

TDMA Scheduling Algorithms for Sensor Networks

Sinem Coleri Ergen and Pravin Varaiya
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720
Email: {csinem, varaiya}@eecs.berkeley.edu

July 2, 2005

Abstract

In developing algorithms for scheduling TDMA transmissions in multi-hop networks, the problem is usually to determine the smallest length conflict-free assignment of slots where each link or node is activated at least once. This is based on the assumption that there are many independent point-to-point flows in the network. In sensor networks however often data are transferred from the sensor nodes to a few central data collectors. The scheduling problem is therefore to determine the smallest length conflict-free assignment of slots during which the packets generated at each node reach their destination. We show that this problem is NP-complete. We first propose two centralized heuristic algorithms for solving the problem: One is based on direct scheduling of the nodes, node based scheduling, whereas the other is based on scheduling the levels in the routing tree before scheduling the nodes, level based scheduling. The performance of these algorithms depends on the distribution of the nodes across the levels. We then propose a distributed algorithm based on the distributed coloring of the nodes, that increases the delay by a factor of 10 – 70 over centralized algorithms for 1000 nodes. We also obtain upper bound for these schedules as a function of the total number of packets generated in the network.

1 Introduction

Wireless sensor networks have been proposed for a wide range of monitoring applications such as traffic and seismic monitoring, and fire detection [8]. Such networks consist of a group of nodes, with sensing, signal processing and wireless communication capabilities and limited battery energy. The nodes must quickly report the results to a data collection node or access point. Since the nodes are battery-powered, the medium access control (MAC) protocol is critical in determining net-

work lifetime.

Proposed MAC protocols for sensor networks provide either contention based access or time division multiple access (TDMA). The former, e.g. IEEE 802.11 [10], consume more energy than TDMA protocols because they waste energy in collisions and idle listening. Moreover, they do not give delay guarantees. TDMA protocols are more power efficient since nodes in the network can enter inactive states until their allocated time slots. They also eliminate collisions and bound the delay. For example, the TDMA protocol for a traffic monitoring network described in [6] has a lifetime of 1,200 days compared with ten days using the IEEE 802.11 protocol.

The main task in designing a TDMA schedule is to allocate time slots depending on the topology and the node packet generation rates. A proper schedule not only avoids collisions by silencing the interferers of every receiver node in each time slot but also minimizes the number of time slots hence the latency: The larger latency may require a higher data rate (and hence higher energy consumption) to satisfy a deadline [3]. We therefore try to find a TDMA schedule that minimizes the number of time slots.

TDMA algorithms consider either one-hop or multi-hop scheduling. The former are for networks in which the nodes are one hop away from the base station [3, 7], and allocate time slots in the reverse channel depending on allocation request and deadline of the nodes. Because the base station is the common receiver of the transmissions, only one node can transmit in a slot. In some sensor networks however direct transmission from all sensor nodes to the base station may not be feasible nor power efficient [5].

Multi-hop TDMA scheduling is more challenging than one-hop scheduling because spatial reuse of a time slot may be possible: More than one node can transmit at the same time slot if their receivers are at non-conflicting parts of the network. There are two types of conflicts, namely, primary conflict and

secondary conflict. A primary conflict occurs when a node transmits and receives at the same time slot or receives more than one transmission destined to it at the same time slot. A secondary conflict occurs when a node, an intended receiver of a particular transmission, is also within the transmission range of another transmission intended for other nodes. In the context of TDMA, the problem is to determine the smallest length conflict-free assignment of slots where each link or node is activated at least once [12]. Previous work on scheduling algorithms focus on either decreasing the length of schedules [12, 9, 2] or distributed implementation [13, 4, 19, 20, 15].

Previous scheduling algorithms activating each link or node at least once during a TDMA frame is based on the assumption that there are many independent point-to-point flows in the network. In sensor networks however often data are transferred from the sensor nodes to a few central data collectors. In traffic monitoring [18], for example, the nodes sense the passage of vehicles at several freeway locations or at an intersection, and transmit the data to the access point on the side of the freeway or intersection. The packets are transferred to the access point over the routing tree in multiple hops. The problem therefore is to determine the smallest length conflict-free assignment of slots during which the packets generated at each node reach the access point over the routing tree.

To the best of our knowledge, this is the first work that exploits application-specific characteristics of sensor networks to improve the delay. This problem has not been discussed before in the literature. Since each packet is relayed on the routing path from the originating sensor node to the access point, the problem requires considering precedence relations: If the packet follows the routing path $(k, k-1, \dots, 2, 1)$, node j should not be scheduled before node i for that packet if $j \leq i$. Precedence constrained graphs have been studied in the context of allocating the tasks in the precedence constrained task graphs to the processors in the processor network so that the schedule length is minimized [14]. In this problem however the tasks are already assigned to the processors since a packet at node j has to be processed by node j . The difficulty on the other hand comes from the requirement of eliminating primary and secondary conflicts in the processor network.

The rest of the paper is organized as follows. The transmission and network model is described in Section 2. Section 3 describes the scheduling problem and proves that it is NP-complete. We then propose two heuristic centralized algorithms for solving the problem: The one in Section 4 is based on direct scheduling of the nodes whereas the other is based on scheduling the levels in the routing tree before scheduling the nodes as explained in Section 5. Section 6 describes a token based distributed scheduling algorithm. The algorithms

are analyzed in Section 7. Simulations are given in section 8. Section 9 collects some conclusions.

2 Network and transmission model

We consider a network comprising a single access point (AP) and several sensor nodes that periodically generate data, possibly at different rates, for transfer to the AP. Links are assumed to be bidirectional. This is required for proper functioning of network protocols such as distributed Bellman-Ford algorithms [16]. Bidirectionality is achieved if all sensor nodes transmit at the same power. Differences in actual transmission power due to the hardware differences can be compensated by setting up links based on received signal strength as explained in [6].

