Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 837074, 14 pages
http://dx.doi.org/10.1155/2015/837074

Research Article

Optimal Multi-TDMA Scheduling in Ring Topology Networks

Gergely Vakulya,[1] Zsolt Tuza,[1,2] and Gyula Simon[1]

[1] *University of Pannonia, Veszprém 8200, Hungary*
[2] *MTA Rényi Institute, Budapest 1053, Hungary*

Correspondence should be addressed to Gergely Vakulya; vakulya@dcs.uni-pannon.hu

Received 16 September 2014; Revised 12 December 2014; Accepted 18 December 2014

Academic Editor: Trung Nguyen-Thoi

A scheduling algorithm will be proposed for wireless ring topology networks, utilizing time division multiple access (TDMA) with possible simultaneous operation of nodes. The proposed algorithm finds the optimal schedule to minimize the turnaround time for messages in the network. The properties of the algorithm are mathematically analyzed and proven, and practical test results are also provided.

1. Introduction

Sensor networks consist of sensor nodes which can measure various parameters of their environment, make computations, and communicate with each other. Sensor nodes are resource constrained devices, but despite the simplicity of a single device the whole network can solve various tasks efficiently, where the use of other kind of systems is inefficient or impossible [1].

Data dissemination is a common service in wireless sensor networks. Communication patterns may differ greatly in networks depending on the application. The most common tasks are as follows:

(i) convergecast, where information from network nodes is directed to a sink node; this is a typical task in data collection applications;

(ii) broadcast, where information from a source node is transmitted to all network nodes, used mainly to send commands into the network;

(iii) peer-to-peer, where any node can send information to any other node; used in distributed computing, where local communication is required.

A special kind of broadcast is when any node can serve as source; thus the communication pattern is any-to-all. This pattern was used for example in decentralized security systems, when all nodes were able to make decisions, based upon all the available information in the network. A possible solution is the ring topology, where information flows along a circle in the network [2]. This paper focuses on this communication pattern.

Communication networks use either contention based or contention free channel access mechanisms. The advantage of contention based protocols is their dynamism and simplicity. Nodes can compete for channel access when communication is required; thus channel bandwidth is allocated on demand. However, when channel demand is high, frequent collisions may severely decrease network efficiency [3]. Due to the inherently asynchronous nature of contention based communication, in many implementations nodes are continuously switched on, which has a negative impact on power efficiency. To decrease the effect of idle listening, many sophisticated MAC protocols have been developed, for example, RAW [4], S-MAC [5], and μ-MAC [6].

Contention free or schedule based protocols use preallocation schemes (e.g., time division multiple access, TDMA); thus the communication can be performed in the preallocated slots, avoiding collisions. However, systems with fixed assignments may be inefficient, when the preassigned slots are poorly utilized. This problem can be handled by on-demand assignment at the price of network overhead [7].

Efficient TDMA schedules provide optimal performance, where optimality may be defined by various design objectives, depending on the application requirements. Design objectives can be short schedule length, minimal latency,

or minimal power consumption, just to name few (see [8] for a comprehensive list). To achieve the desired design objectives, time slices may be used by multiple nodes to transmit messages, if these transmissions do not interfere with each other. Such reuse of time slots will be referred to as multi-TDMA schedule, which is an NP-complete problem in general [9]. Various heuristics were proposed to calculate multi-TDMA schedules, most of them providing solutions, which are suboptimal, but still acceptable in practice. A review on TDMA protocols in wireless sensor networks can be found in [9]. The most widely studied topology is tree topology [8], but other topologies (e.g., line and star) were also examined [10]. TDMA scheduling in ring topology, to our best knowledge, was not specifically studied.

TDMA scheduling in ring networks, when only one node can transmit in a time slot, is a trivial task in itself. In this case consecutive nodes in the network are assigned to consecutive time slots in the schedule. We will refer to this simple case as linear scheduling. The problem, however, becomes much more complicated, when network characteristics allow multi-TDMA schedule. In this paper, we provide algorithms that are able to efficiently solve the multi-TDMA scheduling problem in ring topology, with minimal worst-case message turnaround time, as design objective.

Note that message turnaround time, in our approach, includes not only the time the message spends in the network (usually used as design objective) but also the delay from the appearance of the information (e.g., measurement by a sensor) to the injection of the message into the network. We call this approach Minimum Worst Case Turnaround Time *(MWCTT)* problem. Since both delay components in worst case depend on the parameters of TDMA schedule, the schedule may be optimized to include both components, providing more realistic objective for time-critical systems.

The novelties of the paper are the following.

(i) The Minimum Worst Case Turnaround Time problem is defined.

(ii) A novel scheduling algorithm will be proposed which supports multi-TDMA scheduling in ring topology networks. The proposed algorithm guarantees optimal scheduling for the MWCTT problem and still has polynomial computational complexity. The correctness, complexity, and performance of the proposed algorithm will be mathematically analyzed and proven.

(iii) An accelerated version of the optimal scheduling algorithm will also be introduced. The accelerated version also provides optimal solution but requires much shorter computational time in most practical cases.

