wave moves down the network tree. The scheduling technique used in [10] allows multiple link transmissions in the same frame, which is not appropriate for mesh networks; 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. As we will show later, coupling scheduling with schedule activation would unnecessarily prolong iterative scheduling used in this paper.

## II. TDMA Network Model

The mesh network is using a time division multiple access (TDMA) MAC protocol [3]. In TDMA MAC protocols, the time is divided into slots of fixed duration, which are then grouped into frames. As in 802.16, the number of the current frame is known throughout the network. A fixed portion of each frame is dedicated to control traffic, while the rest of the slots are used for data traffic; we assume that $T$ slots are reserved for data traffic. The slots in the data frame are assigned by the network layer, based on the demands from the mesh routers. A scheduling algorithm establishes a common transmission schedule, which repeats in every frame until new traffic demands can be incorporated into the schedule.

We model the mesh 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 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. We use the convention that $v_1$ is the POP. All links are directional, so for a link $e_k \in E$, $f_t(e_k) = (v_i, v_j)$ means that traffic on the link is transmitted from $v_i$ to $v_j$.

We assume that a protocol external to the MAC protocol assigns bandwidth on each link in the topology, so that QoS is satisfied for all nodes. Since all network traffic originates or terminates at the point-of-presence (POP), the set of links with positive bandwidths form two disjoint routing trees rooted at the POP. One tree is for the uplink traffic and the other tree is for the downlink traffic. The two trees may also coincide as in 802.16 centralized scheduling protocol.

Link bandwith is assigned through the number of slots a link can use in a frame $d : E \to \mathbb{Z}_{[0,T]}$. The number of slots required to achieve the bandwidth $B_i$ on link $e_i$ can be found with:

$$d_i = \left\lceil \frac{B_i T_f}{b_i} \right\rceil, \tag{1}$$

where $\lceil \cdot \rceil$ denotes the ceiling of a real number, $b_i$ is the number of bits transmitted in each slot and $T_f$ is frame duration. The number of bits in each slot depends on the modulation chosen during the transmission in the slot.

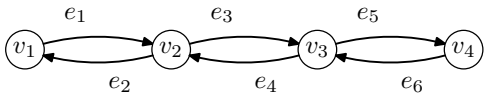
We assume that the signal-to-noise ratio of each link depends on the wireless channel alone and not other links in the network. This means that competing links do not transmit at the same time. Under this model of transmission, in TDMA networks, a receiver can only have one active link at any given time. We can construct a conflict graph based on the conflicts in the network; the vertices in the conflict graph correspond to links in the topology and the edges correspond to the conflicts. For example, for a four node network shown in Fig. 1a we construct a conflict graph in Fig. 1b. In this example, the links that do not interfere with each other are $e_1$ and $e_6$ and $e_2$ and $e_5$.

The nodes keep track of link conflicts with topological information about the two-hop neighbourhood of the node. This information can be obtained, directly, from the routing tables[1] or from the MAC neighbour table.[2] Given the two-hop topology around node $v_i$, the node can determine the set of conflicting links for each of its links and so provide them with a partial view of the conflict graph.

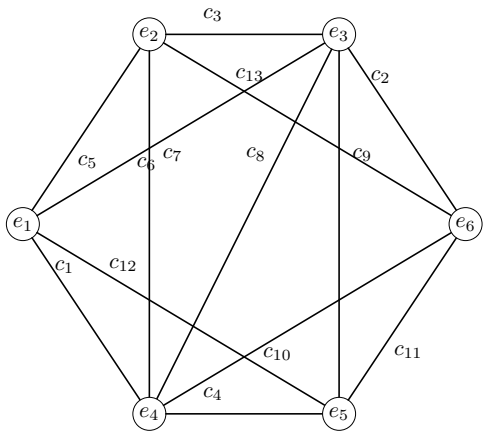
We use a ranking function $R : E \to \mathbb{Z}$ to specify the order of link transmissions. We use the convention that if $R_j > R_i$, $e_i$ transmits before $e_j$. We have shown in [5] that the ranking

[1]For example, OLSR keeps a two-hop neighbourhood [11].