The network is represented by a graph $G = (V, E)$. V is the set of nodes, including the access point AP as node 1. $N = |V|$ is the number of nodes in G . The (undirected) edges $E \subset V \times V$ are the (transmission) links to be scheduled. The graph forms a *tree*. All traffic is destined for AP , so every data packet at a node is forwarded to the node's parent in the tree rooted at the AP.

A node may interfere with another node, so these nodes should not transmit simultaneously. The *interference* graph $C = (V, I)$ is assumed known. $I \subset V \times V$ is the set of edges such that $(u, v) \in I$ if either u or v can hear each other or one of them can interfere with a signal intended for the other (even if they cannot hear each other). So, if u is transmitting, v should not be scheduled to receive from another node at the same time.

The *conflict* graph corresponding to $G = (V, E)$ and $C = (V, I)$ is called $GC = (V, EC)$. In GC , each node $i \in V$ corresponds to the link $(i, p_i) \in E$ where p_i is the parent of node i in the routing tree G rooted at AP. EC comprises the edges between node pairs in G that should not transmit at the same time. It is generated by taking into account the primary and secondary conflicts described in Section 1. EC contains two kinds of edges. First, if $(i, j) \in E$, $(i, j) \in EC$, because a parent node and a child node cannot transmit at the same time. Second, if $(i, j) \in I$ or $(i, j) \in E$ and c_j is a child of j in G , $(i, c_j) \in EC$: Because i and j interfere, if i is transmitting, the child c_j of j cannot transmit at the same time because j would hear from both i and c_j .

A *scheduling frame* is the time duration that starts when each node has generated an integer number of packets and ends when all these packets have reached AP . It is divided into time slots. A slot is long enough to transmit one data packet

plus a guard interval to compensate for synchronization errors. A *schedule* assigns one or more time slots to each edge in G or, equivalently, to each node in GC . A node u may receive a packet from its child v during a time slot assigned to $(v, u) \in E$ or to node $v \in V$ since its parent u is already known.

We use the following notation. The *distance* $d(u, v)$ between nodes u and v is the number of edges in the path between them in G ; and a node u is at *level* k if it is at distance k from AP .

3 The scheduling problem

Each node of G (except AP) generates a positive integer number of packets at the beginning of the scheduling frame. Given the interference graph C , the scheduling problem is to find a minimum length frame during which all nodes can send their packets to AP .

Theorem 1 *The scheduling problem is NP-complete.*

Proof We reduce the NP-complete problem of finding the chromatic number of a graph to the scheduling problem.¹ Let $GP = (VP, EP)$ with $VP = \{v_1, \dots, v_N\}$ be an instance of a graph whose chromatic number we want to find. We first construct a conflict graph $GC = (V, EC)$. First, GC includes all the nodes and edges of GP . Next, for each node v_i , add another node w_i . Then add edges $(w_i, w_j), (v_i, w_j) \in EC$ for all i, j . Lastly add another node AP and edges (AP, w_i) for all i . See figure 1.

The conflict graph GC is such that if w_i is active, none of the nodes in $V \setminus \{w_i\}$ can be active at the same time. Also, if v_i is active, none of the nodes w_j or the conflicting nodes from VP , determined by the edges EP , can be active.

We now construct a tree $G = (V, E)$ and an interference graph $C = (V, I)$ whose conflict graph is $GC = (V, EC)$. The edges of the tree are $E = \{(AP, w_i), (w_i, v_i) \mid 1 \leq i \leq N\}$. Because AP is a parent of w_i , $(w_i, AP) \in EC$ for all i ; moreover $(w_i, w_j) \in EC$ for all i, j , because they have the same parent, AP . And $(v_i, w_i) \in EC$ because w_i is the parent of v_i .

Let I consist of edges (v_i, AP) for all i , and $(v_i, w_j), (v_j, w_i)$, whenever $(v_i, v_j) \in EC$. Since $(v_i, AP) \in I$ and $(w_j, AP) \in E$, $(v_i, w_j) \in EC$ for all i, j . Lastly, if $(v_i, w_j) \in I$ and $(v_j, w_i) \in I$, $i \neq j$, $(v_i, v_j) \in EC$ because the parent of one

¹The chromatic number of a graph G is the smallest number k such that G is k -colorable. G is k -colorable if its vertices can be colored using k different colors in such a way that adjacent vertices have different colors.

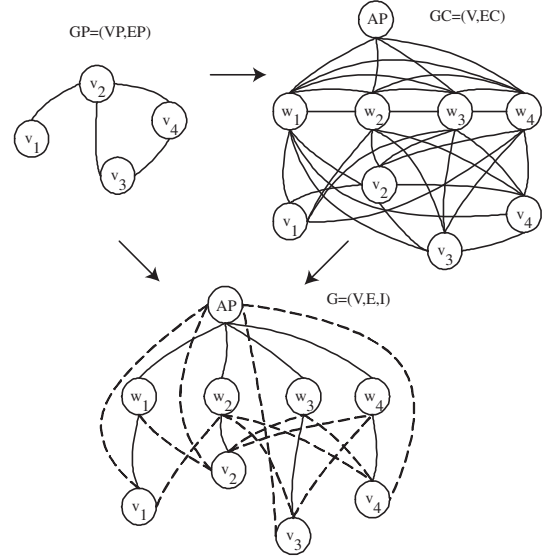


Figure 1: Transformation from $GP = (VP, EP)$ to $GC = (V, EC)$ and then to a tree network $G = (V, E)$ (solid lines belong to E) with interference graph $C = (V, I)$ (dashed lines belong to I).

of them is interfered by a transmission of the other. Thus GC is indeed the conflict graph corresponding to the tree graph G and interference graph C .