(iv) The performances of the proposed algorithms are compared to a reference naive breadth first search algorithm.

(v) The properties of the generated optimal schedules are also analyzed in terms of speed and power consumption.

(vi) The performance of the generated schedules are compared, using the proposed algorithm and a reference method.

In Section 2 related work is reviewed. In Section 3 a mathematical formalization will be given, including the definition of the MWCTT problem, followed by the definition of the new scheduling algorithms for multi-TDMA rings, and finally the properties of the algorithms (optimality and complexity) will be proven. In Section 4 further analytical properties (delivery time and power consumption) will be discussed, along with a performance comparison. Section 5 provides test results, and the performance of the proposed algorithms is analyzed using real implementation. Section 6 concludes the paper.

2. Related Work

Most TDMA scheduling algorithms use the high level protocol model, where a connectivity graph represents the network, the vertices, and edges corresponding to nodes and communication links, respectively. In this model the goal is to create a schedule where two constraints are satisfied: (C_1) a node cannot receive and transmit at the same time slot, and (C_2) two transmissions are not allowed to interfere with each other; that is, if more transmitters are scheduled in the same time slot, no link can be present from a transmitter to the receiver of another transmitter [11]. Some algorithms use the physical model, where the communication links and potential collisions are more accurately modelled by the signal-to-interference-plus-noise-ratio, resulting in potentially more accurate, but usually more complicated, models. In general, protocol-based models provide more pessimistic results, where as much as 30 percent of the network capacity may be lost, compared to accurate physical model-based scheduling [12].

Depending on the application requirements, TDMA design objectives may be very different. In applications, where message latency is important, TDMA schedulers use latency-related metrics, for example, schedule length, or average message latency [9, 10, 13]. In networks, where large amount of data must be delivered, throughput capacity [14] and fairness [15] may be maximized. In many applications energy utilization and thus network lifetime is a key design factor; in these cases energy efficiency is the main design objective [16, 17]. Some solutions address multiobjective scheduling [8, 13, 18].

The most widely studied network topology is tree topology (e.g., [8, 19]), but also line [10] and arbitrary topologies [11] were investigated. Scheduling algorithms also consider various data models. Most networks are designed to primarily perform data collection (convergecast) or data dissemination (broadcast) tasks [10], but other objectives, for example, small round trip delays, were addressed as well [20]. Some algorithms take into consideration varying data intensity as well [10, 21].

The TDMA scheduling problem is known to be NP-complete, since it is equivalent to the graph coloring problem [8, 9]. Since the problem is hard to solve in general, several

approximate solutions were proposed to provide computable schedules, even if they are suboptimal. The scheduling problem in most cases may be transformed to coloring of the conflict graph, the nodes of which are edges of the connectivity graph, and its edges represent conflicting links, according to constraints C_1 and C_2 [11]. Since the coloring problem is NP-complete in general, various heuristics were proposed. For tree topologies, in [9] two heuristics were proposed, the effectiveness of which was shown to depend on packet generation density in the network. In [22] the conflict graph was expanded to represent the size of the subtree, connected to a parent node, and the coloring was done on the expanded conflict graph, using an approximate coloring algorithm.

In [19] the Minimum Information Gathering Time Problem was solved, providing shortest time schedule for data collection applications, where each node sends one message in every period towards a sink node. The optimal solution was provided for line and tree topologies, for general networks heuristics were proposed. A similar problem was solved in [21], using Integer Linear Programming formulation.

A unified approach was proposed in [11] for a wide class of channel assignment problems, including TDMA. In this framework the search for the solution is aided by a heuristic labelling step, which is followed by the graph coloring step, where the order of the coloring is determined by the labels. The scheduler is simple and fast but the optimum is not guaranteed.

Ring topology networks are special type of a broadcast networks, where every node can send information to every other node (all to all) [23]. A ring is selected from the communication graph (Hamiltonian cycle), along which is a linear TDMA schedule is generated.

Creating a ring in an ad hoc wireless network is not a trivial problem; it is equivalent to finding a Hamiltonian cycle in the connectivity graph of the network, which is not only NP-hard in general, but not constant approximable even in special cases like cubic Hamiltonian graphs [24]. Based on Pósa's seminal rotational algorithm [25], several heuristics have been proposed to enhance the efficiency of the search method, for example, by applying precheck mechanisms to increase the probability of the right choices during the search [26], or by avoiding backtracking [27]. Fiedler vectors can be used to find pseudo-Hamiltonian cycles [28]. A semiautomatic heuristic approach, especially the one designed for connectivity graphs of sensor networks, was proposed in [29].

TDMA in ring topology was addressed in [30], where the clocks of the nodes were automatically and evenly distributed to create a stable TDMA ring network. Extended ring topologies were used in [23] to represent network hierarchy. A heuristic TDMA scheduling algorithm for such extended ring topologies was proposed in [31]. The heuristic approaches provide in most cases practically acceptable results, but they cannot guarantee optimal scheduling.