[2]802.16 neighbour table keeps track of the two-hop neighbourhood.




(a) A four node chain




(b) Conflict graph for the four node topology

Fig. 1.

function is directly related to TDMA delay. TDMA delay occurs if the outbound link is scheduled to transmit before the inbound link. In [5], we have given a method to find the ranking with the minimum delay, as well as a heuristic to find rankings with small delay. In particular, we have shown that if for every return path $\mathcal{P}$ from the POP to a WAP, the ranking is assigned with:

$$R_k = \max\{R_k, R_l + 1\}, \forall e_k, e_l \in \mathcal{P} : e_l \rightsquigarrow e_k, \qquad (2)$$

where $e_l \rightsquigarrow e_k$ means that $e_l$ is the predecessor of $e_k$ on the return path $\mathcal{P}$, the return trip delay for all connections in the network is one frame. However, this ranking does not take advantage of spatial re-use. We have therefore suggested a heuristic that takes spatial re-use into account and found the heuristic ranking with:

$$R_i^H = R_i \pmod H, \qquad (3)$$

where $H$ is the amount of spatial re-use introduced in the network. With this heuristic, the maximum TDMA delay on all return paths in the routing tree is bounded by:

$$D_{\max} = \max_{\forall v_i \in V} \frac{|\mathcal{P}_i|}{H} T_f = \frac{2h}{H} T_f, \qquad (4)$$

where $|\mathcal{P}_i|$ is the number of hops on the return path from $v_1$ to the mesh node $v_i$, $T_f$ is frame duration, and since the longest path is also twice the height of the tree, $h$, we get the second part of the equality. Since finding the rank in this way can be easily distributed, we assume that the rank of each link is known in the link's two-hop neighbourhood.

### A. TDMA Schedules

We define the TDMA schedule as the pair of vectors $S(\boldsymbol{\pi}, \boldsymbol{d})$, where $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_m]^T$ are the starting times of

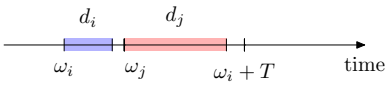



Fig. 2. Conflict-free Conditions

the links in the data part of the frame and $\boldsymbol{d} = [d_1, \ldots, d_m]^T$ is the duration of each transmission. The TDMA scheduling problem is to find a transmission schedule which is both *valid* and *conflict-free*. A valid transmission schedule assigns the number of slots allocated to the links due to QoS requirements, so by definition a schedule is valid if $\boldsymbol{d}$ corresponds to the assigned link durations. A conflict-free schedule ensures that transmissions of conflicting links do not overlap.

We have defined conditions for conflict-free scheduling in [5]; we briefly summarize those results here. First, we note that for each link $e_i$, it is sufficient to limit $\pi_i \in [0, T)$. Second, since the schedule repeats every $T$ slots, it is really a series of activation times $\Pi_i = \{\pi_i + z_i T, z_i \in \mathbb{Z}\}$. The conflict-free conditions for a schedule can be expressed in terms of points in the sequences $\Pi_i$, $\forall e_i \in E$. We have shown in [5] that a schedule is conflict-free if for any two conflicting links $e_i$ and $e_j$, such that $R_j > R_i$:

$$d_i \leq \omega_j - \omega_i \leq T - d_j, \quad 0 \leq \omega_j - \omega_i \leq T, \qquad (5)$$

where $\omega_i \in \Pi_i$ and $\omega_j \in \Pi_j$.

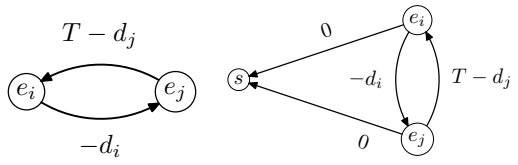
Fig. 2 shows why (5) is necessary for the schedule to be conflict-free. Full proof of necessity and sufficiency of (5) can be found in [5]. Since $R_j > R_i$, we assume that we are comparing the timing of $e_i$'s transmission to the first transmission of $e_j$ that follows it. Clearly, it is necessary that $\omega_j \geq \omega_i + d_i$ since $e_j$ cannot start its transmission before $e_i$ finishes. Also, the next transmission of $e_i$ should be after $e_j$ has finished its transmission, so $\omega_i + T \geq \omega_j + d_j$. The two cases can be combined into (5).