Consider the minimum schedule length for GC such that each node $v_i, w_i, 1 \leq i \leq N$, has one packet destined for AP . A packet in w_i takes the path (w_i, AP) and a packet in v_i takes the path (v_i, w_i, AP) . Because each w_i conflicts with the nodes $w_j, j \neq i$ and all nodes v_i , it takes N slots to transmit the packets generated at level one to AP , independently of the rest of the network. Also, when the N packets from level two arrive at level one, it takes another N slots to forward them to AP .

Thus to minimize the time to transmit all packets to AP , we must minimize the time to transmit the packets from level two to level one. But the conflict graph at level two is determined by the original graph GP , so the minimum scheduling time is exactly $2N+c$, where c is the chromatic number of the original graph GP . \square

The scheduling problem is difficult because many subsets of non-conflicting nodes are candidates for each time slot, and the subset selected for transmission in one slot affects the number of transmissions in the next time slot, as some schedulable nodes may not have any packets to transmit because of the subset selected in the previous slot.

Lemma 1 Assume that node $i \in V$ has generated g_i packets to transmit. The minimum schedule length is at least $\sum_{i \in V} g_i$.

Proof AP can receive at most one packet in each slot, so at least $\sum_{i \in V} g_i$ slots are needed for all packets to reach AP . This gives the lower bound. \square

4 Node Based Scheduling Algorithm

The node based scheduling algorithm has two parts. In the first part, we color the conflict graph $GC_c = (V_c, EC_c)$ where $V_c = V \setminus \{1\}$, $EC_c = EC \setminus N_1$ and $N_1 = \{(i, j) | i = 1\}$. In the second part, we schedule the links in the original network, $(u, v) \in E$, based on this coloring.

4.1 Coloring the network

Any algorithm can be used to color the conflict graph GC_c such that nodes i and j are assigned different colors if $(i, j) \in EC_c$. Computing the chromatic number of a graph is NP-complete. Incremental methods appear to be the heuristic choice of vertex coloring [13]: Vertices are colored sequentially with the colors chosen in response to colors already assigned in the vertex's neighborhood. These methods vary in how the next vertex is selected and how it is assigned a color.

Figure 2 shows such a heuristic coloring algorithm. At the beginning of the algorithm, the nodes are ordered according to some criterion, e.g. non-increasing order of degree since high degree vertices have more color constraints and so are more likely to require an additional color if inserted late. This algorithm assigns a slot to node i in $O(i)$ steps, so the running time of this algorithm is $O(|V|^2)$.

4.2 Scheduling the Network

A *superslot* in node based scheduling algorithm is a collection of consecutive time slots such that each node with at least one packet at the beginning of the superslot transmits at least one packet during the superslot. Because two nodes assigned the same color can transmit at the same time, the number of slots in a superslot is at most equal to the total number of colors used for coloring the network.

The algorithm is given in Figure 3. After determining the

```

Input:  $V_c = \{2, 3, \dots, N\}$ , conflict graph
 $GC_c = (V_c, EC_c)$ .
Output: One color assigned to each node
 $\{(2, c_2), (3, c_3), \dots, (N, c_N)\}$  in which
 $c_i \in \{1, 2, \dots, M\}$  and  $M$  is the number of
colors.
begin
  Order the nodes as  $(n_1, n_2, \dots, n_{N-1})$ 
  for  $l = 1$  to  $N - 1$ 
     $i = 1$ 
    while  $(\exists j$  assigned to color  $i$  st.
 $(j, n_l) \in EC_c)$ 
       $i = i + 1$ 
    assign color  $i$  to  $n_l$ 
end

```

Figure 2: Assigning one color to each node in the network.

nodes corresponding to the current time slot from the network coloring, additional nodes assigned to other colors are added as long as the resulting set is non-conflicting. The running time of the algorithm is then $O(ld_{max}|V|)$, where d_{max} is the maximum degree of a node in GC and l is the total number of slots in the schedule.

Two examples are given in Figures 6 and 7. $G = (V, E)$ and $C = (V, I)$ are shown on the left with the resulting $GC = (V, EC)$. The nodes are ordered based on their degrees in GC for the coloring, which are $(s3, s2, s4, s5, s6, s7, s1)$ and $(s2, s3, s5, s6, s1, s4)$ in Figures 6 and 7 respectively. The resulting schedules are shown in part-(a) of the figures.

5 Level Based Scheduling Algorithm

The level based scheduling algorithm has three parts. In the first part, we obtain a linear network $GL = (VL, EL)$ with interference graph $CL = (VL, IL)$ resulting in conflict graph $GCL = (VL, ECL)$ corresponding to the original network. In the second part, we color this linear network. In the third part, we schedule the links in the original network, $(u, v) \in E$, based on the coloring of the linear network.

5.1 The linear network

If the original tree network has depth N , the linear network $GL = (VL, EL)$ has nodes $VL = \{v_1, \dots, v_N\}$ with node

```

Input: Graph  $G = (V, E)$  with conflict graph  $GC = (V, EC)$ , color assignment of the nodes  $V_c$  using  $M$  colors.
Output: Transmission schedule for nodes of  $G$ .
begin
  while (at least one packet has not reached  $AP$ )
    for  $s = 1$  to  $M$ 
       $set_s =$  set of nodes corresponding to color  $s$  with at least one packet
       $T = set_s$ 
      if  $T \neq \emptyset$ 
         $set_{os} =$  set of nodes not corresponding to color  $s$  with at least one packet
        for each node  $k \in set_{os}$ 
          if  $(k, j) \notin EC \forall j \in T$ 
             $T = T \cup \{k\}$ 
          assign current slot to set  $T$ 
          update the place of the packets
        end
      end
end

```

Figure 3: Node Based Scheduling Algorithm.

v_l corresponding to all nodes at level l in the original network and edges $(v_i, v_{i+1}) \in EL$ for $1 \leq i < N$. The interference graph $CL = (VL, IL)$ includes edge (v_j, v_l) if there is an interference edge between a node at level j and any node at level l in the original network for $j, l \geq 1$. The resulting conflict graph $GCL = (VL, ECL)$ thus includes edge (v_j, v_l) if the transmissions of a node at level j and a node at level l conflict in the original network. The algorithm in figure 4 finds EL, CL and ECL . Its running time is $O(|V|^2)$.