In this paper multi-TDMA scheduling in ring topology networks is addressed, where multiple nodes are allowed to transmit in the same time slot, if network topology allows collision-free communication. The goal is to find optimal schedule with shortest turnaround time, as design objective. We will prove that optimal multi-TDMA scheduling is possible in polynomial time, and two scheduling algorithms will be proposed and analyzed.

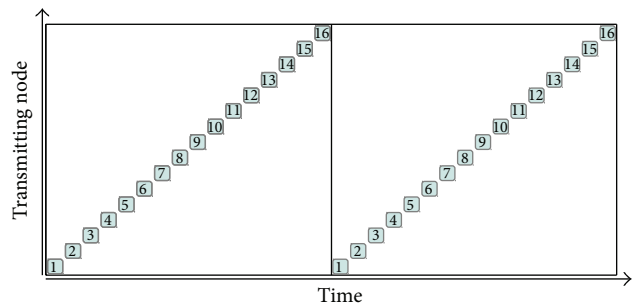

Figure 1: Timing diagram of a linear TDMA schedule for 16 nodes. Each node uses exclusively one time slice for transmission. Two periods are shown, each requiring 16 time slices.

3. Linear and Multi-TDMA Scheduling

3.1. Basic Principles. The TDMA approach uses dedicated time slices for the transmitter nodes to avoid collisions. The linear round-robin TDMA operates with a predefined periodic schedule. The periods are divided into time slices and each time slice is assigned to one transmitter-receiver pair. The nodes are arranged in a cycle. In the first time slice the first transmitter node transmits a packet to its neighbor, and in every later time slice the node that has just received the message in the previous time slice transmits a packet to its neighbor. In the last time slice the last node will transmit to the first one and this period is repeated, as shown in the example with 16 nodes in Figure 1. The figure indicates time instants when nodes are transmitting. In linear round-robin TDMA the number of time slices in a period is equal to the number of nodes.

Multi-TDMA allows more nodes to transmit at the same time, provided the transmitters do not disturb the messages of each other. Figures 2 and 3 show two possible multi-TDMA schedules for 16 nodes. In Figure 2 the length of the period is 6 and the period must be repeated 3 times in order that a message reaches all nodes in the network and arrives back to node 1. In Figure 3 the period length is 7 and 4 periods are required to deliver a message to all of the nodes and return it back to node 1. In both examples maximum 3 nodes transmit at the same time. (Naturally nodes scheduled for the same time slices must not interfere with each other.)

Utilization of multi-TDMA networks can be advantageous if short message delay is required. Suppose that a node detects an event and sends a message about it to the network. For the linear TDMA schedule in Figure 1, the worst case turnaround time from the detection until all nodes receive the message and it gets back to the source node is 32 time slots; the source node must wait at most 16 time slots from the detection until its scheduled transmission time slot and then 16 time slots are required to propagate the message to

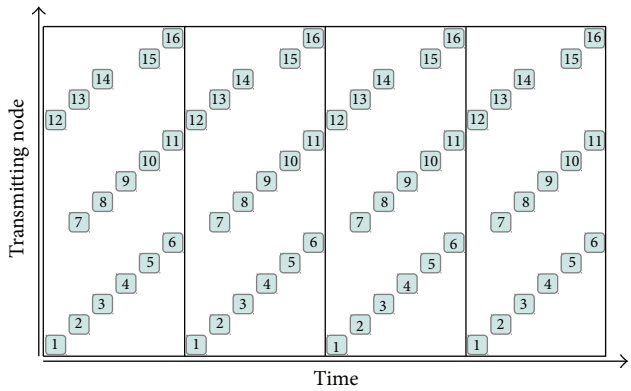

Figure 2: Timing diagram of a multi-TDMA schedule for 16 nodes. Independent transmissions are scheduled for the same time slice. A period requires $c = 6$ time slices. In this example $k = 3$ periods are required to deliver a message from any node to all other nodes.

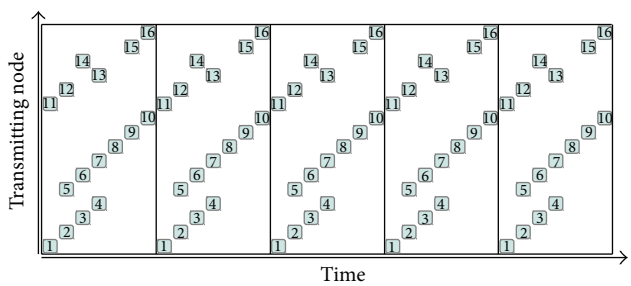

Figure 3: Timing diagram of a multi-TDMA schedule for 16 nodes. Independent transmissions are scheduled for the same time slice. A period requires $c = 7$ time slices. In this example $k = 4$ periods are necessary for a message to turn around.

all other nodes and get it back. In Figure 2, where multi-TDMA schedule is used, the worst case turnaround time is only 24 time slots (6 time slices to start the transmission and $3 \cdot 6$ to propagate the message). Note that inappropriate multi-TDMA schedules may give larger propagation delay than the linear schedule, as in the example of Figure 3, where the worst case delay is 35 $(7 + 4 \cdot 7)$ time slots.