## III. Distributed TDMA Scheduling

In this section, we present a distributed TDMA scheduling procedure that finds schedules if nodes use local neighbourhood information only. First, we show that the scheduling problem is equivalent to finding shortest paths on an augmented conflict graph for the network. Second, we show how to implement the procedure with a distributed version of the Bellman-Ford algorithm.

A key observation, which allows us to find feasible schedules with minimum distance algorithms is that the set of feasible conditions (5) is a set of difference inequalities defined on the conflict graph for the network. This type of difference inequalities can be solved with any minimum distance algorithm [12]. Given a solution to the difference inequalities $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_m]$, a feasible schedule, $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_m]$, can be found by applying $\pi_i = \omega_i \pmod T$ for each $e_i \in E$, since we have $\omega_i = \pi_i + z_i T$ for some $z_i \in \mathbb{Z}$. This leads us to a procedure that finds TDMA schedules.

First, create a weighted and directed symmetrical conflict graph for the network $G_c(E, C, f_c)$, where the set of vertices




(a) Weights $R_j > R_i$ (b) Arcs to $s$

Fig. 3. Scheduling Graph Construction

is the set of all active links $E$, $C$ is the set of symmetrical arcs between the vertices in the graph, and $f_c : E \rightarrow C$ associates arcs with the vertices. The arcs in the graph correspond to conflicts between links, so each pair of conflicting links has two arcs (Fig. 3a). The cost of each arc is defined by the limits in the inequality (5) associated with the conflict, so for arc $c_k$, with $f_c(c_k) = (e_i, e_j)$:

$$l(c_k) = \begin{cases} -d_i, & \text{if } R_i < R_j \\ T - d_i, & \text{if } R_i > R_j. \end{cases} \tag{6}$$

This assignment of arc costs is shown in Fig. 3a for $R_j > R_i$.

Second, the conflict graph is a amended with an additional vertex $s$, which acts as a destination vertex for all the other vertices in the graph (Fig. 3b). The new vertex is connected to the original vertices in the conflict graph with zero length arcs pointing to it. If there are $r$ conflicts in the network, we denote each new arc with $c_{r+i}$ indicating that it points from $e_i$ to $s$. The addition of the new vertex and the new arcs create the scheduling graph $G_s(E_s, C_s, f_s)$, where $E_s = E \cup \{s\}$, $C_s = C \cup \{c_{r+1}, \ldots, c_{r+m}\}$, and

$$f_s(c_k) = \begin{cases} f_c(c_k) & \text{if } k \leq r \\ (e_i, s) & \text{if } k > r, k - r = i \end{cases} \tag{7}$$

Third, a TDMA schedule is found from the minimum distance from every vertex in the scheduling graph to $s$. This can be easily verified by noting that if $\omega_i$ and $\omega_j$ are the minimum distance from $e_i$ and $e_j$ and they are connected as in Fig. 3b. First, $\omega_j \leq w_i + T - d_j$ since either $e_i$ is on the minimum path to $s$ in which case there is an equality, or there is an alternate path which has smaller minimum distance than the path through $e_i$. By a similar argument it can be shown that $\omega_i \leq w_j - d_i$. Since the minimum distances in the scheduling graph have the conflict-free property, they can be used to derive a conflict free schedule with the modulo operation.

We note that the necessary and sufficient condition that shortest paths exist, and so a feasible schedule exists, is that there are no negative cycles in the scheduling graph [12]. We will assume in the rest of the paper that the bandwidth allocation algorithm has ensured that the bandwidth assignment is feasible, i.e. there are no negative cycles in the scheduling graph. Due to space restrictions, we do not include a negative cycle detection algorithm. However, we note that this is a well studied problem with both centralized [12] and distributed [13] solutions.