5.2 Coloring the linear network

Any coloring algorithm can be used to color the conflict graph of the linear network $GCL = (VL, ECL)$. The algorithm given in Figure 2 can be used for this purpose with $V_c = VL$ and $GC_c = GCL$ as input, and one color assigned to each node in VL and the number of colors, M , as output.

5.3 Scheduling the original network

If nodes v_i, v_j in the linear network are assigned the same color, they do not interfere. By construction of the linear network any two nodes in the original network, one chosen from

```

Input:  $(V, E, I, EC)$ .
Output:  $(VL, EL, IL, ECL)$ .
begin
  add node  $v_1$  to  $VL$ 
   $l = 2$ 
  while  $l \leq levelOfTree$ 
    add node  $v_l$  to  $VL$ 
    add edge  $(v_{l-1}, v_l)$  to  $EL$ 
    If  $\exists(u, v) \in I(EC)$  with  $u$  at level  $l$  and  $v$  at level  $j$  satisfying  $j < l$ 
      add edge  $(v_j, v_l)$  to  $IL(ECL)$ 
     $l ++$ 
  end

```

Figure 4: Algorithm to find linear network corresponding to original network.

level i and the other from level j , can transmit at the same time.

A *superslot* in level based scheduling algorithm is a collection of consecutive time slots such that each level of the tree with at least one packet at the beginning of the superslot forwards at least one packet to the lower level during the superslot. Because two nodes at different levels assigned the same color can transmit at the same time, the number of slots in a superslot is at most equal to the total number of colors used for coloring the linear network.

The algorithm is given in Figure 5. After determining the levels corresponding to the current time slot from the linear network coloring, a nonconflicting set of nodes at these levels that have packets to transmit are selected for transmission. Additional nodes from other levels are then added as long as the resulting set is non-conflicting. The running time of the algorithm is $O(ld_{max}|V|)$, where d_{max} is the maximum degree of a node in GC and l is the total number of slots in the schedule.

Two examples are given in Figures 6 and 7. $G = (V, E)$ and $C = (V, I)$ are shown on the left with the resulting $GCL = (VL, ECL)$. The levels are in increasing order for coloring. The ordering does not affect the number of colors used in linear network nor the schedule length for these examples. The resulting schedules are shown in part-(b) of the figures.

Figure 6 illustrates a topology where level based scheduling performs better than node based scheduling whereas Figure 7 illustrates a network where node based scheduling outperforms level based scheduling. Figure 6 demonstrates the ad-

vantage of level based scheduling in balancing the movement of packets across the network in a network of higher density of the packets at high levels. In topologies of equal density of the packets across the network or higher packet density at low levels, giving equal chance to the nodes balances the movement of packets as shown in Figure 7.

Input: Graph $G = (V, E)$ with conflict graph $GC = (V, EC)$, color assignment of the corresponding linear network GCL using M colors.

Output: Transmission schedule for nodes of G .

```

begin
  while (at least one packet has not reached AP)
    for  $s = 1$  to  $M$ 
       $set_s =$  set of levels corresponding to color  $s$ 
       $T = \emptyset$ 
      for  $j = 1$  to  $|set_s|$ 
         $T = T \cup \{ \text{a nonconflicting set of nodes from level } set_s(j) \text{ with at least one packet} \}$ 
        if  $T \neq \emptyset$ 
           $set_{os} =$  set of levels not corresponding to color  $s$ 
          for each node  $k$  belonging to a level in  $set_{os}$ 
            if  $(k, j) \notin EC \forall j \in T$ 
               $T = T \cup \{k\}$ 
            assign current slot to set  $T$ 
            update the place of the packets
      end
end

```

Figure 5: Level Based Scheduling Algorithm.

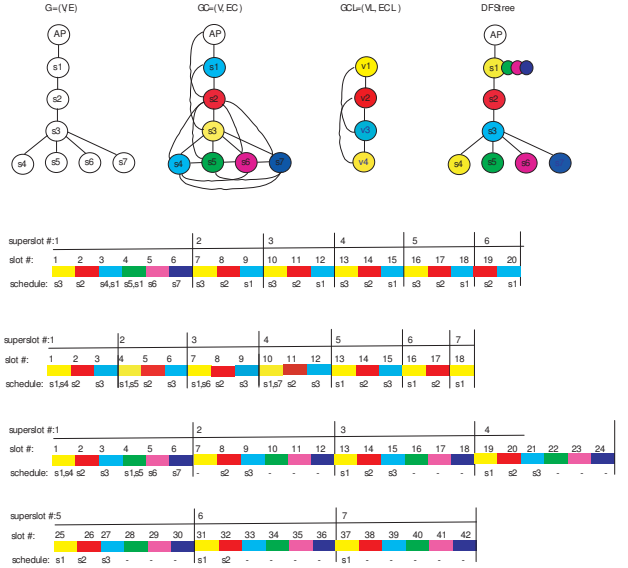


Figure 6: An example network where level based scheduling performs better than node based scheduling. a) Schedule for node based scheduling algorithm. b) Schedule for level based scheduling algorithm. c) Schedule for distributed scheduling algorithm.

6 Distributed Implementation

The node based and level based scheduling algorithms described above require complete topology information, and there are two options for implementation. The first option is to send the topology information to a central controller, which then performs the slot assignment and sends it back to the nodes in the network. The second option is that each node learns the entire network topology and executes the algorithm independently to produce identical schedules. Both options may require a lot of communication among the nodes, and may become inappropriate for large networks.