3.2. Design Objective. The multi-TDMA scheduling to be proposed implements a data dissemination service, where the information, originating at any node, is disseminated to every other node in the network, along the path defined by the ring topology.

The design objective is to minimize the worst case turnaround time, introducing the Minimum Worst Case Turnaround Time (MWCTT) problem. The MWCTT is measured from the time point when the information (e.g., measurement or alert) to be disseminated appears asynchronously in the initiator node, to the time point when messages in the ring topology network reach the initiator node again. This scenario models the real-life situation when a node sends an alert to the network and waits for the reactions of other nodes.

As shown in Section 3.1, MWCTT consists of two components. First the initiator node has to wait for its transmission time slice, which requires c time slices in worst case, where c is the lenght of a period, for example, in Figure 2, $c = 6$, while in Figure 3, $c = 7$. Note that the worst case scenario is when the information to be disseminated appears at the initiator node just after it has begun its transmission. After the initiator node transmitted the message into the network, the message must reach all the other nodes and arrive back to the initiator node, which requires $k \cdot c$ time slices, where k is the necessary number of the repetitions of the periods (e.g., in Figure 2, $k = 3$, while in Figure 3, $k = 4$). Thus the worst case dissemination time is equal to WCTT $= c + c \cdot k = c \cdot (k + 1)$ time slices. The goal of the scheduling is to minimize WCTT.

3.3. Mathematical Formalization. The network is represented by a connectivity graph $C = (N, L)$, where N denotes the set of vertices corresponding to the sensor nodes and L denotes the set of edges representing the links between nodes. The cardinality of N is denoted by n.

The $lq : L \rightarrow (0, 1]$ *link quality function* shows the quality of the links according to a predefined metric. Only the links with link quality greater than a threshold z should be used for communication. L_C denotes the set of possible communication links:

$$L_C = \{p \in L \mid lq(p) > z\}, \tag{1}$$

and the C_C *communication graph* is defined as follows:

$$C_C = (N, L_C). \tag{2}$$

Note that the weaker links also have to be considered in multi-TDMA scheduling, because even weaker signal strength levels may cause collisions when a receiver hears multiple transmitters at the same time.

Figure 4 shows a 16-node example, where continuous lines denote good links (with $lq > z$), while dashed lines denote weaker links (with $0 < lq \leq z$).

In the discussed communication model each node transmits to the next node; therefore, a Hamiltonian cycle must exist in C_C. Without loss of generality it can be assumed that $N = \{N_1, \ldots, N_n\}$ and the edges of the Hamiltonian cycle are $(N_1, N_2), (N_2, N_3), \ldots, (N_{n-1}, N_n), (N_n, N_1)$, as shown in Figure 4 for $n = 16$.

3.4. Definitions

Definition 1. Each node N_i transmits to its *receiver node*, denoted by $r(N_i)$:

$$r(N_i) = N_{(i \bmod n)+1}. \tag{3}$$

Definition 2. The $G_I = (N, I)$ *transmission-interference graph* (or constraint graph) represents transmitter interferences. There is an edge between two nodes if they should not transmit in the same time slice, because of C_1 and C_2; that is,

$$\begin{aligned}(p, q) \in I \text{ if } (p, r(q)) \in L \text{ or } (q, r(p)) \in L \text{ or} \\ q = r(p) \text{ or } p = r(q).\end{aligned} \tag{4}$$

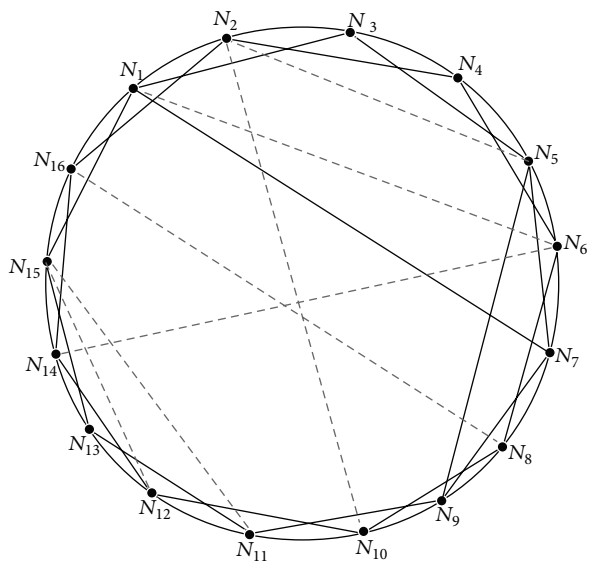

Figure 4: A possible connectivity graph (continuous and dashed lines) and communication graph (continuous lines only).

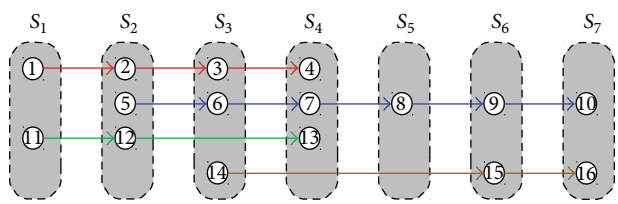

Figure 5: The schedule connection graph for the example shown in Figure 3. The schedule itself is represented by $S_1, S_2, \ldots, S_7$. Its width is 4, as shown by the 4 monochromatic paths.