### *A. Iterative Schedule Construction*

In order to distribute schedule construction, we associate an independent scheduler with each link in the network. We denote a scheduler associated with link $e_i$ with $S_i$. Each scheduler resides on the node where its associated link originates. Since each scheduler corresponds to a vertex in the scheduling graph, they can find the shortest route to $s$ with the distributed Bellman-Ford algorithm as independent routers would. The scheduler's owner node facilitates the iterative algorithm by providing it procedures for communication to other schedulers and a table of its conflicting schedulers (schedulers it is directly connected to in the scheduling graph).

The communication between the schedulers is facilitated with the use of a reliable transmission protocol such as TCP. When a node receives a message request from one of its schedulers, it uses the two-hop neighbour table to find the node where the receiving scheduler resides and then forwards the message to that node using the reliable transmission protocol. Conversely, when a node receives a message for one of it's schedulers it forwards the message to that scheduler. We note that it takes at most two TDMA frames for any message to reach a peer scheduler.

The conflict table for a scheduler contains an entry for the scheduler and one entry for each of its conflicting schedulers. An entry in the conflict table corresponding to scheduler $S_j$ contains the rank of the link $R_j$ and the current distance to $s$ in the scheduling graph via that scheduler, $\omega_j$. The conflict table is initiated by the scheduler's owner node with the information available about the two-hop neighbourhood of the node. Initially, each conflicting entry $S_j$ in the conflict table for $S_i$ is initialized with $\omega_j = \infty$, while the entry for the scheduler is initialized to $\omega_i = 0$ to indicate that it is directly connected to $s$.

Each scheduler finds the shortest route to the vertex $s$ in the scheduling graph independently, using a modified version of the distributed Bellman-Ford algorithm [14]. Unlike, the original Bellman-Ford algorithm, which only uses one update message type, the version we present uses three types of messages. The `UPD` messages are sent by conflicting schedulers when their schedule changes, i.e. distance to $s$ is changed. Upon receiving the update message, the scheduler updates it's distance estimate to $s$ with the information in the message. If the estimated distance has changed, the scheduler sends the `UPD+ACK` message to the scheduler that initiated the update and a `UPD` message to other neighbouring schedulers; otherwise the scheduler sends an `ACK` message to the originator of the update. A scheduler is said to be locally scheduled if it has no unacknowledged updates. The network is scheduled when all schedulers are locally scheduled.

We introduce synchronization in the Bellman-Ford iteration with a rule that `UPD` and `UPD+ACK` messages are sent during even numbered frames. In the odd number frames, the only messages are the messages forwarded to schedulers two hops away in the mesh topology. Since this transmission schedule for the update messages is equivalent to performing all updates

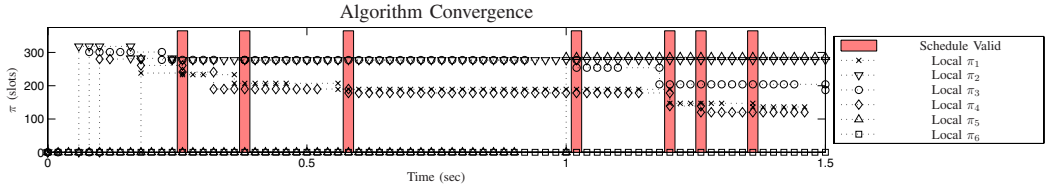



Fig. 4. Example Algorithm Convergence

and computations at the begining of even numbered frames, the algorithm is equivalent to the synchronized version of the Belman-Ford algorithm [14].

We implement the scheduling procedure with the OMNeT++ discrete time simulator [15]. We show an example of algorithm convergence in Fig. 4, where we used the topology from Fig. 1a and ranks assigned with $H = 4$. In the scenario, we use frame length of $T_f = 10$ms, $T = 365$ data slots per frame and slot duration of $25\mu$s long, corresponding to a subset of parameters specified in 802.16. The link change times are a Poisson arrival process with an average interarrival time of 25ms and link durations are picked from a uniform distribution with the mean of 50 slots. Before a link change is requested, we check if that link duration will result in a feasible schedule with a centralized negative cycle detection algorithm [12]. If a link duration is infeasible we decrease it until it becomes feasible. Fig. 4 shows the locally converged schedule times for each of the links and it also shows when the algorithm is globally converged.