Distributed algorithms, in which the schedules of the nodes are generated based on the local topology information of the nodes, are preferred in large networks due to their scalability. However, it is very hard to obtain a distributed version of node

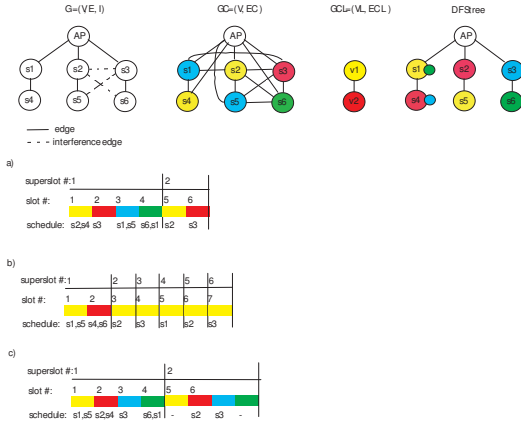


Figure 7: An example network where node based scheduling performs better than level based scheduling. a) Schedule for node based scheduling algorithm. b) Schedule for level based scheduling algorithm. c) Schedule for distributed scheduling algorithm.

based and level based scheduling algorithms. The main reason is that these algorithms check whether the nodes that are potential transmitters in the current slot have any packets and skip that slot otherwise. Another reason is that the algorithms schedule the nodes in other colors if some of the nodes that are potential candidates for the current slot do not have any packets. These however may not be performed in their distributed version since the nodes cannot know how many packets their interferers have due to the lack of knowledge of global topology information.

To get an idea of the performance of a distributed algorithm, we propose a simple algorithm based on the distributed coloring of the network similar to the one described in [13]. Notice that the conflicting nodes, the nodes that have edge between them in GC , are either one hop or two hops away from each other in the graph $G_u = (V, E \cup I)$. Assume the nodes i and j can transmit to each other if $(i, j) \in E \cup I$, e.g. this can be performed by increasing the transmission range of the nodes [6]. We also assume that all the transmissions during the generation of the schedules are successful, which can be guaranteed by an acknowledgement. The nodes first learn about all of their one hop and two hop neighbors in G_u and their parents so that they can determine their interferers in GC .

Similar to the algorithm described in [13], the color assignment is performed in two stages. During the first stage of the algorithm, each node picks one slot for transmission in the order of the traversal of the depth first search (DFS) [11] of the graph G . In the second stage, the DFS is repeated and now

each node picks as many of the remaining colors as it can for transmission. At both stages, the nodes send this information to their one-hop and two-hop neighbors in G_u so that all their interferers in GC learn about the assignment.

The DFS traversal starts with a TOKEN message generated at the AP. Upon receipt of the token, the node performs the color assignment and then sends this information to its one-hop and two-hop neighbors in G_u . It then sends the token to each of its neighbors in G who have not received the token yet. Once it finds that all its neighbors have received the token, it sends the token back to its parent, which is the node from which it receives the token for the first time. At the end of the traversal, the token carries the information of the number of colors used in the network back to the AP.

This distributed algorithm turns out to be the distributed version of the color assignment algorithm shown in Figure 8. Once the colors are assigned to each node, the nodes only transmit in the assigned timeslots assigned to these colors if they have a packet to transmit.

The total number of token transmissions is $O(|E|)$ at each stage and the total number of transmissions for distributing the color assignments is $O(d_{max}|V|)$, in which d_{max} is the maximum degree of the nodes in G_u .

Two examples are given in Figures 6 and 7. The coloring of the nodes are shown on the top right of the figures. The colors of the nodes are assigned at the first stage whereas the colors of the small circles next to the nodes are assigned at the second stage. The DFS traversal order are $(s_1, s_2, s_3, s_4, s_5, s_6, s_7)$ and $(s_1, s_4, s_2, s_5, s_3, s_6)$ in Figures 6 and 7 respectively. The resulting schedules are shown in part-(c) of the figures.

7 Analysis of the Algorithms

We consider four cases:

Case 1: The tree graph $G = (V, E)$ is linear, that is each node $u \in V$ has at most one child. The interference graph $C = (V, I)$ is such that $I = \emptyset$.

Case 2: The tree graph $G = (V, E)$ is general. The interference graph $C = (V, I)$ satisfies the ancestor property, that is, there do not exist u, v, b such that $(u, v) \in I$ and $|d(u, b) - d(v, b)| > 1$. This represents the case where shortest path routing is used with the cost of each path being equal to the number of nodes on that path and only nodes that can hear each other can interfere, which is the assumption of previously proposed TDMA scheduling algorithms.

```

Input:  $V = \{1, 2, \dots, N\}$ ,  $G = (V, E)$ ,
 $G_u = (V, E \cup I)$ , conflict graph
 $GC_c = (V_c, EC_c)$ .
Output: Color assignment of the nodes in  $V_c$  such
that each color corresponds to a maximal
nonconflicting set in  $GC_c$ .
begin
  Order the nodes as  $(n_1, n_2, \dots, n_N)$  in DFS
traversal of  $G$ 
  for  $l = 1$  to  $N$ 
    if  $n_l \neq 1$ 
       $i = 1$ 
      while  $(\exists j$  assigned to color  $i$  st.
 $(j, n_l) \in EC_c)$ 
         $i = i + 1$ 
      assign color  $i$  to  $n_l$ 
   $M$  is maximum assigned color
  for  $l = 1$  to  $N$ 
    if  $n_l \neq 1$ 
      for  $i = 1$  to  $M$ 
         $T$  is the set of nodes assigned to color
 $i$ 
        if (no node  $j \in T$  st.  $(j, n_l) \in EC_c$ )
          add color  $i$  to the color set of  $n_l$ 
end

```

Figure 8: Distributed Scheduling Algorithm.

Case 3: The tree graph $G = (V, E)$ is general and the interference graph $C = (V, I)$ is such that the maximum difference between the levels of two interfering nodes is K .

Case 4: The tree graph $G = (V, E)$ and the interference graph $C = (V, I)$ are both general.