Definition 3. A set of nodes in G_I is *polite* if all nodes in the set can transmit in the same time slice without collision. POL is the set of the polite transmitter-sets; that is,

$$\mathrm{POL} = \{T \subseteq N \forall p, q \in T,\ \ p \neq q : (p, q) \notin I\}. \quad (5)$$

Definition 4. A *schedule* is a sequence of node-sets $S = (S_1, \ldots, S_l)$, where $S_i \subseteq N$ and l is the length of the schedule, such that S_i contains the transmitting nodes for the ith time slice of the schedule. Each node transmits exactly once in the sequence to its receiver node; that is,

$$1 \leq i < j \leq l : S_i \cap S_j = \emptyset, \quad \forall i, j,$$
$$\bigcup_{1 \leq i \leq l} S_i = N. \quad (6)$$

Figure 5 shows the schedule shown in Figure 3, where $S_1 = \{N_1, N_{11}\}$, $S_2 = \{N_2, N_5, N_{12}\}$, $S_3 = \{N_3, N_6, N_{14}\}$, $S_4 = \{N_4, N_7, N_{13}\}$, $S_5 = \{N_8\}$, $S_6 = \{N_9, N_{15}\}$, and $S_7 = \{N_{10}, N_{16}\}$.

Note that the schedule describes the operation of the network for exactly one period; such periods are repeated periodically and in each period the nodes are operated according to the schedule.

Definition 5. To a given schedule S, a *schedule connection graph* $J(S) = (N, Y)$ is defined, where

$$(p, q) \in Y \quad \text{if } \exists i, j : i < j,\ \ p \in S_i,\ \ q \in S_j,\ \ q = r(p). \quad (7)$$

The meaning of edge (p, q) is that node q can retransmit the message received from node p, in a subsequent time slice *in the same period*. Figure 5 shows the schedule connection graph as well, for the schedule shown in Figure 3. There is an edge between node 2 and node 3, because the message received from node 2 in time slice 2 can be retransmitted by node 3 to node 4 in time slice 3, in the same period. Similarly, node 4 can transmit to node 5 in the same period; therefore, there is an edge between node 3 and node 4. But node 5 cannot retransmit the message, received from node 4, in the same period: node 5 can transmit only in time slice 2 of the next period. Therefore, there is no edge between node 4 and node 5.

Definition 6. The *width of schedule* S is the number of independent components in the schedule connection graph $J(S)$.

In Figure 5 the width of the schedule is 4. Note that the width has an important practical meaning. If the width of the schedule is k then the schedule must be repeated k times in order to deliver a packet from any node to all other nodes.

Definition 7. Assuming that the Hamiltonian cycle has been partitioned into connected parts (i.e., subpaths), a *segment* is the node-set of any of those parts.

Note that the independent paths in the schedule connection graph are (independent) segments. In Figure 5 red, blue, green, and brown paths correspond to independent segments. Thus the following definition naturally follows.

Definition 8. For any k-wide schedule a *segmentation* $R = (R_1, \ldots, R_k)$ can be defined on the Hamiltonian cycle, where $R_i = (R_{i,1}, \ldots, R_{i,l_i})$ is a segment such that $(R_{1,1}, \ldots, R_{1,l_1}, \ldots, R_{k,1}, \ldots, R_{k,l_k}) = (N_1, \ldots N_n)$, where the length of the ith segment is l_i and the vertices of R_i are denoted with $R_{i,1}, \ldots, R_{i,l_i}$.

Note that a k-wide schedule contains k independent segments, such that the union of the segments is N.

In Figure 6 the segmentation for the schedule of Figure 5 is shown.

Definition 9. Given a segmentation R, the *state* $P = \langle R_{1,i_1}, R_{2,i_2}, \ldots, R_{k,i_k} \rangle$ is a list of nodes where node R_{j,i_j}, $0 \leq i_j \leq l_j$ is the node which transmitted for the last time in segment j. If in segment j there was no transmission yet then notation $R_{j,0} = \otimes$ is used. The state representing that no transmission has occurred yet is denoted by $\langle \otimes, \otimes, \ldots, \otimes \rangle = (\otimes)_k$. The ith element of state P, corresponding to segment R_i, is denoted by $P[i]$.

For the construction of a k-wide schedule the directed graph $A_R = (\mathcal{G}, D)$ is used. The vertices in A_R denote the

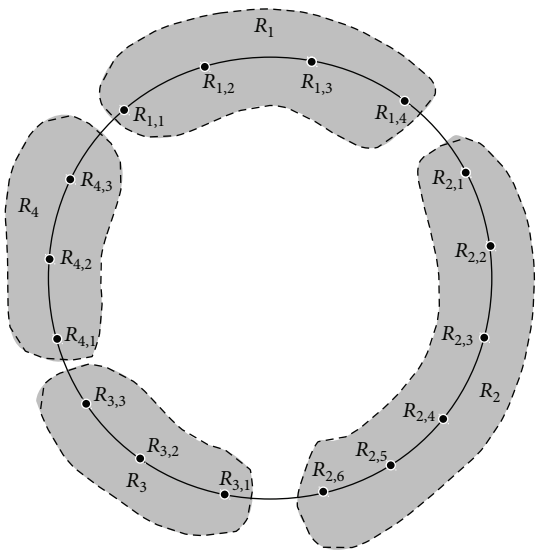

Figure 6: Segmentation of the Hamiltonian cycle.