In general, a scheduling algorithm based on the asynchronous distributed Bellman-Ford algorithm convergence after a finite and possibly exponential number of steps [14]. However, the iterative scheduling procedure in this paper converges in a linear number steps since we take advantage of the synchronization in TDMA networks. We note the scheduling procedure is equivalent to the synchronized Bellman-Ford algorithm because all updates are received before calculations are done in even numbered frames. The scheduling procedure therefore converges in the number of steps equal to the longest path in the scheduling graph. Since there are $m + 1$ vertices in the scheduling graph, the scheduling algorithm converges in $2m$ frames.

We use the the chain topology to further show the convergence properties of the scheduling algorithm. We have used the same setup as in the previous experiment, except that the link times are Poisson with an average of 100ms. In the chain topology the number of links in the network grows linearly with the number of nodes and so does the worst case convergence time of the algorithm (Fig. 5) . However, as we show in Fig. 5, as the number of nodes grows the average time for the schedule to converge does not increase with the number of nodes. The reason for this is that very few of the link changes result in schedule changes for the entire network. This behaviour is due to the nature of shortest path tree in constructed by the Bellman-Ford algorithm; only links downstream of the change need to update their schedules. We also observe that the $H$ heuristic does not significantly affect the convergence properties of the algorithm.

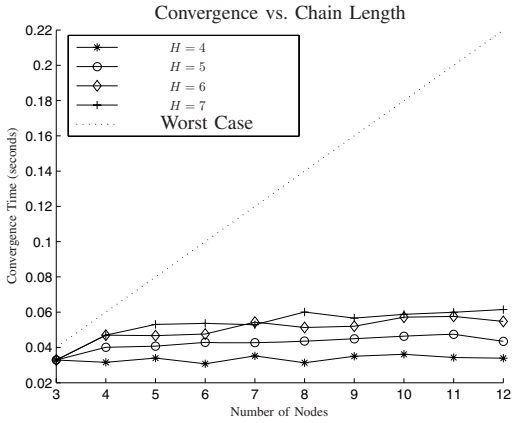



Fig. 5. Average Convergence Time

## IV. Feasible Schedule Detection

In the previous section, we have shown that a feasible schedule can be found with a distributed iterative procedure. However, in order for the scheduling algorithm to be practical, the network needs a mechanism to detect when a schedule has converged and a new schedule should take place. We use a wave based termination detection procedure for this purpose [6].

The schedule detection procedure is initiated by the POP after a previous detection procedure has terminated. The POP starts the procedure by multicasting a `SDA-DOWN` message through the routing tree for the network.[3] As the wave of `SDA-DOWN` messages traverses the network it resets the state of each node as `done`. When a leaf node is reached, the wave reverses and the leaf node sends a `SDA-UP` message back to the POP. As the nodes transmit the `SDA-UP` messages, they also record their schedule, so that it is not lost if there are further link changes. The `SDA-UP` messages collect the state of the nodes in the tree, so when the POP receives `SDA-UP` from all of its children, it can decide if the scheduling algorithm has converged.

The validity of the algorithm depends on the transmission rules used to forward the `SDA-UP` messages and the payload of the messages. Each node in the tree transmits a `SDA-UP`

[3]The termination detection procedures requires that the uplink and downlink trees coincide.

message to its parent when it has received a SDA-UP message from each of its children and it has detected that all of its schedulers are locally scheduled. The payload of the SDA-UP messages indicates the combined state of the node and all other nodes in the subtree rooted at the node. A node transmits a SDA-UP message with the payload done only if it is in the done state and all the tokens it has collected carried done. Otherwise, the node transmits a SDA-UP message with the payload working. In order for the algorithm to work correctly, a node must change its state to working if it transmits a UPD or UPD+ACK message while waiting for tokens from its children. If a node enters the working state, it remains in it even when it becomes locally scheduled.

The POP detects that the distributed scheduling has finished when it receives SDA-UP messages with the payload done from all of its children. If the POP receives one or more messages with the payload working, the POP restarts the algorithm with the SDA-DOWN messages. The validity of this termination algorithm is shown in [6].