Theorem 2 *Assume that each node has one packet to transmit. For level based scheduling algorithm, in cases 1 and 2 the maximum length of the frame is $3|V| - 3$ time slots; in case 3 it is $(K + 2)(|V| - 1)$; and in case 4 it is $\alpha(|V| - 1)$, in which α is the number of colors used in the linear network corresponding to G and C .*

Proof *Case 1.* If the tree graph G is linear and the interference graph C satisfies $I = \emptyset$, the corresponding linear tree interference graph CL also satisfies $IL = \emptyset$. It is easy to see that this linear tree can be colored optimally with three colors when the number of levels is more than two. The colors are assigned in a round robin fashion starting with the node at level 1.

At the beginning of the frame, each node has exactly one packet. In the first superslot, one packet is transmitted from any level to the next lower level. Because each node is a parent of exactly one node except for the node at the highest level $|V| - 1$, it also receives one packet during the superslot. Thus, at the end of the first superslot, each node at level less than $|V| - 1$ has exactly one packet to transmit, the node at level $|V| - 1$ has no packet, and each node has transmitted exactly one packet during the superslot. This means that at the end of the first superslot, each packet has moved by one hop and one packet has reached the final destination AP .

In the same way, at the beginning of the second superslot, each node at level less than $|V| - 1$ has one packet to transmit, and at the end of the second superslot, each packet has moved by one more hop, there are no more packets at levels greater than or equal to $|V| - 2$ and the node AP has received exactly one packet. Continuing in this manner, at the end of $(|V| - 1)$ superslots, all packets will have reached the final destination AP .

The maximum number of time slots in each frame is at most the product of the maximum number of slots in each superslot and the maximum number of superslots necessary for all packets to reach the destination AP , namely $3(|V| - 1)$.

Case 2. Because the interference graph of the tree network satisfies the ancestor property, the corresponding linear tree interference graph CL satisfies $IL = \emptyset$. It can therefore be colored optimally with 3 colors.

First assume that we select exactly one node to transmit from each level (of the original tree graph $G = (V, E)$) corresponding to the color of the slot. At the beginning of the frame, each node has one packet. In the first superslot, one packet is transmitted from each level to the next lower level. Except at the highest level, each level receives one packet. Therefore, one packet has moved one hop closer to the AP at each level, one packet from level one has reached AP , and nodes at the level of the *depth* of the tree may have no more packets.

At the end of the second superslot, the number of packets transmitted from one level to one lower level is again one except, possibly, for level *depth*. Each level less than *depth* - 1 has one packet to transmit, while nodes at levels *depth* or *depth* - 1 may have exhausted all packets. Continuing in this manner, by the end of i -th superslot, there are no more packets above some *threshold* level, and there is at least one packet at levels lower than this threshold. Since each level below the threshold is guaranteed to have a packet, and all levels with at least one packet can transmit once in each superslot, one packet reaches AP in each superslot. Therefore, the number of superslots required for all packets to reach AP

is $|V| - 1$. Since there are three slots in each superslot, the maximum frame length is again $3(|V| - 1)$.

The scheduling algorithm allows a subset of non-conflicting nodes (instead of a single node) at each level to transmit so the resulting frame length will also be at most $3(|V| - 1)$.

Case 3. The worst case is when there is an interfering edge between a node at level j and every node at level i with $|i - j| \leq K$. The corresponding linear graph can be colored by $K + 2$ colors in that case. Assign color 1 to v_1 . The color of the nodes $\{v_2, \dots, v_{K+2}\}$ cannot be 1. Assign the smallest color, 2, to node v_2 . The color of $\{v_3, \dots, v_{K+3}\}$ cannot be 2. Assign the smallest color, 3, to v_3 . Continuing in this way, v_{K+2} is assigned color $K + 2$. Node v_{K+3} is assigned color 1, since its color is restricted not to be $2, \dots, K + 2$. Thus, the algorithm colors this network with $K + 2$ colors in a round robin fashion with color 1 assigned to v_1 . The interference graph of any other network is a subgraph of this worst case. \square

The same reasoning as in Case 2 now indicates that at least one packet reaches AP in each superslot so the number of superslots needed is at most $|V| - 1$. Hence the frame length is at most $(K + 2)(|V| - 1)$ time slots.

Case 4. The number of superslots required for all packets to reach AP is the number of packets in the network, which is $|V| - 1$. The maximum number of slots in each superslot is the number of colors, α . The upper bound on the frame length is then $\alpha(|V| - 1)$. \square

Theorem 3 *Assume that node $i \in V$ has generated g_i packets to transmit. For level based scheduling algorithm, in cases 1 and 2 the maximum length of the frame is $3\sum_{i \in V} g_i$ time slots; in case 3 it is $(K + 2)\sum_{i \in V} g_i$; and in case 4 it is $\alpha\sum_{i \in V} g_i$, in which α is the number of colors used in the linear network corresponding to G and C .*

Proof The proof is similar to that of Theorem 2. The number of superslots required for all packets to reach AP is the number of packets in the network, which is $\sum_{i \in V} g_i$. The maximum number of slots in each superslot is the number of colors. \square

Theorem 4 *Assume that node $i \in V$ has generated g_i packets to transmit. For node based scheduling algorithm, the maximum length of the frame is $\alpha\sum_{i \in V} g_i$, in which α is the number of colors used in the conflict graph GC .*

Proof The proof is similar to that of Theorem 2. In node based scheduling, during each superslot, each node is given at least one chance to transmit.