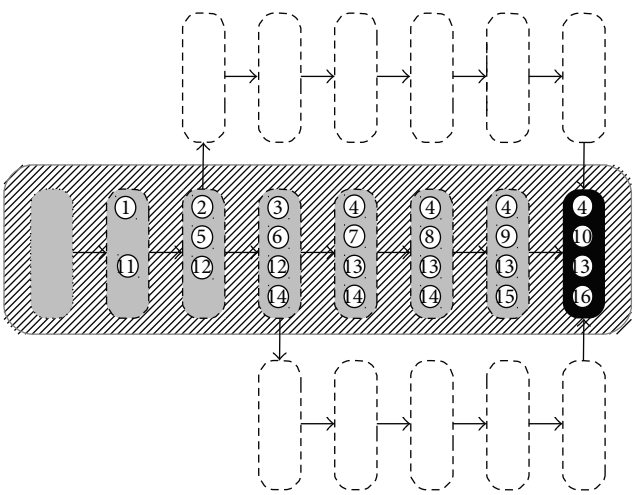

Figure 7: A subgraph of the state graph. The highlighted path leads to an optimal schedule.

states of the schedule. The edges of the graph define possible sequences of states representing valid schedules.

Notice that state transition $(P, Q) \in D$ means that the set of actual transmitters is

$$\bigcup_{i=1}^{k} (\{Q[i]\} \setminus \{P[i]\}). \tag{8}$$

The highlighted part of Figure 7 shows the states and state transitions corresponding to the schedule in Figure 5. The state graph starts from state $(\otimes)_4 = \langle \otimes, \otimes, \otimes, \otimes \rangle$. After the time slice corresponding to S_1, nodes 1 and 11 are the nodes which transmitted for the last time from their segments (from the other two segments no transmission has happened yet); thus the corresponding state is $\langle N_1, \otimes, N_{11}, \otimes \rangle$. After the time slice of S_2 nodes 2, 5, and 12 were the last transmitters in their segments, resulting state $\langle N_2, N_5, N_{12}, \otimes \rangle$. After the time slice of S_3 nodes 3, 6, 12, and 14 are the last transmitters, with the corresponding state of $\langle N_3, N_6, N_{12}, N_{14} \rangle$. Note that no node transmitted from the segment of node 12 (along the green path in Figure 5) in the actual time slice; therefore, in this case node 12 remains in the state.

The formal definition of A_R is as the follows.

Definition 10. For a segmentation $R = (R_1, \ldots, R_k)$, $A_R = (\mathcal{G}, D)$ is called the *state graph* of segmentation R. The nodes in the node set $\mathcal{G}$ of A_R represent the states which may occur in a schedule compatible with R. The edges of the state graph are defined as follows.

$(P, Q) \in D$ if

(TR1) $\forall i : Q[i] \neq \otimes \wedge P[i] = \otimes \Rightarrow Q[i] = R_{i,1}$; that is, in any segment R_i the first node $(R_{i,1})$ will transmit for the first time;

(TR2) $\forall i : P[i] \neq \otimes \Rightarrow Q[i] \neq \otimes$. It expresses the trivial requirement that if once a node has transmitted in a segment, any later time there is one node in that segment, which transmitted for the last time;

(TR3) $\forall i : P[i] \neq \otimes \Rightarrow V \vee W$, where $V = (Q[i] = P[i])$, $W = (P[i] = R_{i,j} \wedge Q[i] = R_{i,j+1} \wedge j < l_i)$; that is, after a node transmits in the segment, its receiver node will transmit either in the next time slice (condition W), or in a later time slice, in which case the currently latest transmitter node remains unchanged (condition V);

(TR4) $(\bigcup_{i=1}^{k} (\{Q[i]\} \setminus \{P[i]\})) \in$ POL; that is, only a polite transmitter-set is allowed to transmit in one time slice.

Rules (TR1)–(TR4) are the *tracing rules*.

3.5. The FSS Algorithm. In this section first the fixed-segmentation-scheduler (FSS) then the optimal-multi-TDMA-scheduler (OMTS) algorithm will be introduced.

The pseudocode of the FSS algorithm is shown in Pseudocode 1. Notes on notations are the following.

(i) *dequeue*(L) removes the first element of list L, while *enqueue*(L, X) adds X to the end of list L, see lines (16), (21), and (27).

(ii) *reverse*(L) returns reverse of list L.

(iii) *parent*(Q) represents the state from which state Q was traced; see lines (22) and (26).