After detecting that a schedule has converged, the POP sends a SDA-DONE message to indicate that the schedule has converged and the new schedule should start. The SDA-DONE message carries the time when the new schedule should become valid. The validity time of the schedule depends on the broadcast mechanism used to transmit SDA-DONE messages. If SDA-DONE messages are carried with the regular traffic in the data sub-frame, the schedule becomes valid after $h/H$ frames, where $h$ is the height of the routing tree and $H$ is used in the scheduling heuristic. On the other hand, if the messages are broadcast in the control part of the frame, the schedule becomes valid in $n/\kappa$ frames, where $n$ is the number of mesh nodes and $\kappa$ is the number of distinct nodes that can broadcast in the control part of each frame.[4]

The difference between the detection time and the time when the schedule is globally converged depends on the way SDA-UP and SDA-DOWN messages are transmitted. The worst case delay in detection happens when the last node to locally schedule itself is the POP. So, if the messages are transmitted in the data sub-frame, the worst case detection delay is given by $D_{\max}$, (4), since $D_{\max}$ is the longest time it takes to get a response from all the nodes. On the other hand, if the messages are carried in the control part of the frame, the schedule becomes valid in $2n/\kappa$ frames.

Combining the results for the detection of a feasible schedule and the validity time of the schedule, we conclude that the schedule becomes valid $3h/H$ frames after it converges and the detection messages are carried in the data sub-frame, or the schedule becomes valid in $3n/\kappa$ frames if the messages are carried in the control sub-frame.

Another way to activate a new schedule is to combine our scheduling algorithm with the schedule activation method. In this case, schedulers transmit updates to their conflicting links as the wave moves down the tree and perform calculations as the wave moves up the tree. This is also equivalent to the synchronized Bellman-Ford algorithm, so in the worst case the schedule converges after $mD_{\max}$ seconds. With this method, the new schedule can always be activated $mD_{\max}$ seconds after the last change in link bandwidths, but this is much longer than with the method proposed in this paper.

[4]This is the case in the 802.16 centralized scheduling protocol, where the order of scheduling message broadcasts in the control sub-frames is defined by the routing tree.

## V. Conclusion

We have presented a distributed scheduling algorithm that can be used in TDMA based mesh networks using protocols such as 802.16 and 802.11s. In our algorithm, each link uses its partial view of the conflict graph to run a the distributed Bellman-Ford independently of other links in the network. The algorithm includes an independent wave based termination procedure that detects when the scheduling has converged for all links. The termination procedure also provides a mechanism to notify all nodes when the new schedule should be activated. We have used analysis and simulations to show that the scheduling algorithm converges quickly. This paper extends our work in [5] where we have presented centralized scheduling algorithms for mesh networks.

## References


[1] M. Chee, "The business case for wireless mesh networks," www.nortelnetworks.com, December 2003. [Online]. Available: http://www.nortelnetworks.com

[2] S. Briedenbach, "Building boom for Wi-Fi networks," http://www.networkworld.com/research/2006/030606-municipal-wi-fi.html, March 2006.

[3] "IEEE standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems," 2004.

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

[5] P. Djukic and S. Valaee, "Link scheduling for minimum delay in spatial re-use TDMA," in *Proceedings of INFOCOM*, 2007.

[6] R. W. Topor, "Termination detection for distributed computations," *Information Processing Letters*, vol. 18, no. 1, pp. 33–36, January 1984.

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

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

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

[10] J. Shen, I. Nikolaidis, and J. J. Harms, "A DAG-based approach to wireless scheduling," in *ICC*, 2005, pp. 3142–3148.

[11] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626 (Experimental), Oct. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3626.txt

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

[13] K. Chandy and J. Misra, "Distributed computation on graphs: Shortest path algorithms," *Communications of the ACM*, vol. 25, no. 11, pp. 833–827, 1982.

[14] D. Bersekas and R. Gallager, *Data Networks*. Prentice Hall, 1992.

[15] A. Vargas, "OMNeT++ Discrete Event Simulation System User Manual," March 2005.