Let us assume that another algorithm, namely node-level based

scheduling, schedules only one node containing at least one packet from each level of the routing tree $G = (V, E)$ rooted at the AP and does not schedule any node if that level does not contain any packet. By the same reasoning as in the proof of Theorem 2, the number of superslots required for all packets to reach AP is the number of packets in the network, which is $\sum_{i \in V} g_i$. $\sum_{i \in V} g_i$ is also the maximum number of superslots required for all packets to reach AP in node based scheduling. The maximum number of slots in each superslot is the number of colors. The result follows. \square

Remark The chromatic number of a graph GC is less than or equal to $1 + deg_{max}$ [17], where deg_{max} is the maximum degree of the nodes in GC . The maximum length of the schedule in node based and level based scheduling algorithms is therefore $(1 + deg_{max})\sum_{i \in V} g_i$, in which deg_{max} is the maximum degree of the nodes in GC and GCL respectively. Since the minimum schedule length is $\sum_{i \in V} g_i$ as shown in Lemma 1, the worst case ratio of the length of the frame to the optimal length is $1 + deg_{max}$.

Lemma 2 *Assume that node $i \in V$ has generated g_i packets to transmit. For distributed scheduling algorithm, the maximum length of the frame is $\alpha\sum_{i \in V} g_i$, in which α is the number of colors used in the conflict graph GC .*

Proof The proof is the same as that of Theorem 4. \square

8 Simulation

The goal of the simulations is to compare the delay performance of the centralized node based and level based scheduling algorithms, and the distributed algorithm.

In the simulations, 1000 nodes are randomly distributed in a circular area of radius 100 units. The density of the nodes is λ_1 inside the radius $\frac{100}{\sqrt{2}}$ and λ_2 between the radius $\frac{100}{\sqrt{2}}$ and 100 units. The transmission range, denoted r_s , is chosen to be slightly larger than the threshold necessary for network connectivity [1].

The results discussed below are averages of the performance of ten different random configurations. Shortest path routing is used to construct the routing tree rooted at the AP, which corresponds to $G = (V, E)$ in Section 2. The interferers of the nodes that form $C = (V, I)$ are the nodes that are inside a larger range r_m , $r_m \geq r_s$, of each node other than its parent and children in the routing tree G . In the coloring part of node and level based scheduling algorithms, the nodes are ordered

in non-increasing order of degree since high degree vertices have more color constraints and so are more likely to require an additional color if inserted late.

Figure 9 shows the delay of node based and level based algorithms as a function of $\frac{\lambda_1}{\lambda_2}$ ratio, for the case $\frac{r_m}{r_s} = 2$. As expected from the examples in Figures 6 and 7, the level based algorithm performs better for low $\frac{\lambda_1}{\lambda_2}$ ratios whereas node based schedule performs better for high $\frac{\lambda_1}{\lambda_2}$ ratios. Figure 10 shows that this is true up to a certain value of $\frac{r_m}{r_s}$ ratio. Node based scheduling algorithm performs better at high $\frac{r_m}{r_s}$ ratios.

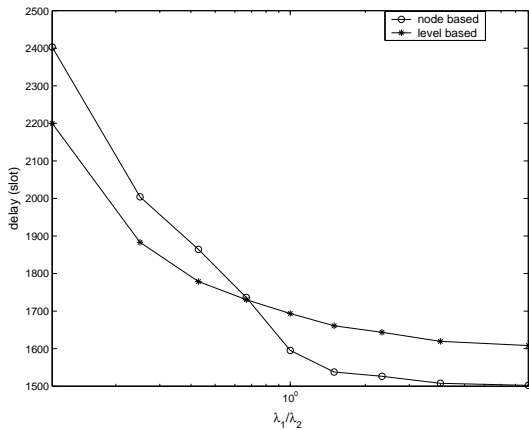


Figure 9: Comparison of the delay of node based and level based scheduling algorithms for different $\frac{\lambda_1}{\lambda_2}$ ratios and $\frac{r_m}{r_s} = 2$.

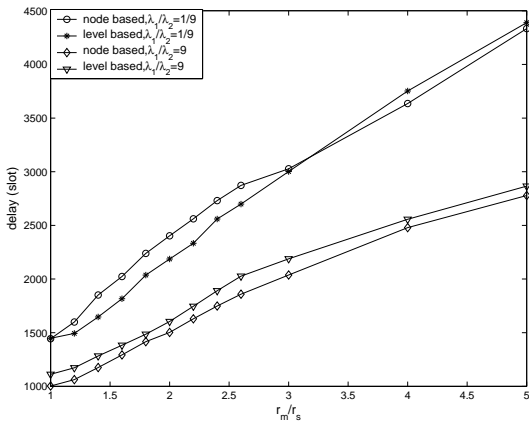


Figure 10: Comparison of the delay of node based and level based scheduling algorithms for different $\frac{r_m}{r_s}$ ratios.

Figure 11 shows the performance of distributed algorithm in terms of the ratio of its delay to that of the centralized node based scheduling algorithm. The delay ratio is in the 10 –

70 range whereas the ratio of the number of colors used in the distributed algorithm to that of centralized algorithm is in the 1 – 1.3 range. This suggests the basic disadvantage of distributed algorithms to be the scheduling of the nodes that do not have any packet in a specific slot and the elimination of other nodes as a result. The delay ratio therefore increases as $\frac{r_m}{r_s}$ ratio increases due to the increase in the number of colors used in the original network.

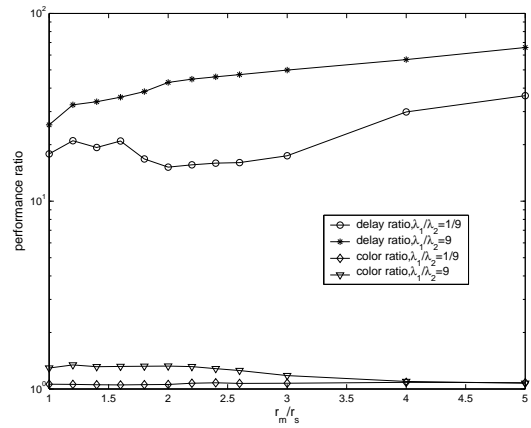


Figure 11: Performance ratio of the distributed algorithm to the centralized node based scheduling algorithm in delay and number of colors used in GC .