The input of the algorithm consists of the connectivity graph C, the communication graph C_C, and the segmentation $R = (R_1, \ldots, R_k)$. The algorithm builds a graph $\widehat{A}_R = (\widehat{\mathcal{G}}, \widehat{D})$, where $\widehat{\mathcal{G}} \subseteq \mathcal{G}$, $\widehat{D} \subseteq D$, and $\widehat{D}$ is represented by the *parent*() mapping; see lines (17)–(22) of the FSS algorithm.

During the iterations $\mathcal{G}$ contains states which are reached by breadth-first search (traced nodes), and $\mathcal{O}$ contains states, already in $\mathcal{G}$, from which breadth-first search was not continued yet (open nodes).

In the beginning the list $\mathcal{O}$ of open nodes contains only the empty set (line (14)) and in each iteration the algorithm traces all possible combinations of transmitting nodes (line (18))

```
FIXED-SEGMENTATION-SCHEDULER (FSS)
(1) input
(2)   C = (N, L): connectivity graph
(3)   C_C = (N, L_C): communication graph
(4)   ℛ = {R_1, ..., R_k}: set of segments
(5) variables
(6)   𝒢̂: set of traced nodes in A
(7)   𝒪: list of open nodes in A
(8)   P: the currently traced node
(9)   T: a possible combination of transmitting segments
(10)  Q: set of possible nodes with directed arc from node T
(11)  parent: mapping from nodes to nodes
(12) begin
(13)  𝒢̂ = {(⊗)_k}
(14)  𝒪 = ((⊗)_k)
(15)  while E_R ∉ 𝒪
(16)    P ← dequeue(𝒪)
(17)    foreach T in 𝒫(ℛ) \ ∅
(18)      Q = {R_{i,1} : R_i ∩ P = ∅, R_i ∈ T} ∪ {r(R_{i,j}) : R_i ∩ P = {R_{i,j}}, R_i ∈ T}
(19)      if (P, Q) satisfies the tracing rules and Q ∉ 𝒢̂:
(20)          𝒢̂ ← 𝒢̂ ∪ Q
(21)          enqueue(𝒪, Q)
(22)          parent(Q) ← P
(23)  Q = E_R
(24)  S = ()
(25)  while Q ≠ (⊗)_k
(26)    P = parent(Q)
(27)    enqueue(S, ⋃_{i=1}^{k} ({Q[i]} \ {P[i]}))
(28)    Q = P
(29)  return reverse(S)
(30) end
```

Pseudocode 1: The pseudocode of the FSS algorithm.

that satisfy the tracing rules (lines (19)–(21)). The schedule is reconstructed from the series of states in $\widehat{A}_R$ from endstate E_R to start-state $(\otimes)_k$ in a backwards manner (lines (23)–(28)).

Note that the FSS algorithm calculates the optimum schedule for a given segmentation. The OMTS algorithm enumerates all the possible segmentations and calls the FSS algorithm for each segmentation.

The pseudocode of the OMTS algorithm is shown in Pseudocode 2. Lines without asterisks give the basic OMTS algorithm. With all lines, including those with asterisk, an accelerated version is given with early cuts (OMTS-A).

The input of the algorithm consists of the connectivity graph C, communication graph C_C, and maximal width $k_{\max}$. Variable *len* stores the length of the shortest known schedule; it is set to ∞ at the beginning (line (12)). The basic algorithm iterates on widths k from $k_{\max}$ down to 1 (line (13)); every possible k-tuple is generated as a starting node for the segmentations (line (15)). List *startNodes* stores the starting points of the segments (line (16)), while mapping *maxSegLength* stores the length of the longest segment for the corresponding k-tuples (lines (17)-(18)). The algorithm iterates on the list *startNodes* (line (20)) and generates the actual segmentations (line (22)). For each segmentation the FSS algorithm is executed (line (23)) which returns the shortest schedule S_R for the given segmentation.

While the OMTS algorithm evaluates all possible segmentations, the OMTS-A algorithm does not call FSS for those cases which trivially results in longer schedules than the current best schedule; since the length of the longest segment is a lower bound for the length of any schedule, during the iterations those segmentations are considered, for which *maxSegLength* < *maxSchedLen* (see lines (21), (27), and (28)). To increase the efficiency of the early cuts, the segmentations are sorted by *maxSegLength* before the iterations (lines (17)–(19)).

3.6. Algorithm Properties

Lemma 11. *For a given* $(R_1, \ldots, R_k) = ((R_{1,1}, \ldots, R_{1,l_1}), \ldots, (R_{k,1}, \ldots, R_{k,l_k}))$ *segmentation, endstate* $E_R = \langle R_{1,l_1}, R_{2,l_2}, \ldots, R_{k,l_k} \rangle$ *gives the last state of all possible schedules, that is, the last node in all possible paths in the state graph corresponding to the given segmentation.*