9 Conclusion

The common scheduling problem in multi-hop networks employing a TDMA MAC protocol is to determine the smallest length conflict-free assignment of slots where each link or node is activated at least once. This is based on the assumption that there are many independent point-to-point flows in the network. In sensor networks where data are often transferred from the sensor nodes to a few central data collectors, the problem is to determine the smallest length conflict-free assignment of slots during which the packets generated at each node reach their destination. This optimization problem is shown to be NP-complete.

We propose two centralized heuristic algorithms for solving the problem: node based scheduling and level based scheduling. In node based scheduling, the schedule is obtained based on the coloring of the original network. The nodes of the color corresponding to each slot with at least one packet are chosen first and additional nodes are added afterwards. In level based scheduling on the other hand the original network is first transformed to a linear network where each node corre-

sponds to a level in the original network. The schedule of the original network is then obtained based on the coloring of the linear network. This scheduling algorithm schedules a non-conflicting set of nodes corresponding to each level of the color for the current slot and then schedules additional nodes if possible. The movement of the packets across the network is balanced much better in level based scheduling for topologies of higher density of the packets further away from the common sink whereas giving equal chance to the nodes in node based scheduling performs better in topologies of equal density of the packets across the network or higher packet density at low levels.

We also propose a simple token based distributed algorithm to understand the performance of distributed algorithms compared to centralized ones. The distributed algorithm is based on a two-stage coloring algorithm at the end of which nodes assigned the same color form a maximal nonconflicting set. We observe that the delay in distributed algorithm increases by a factor of 10 – 70 over centralized algorithms for 1000 nodes although the number of colors used in coloring the network is almost the same. This suggests the basic disadvantage of distributed algorithms to be the scheduling of the nodes that do not have any packet and the elimination of the scheduling of other nodes as a result. This is hard to avoid in a distributed fashion since the global topology information is required to know whether the interfering nodes have any packets. Distributed scheduling algorithms that improve upon this token based algorithm in the context of sensor networks is an interesting research direction.

References

- [1] S. B. Wicker B. Krishnamachari and B. Bejar. Phase transition phenomena in wireless ad-hoc networks. In *Symposium on Ad-Hoc Wireless Networks, IEEE GLOBECOM 2001*, November 2001.
- [2] G. Chakraborty. Genetic algorithm to solve optimum tdma transmission schedule in broadcast packet radio networks. *IEEE Transactions on Communications*, 52(5):765–777, 2004.
- [3] B. Prabhakar E. Uysal-Biyikoglu and A. El Gamal. Energy-efficient packet transmission over a wireless link. *IEEE/ACM Transactions on Networking*, 10(12):487–499, 2002.
- [4] A. Ephremides and T. V. Truong. Scheduling broadcasts in multihop radio networks. *IEEE Transactions on Communications*, 38(4):456–460, 1990.
- [5] S. C. Ergen and P. Varaiya. On multi-hop routing for energy efficiency. *IEEE Communications Letters*, to appear in 2005.
- [6] S. C. Ergen and P. Varaiya. Pedamacs: Power efficient and delay aware medium access protocol for sensor networks. *IEEE Transactions on Mobile Computing*, to appear in 2005.
- [7] H. Fattah and C. Leung. Energy-efficient packet transmission over a wireless link. *IEEE Wireless Communications*, 9(5):76–83, 2002.
- [8] C. Sharp J. Polastre, R. Szewczyk and D. Culler. The mote revolution: Low power wireless sensor network devices. In *Hot Chips 16: A Symposium on High Performance Chips*, August 2004.
- [9] C. Y. Ngo and V. O. K. Li. Centralized broadcast scheduling in packet radio networks via genetic-fix algorithms. *IEEE Transactions on Communications*, 51(9):1439–1441, 2003.
- [10] LAN-MAN Standards Committee of the IEEE Computer Society. *Wireless LAN medium access control(MAC) and physical layer(PHY) specification*. IEEE, New York, NY, 1997.
- [11] T. L. Magnanti R. K. Ahuja and J. B. Orlin. *Network Flows*. Prentice Hall, Inc, New York, NY, 1993.
- [12] S. Ramanathan and E. L. Lloyd. Scheduling algorithms for multihop radio networks. *IEEE/ACM Transactions on Networking*, 1(2):166–177, 1993.
- [13] R. Ramaswami and K. K. Parhi. Distributed scheduling of broadcasts in a radio network. In *INFOCOM 1989*, pages 497–504, April 1989.
- [14] P. Kumar S. Bansal and K. Singh. An improved duplication strategy for scheduling precedence constrained graphs in multiprocessor systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(6):533–544, 2003.
- [15] M. Dawande S. Gandham and R. Prakash. Link scheduling in sensor networks: Distributed edge coloring revisited. In *IEEE INFOCOM 2005*, March 2005.
- [16] R. S. Sreenivas S. Narayanaswamy, V. Kawadia and P. R. Kumar. Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the compow protocol. In *Proceedings of European Wireless 2002*, February 2002.

- [17] D.P. Jacobs S. T. Hedetniemi and P. K. Srimani. Fault tolerant distributed coloring algorithms that stabilize in linear time. In *International Parallel and Distributed Processing Symposium (IPDPS) 2002*, April 2002.
- [18] B. Dondar S. Ganesh C.W. Tan S. Y. Cheung, S. Coleri and P. Varaiya. Traffic measurement and vehicle classification with a single magnetic sensor. *84th Annual Meeting, Transportation Research Board, January 2005*, chosen to be published in *Journal of Transportation Research Board*, to appear in 2005.
- [19] B. Tavli and W. B. Heinzelman. Mh-trace: Multihop time reservation using adaptive control for energy efficiency. *IEEE Journal on Selected Areas in Communications*, 22(5):942–953, 2004.
- [20] Z. Wu and D. Raychaudhuri. D-Isma: Distributed link scheduling multiple access protocol for qos in ad-hoc networks. In *IEEE GLOBECOM 2004*, pages 1670–1675, November 2004.