Proof. Each node has to transmit exactly once in the schedule; therefore, from each segment each node has to be present in

```
OPTIMAL-MULTI-TDMA-SCHEDULER
(OMTS AND OMTS-A*)
(1) input
(2)    C = (N, L): connectivity graph
(3)    C_C = (N, L_C): communication graph
(4)    k_max: maximum width of the schedule
(5) variables
(6)    minSchLen: the length of the shortest known schedule
(4)    k: the actually considered width
(5)    startNodes: list of starting k-tuples of the traced segmentations
(6)    maxSegLength: length of the longest segment in a segmentation
(7)    B: a starting k-tuple of a segmentation
(8)    R: a segmentation
(9)    S_R: an optimal schedule for R
(10)   S: an optimal schedule
(11) begin
(12)   minSchedLen = ∞
(13)   for k = k_max downto 1
(14)      startNodes = ()
(15)      foreach B in {B : B ⊆ N ∧ |D| = k}
(16)         startNodes = enqueue(startNodes, B)
(17)*        maxSegLength(B) =
(18)*           max((j − i + n) mod n), i ≠ j, N_i, N_j ∈ B
(19)*     sort startNodes by maxSegLengths increasing order
(20)      foreach B in startNodes
(21)*        if maxSegLength(B) · (k + 1) < minSchedLen
(22)           R = (R_1, ..., R_k) : R_{i,1} ∈ B, 1 ≤ i ≤ k
(23)           S_R = FSS(C, C_C, R)
(24)           if length(S_R) · (k + 1) < minSchedLen
(25)              S = S_R
(26)              minSchedLen = length(S_R) · (k + 1)
(27)*        else
(28)*           break (foreach)
(29)   return S
(30) end
```

Pseudocode 2: The pseudocode of the OMTS and the OMTS-A algorithms.

at least one state. If a node from a segment is present in state P, and (P, Q) is an edge of the state graph, then Q will also contain a node from that segment. From each segment nodes appear in increasing order in states belonging to a schedule. Therefore, the last state belonging to a schedule will contain the last nodes from each segment. □

Lemma 12. *The shortest schedule for a given segmentation $R = ((R_{1,1}, \ldots, R_{1,l_1}), \ldots, (R_{k,1}, \ldots, R_{k,l_k}))$ corresponds to states on a shortest path in graph A_R from $(\otimes)_k$ to the endstate E_R.*

Proof. For a given segmentation each transition of the state graph describes a time slice of a possible schedule. Thus the length of the schedule is equal to the length of a path in the state graph from $(\otimes)_k$ to the endstate E_R, and any shortest path from $(\otimes)_k$ to the endstate E_R corresponds to one of the shortest schedules. □

The next lemma guarantees the existence of at least one valid schedule.

Lemma 13. *For every $P \neq E_R$ there exists a state $Q = \langle R_{1,j_1}, R_{2,j_2}, \ldots, R_{k,j_k} \rangle$ where (P, Q) satisfies the tracing rules.*

Proof. Since $P \neq E_R$, there exists i such that $R_{i,j_i} \neq R_{i,l_i}$. Let $Q[i] = R_{i,j_i+1}$ and $Q[k] = R_{k,j_k}$, $k \neq i$. In this case (P, Q) satisfies the tracing rules since only one node (R_{i,j_i+1}) is transmitting. □

Definition 14. For a given state $P = \langle R_{1,i_1}, R_{2,i_2}, \ldots, R_{k,i_k} \rangle$ and endstate $E_R = \langle R_{1,l_1}, R_{2,l_2}, \ldots, R_{k,l_k} \rangle$ the distance between P and E_R is defined as $D(P) = \sum_{k=1}^{k} l_k - i_k$. Note that $D(P) = 0$ if and only if $P = E_R$.

Lemma 15. *For every two states $P = \langle R_{1,i_1}, R_{2,i_2}, \ldots, R_{k,i_k} \rangle$ and $Q = \langle R_{1,j_1}, R_{2,j_2}, \ldots, R_{k,j_k} \rangle$, where (P, Q) satisfies the tracing rules, $\Delta_D < 0$, where $\Delta_D = D(Q) - D(P)$.*

Proof. Consider $\Delta_D = D(Q) - D(P) = \sum_{m=1}^{k}(l_m - j_m) - \sum_{m=1}^{k}(l_m - i_m) = \sum_{m=1}^{k}((l_m - j_m) - (l_m - i_m)) = \sum_{m=1}^{k}(i_m - j_m)$.

According to (TR3) either $j_m - i_m = 0$ (case V) or $j_m - i_m = 1$ (case W), for all m. Since at least one value of the state changes (case W), $\Delta_D < 0$. □

Lemma 16. *The FSS algorithm stops; that is, eventually $E_R \in \mathcal{O}$ and also $E_R \in \widehat{\mathcal{G}}$.*

Proof. Let us define $D_{\min}(\mathcal{O}) = \min(D(O[i]))$ and $i_{\min} = \operatorname{argmin}_i(D(O[i]))$, $1 \le i \le |\mathcal{O}|$, in each iteration.

Since the algorithm traces elements of queue $\mathcal{O}$ after each other (see lines (15)–(23)), in each iteration it is true that the actual state $\mathcal{O}[i_{\min}]$ will be traced after $i_{\min}$ iterations. Thus according to Lemmas 13 and 15, after $i_{\min}$ iterations $D_{\min}$ will decrease.

Repeating this process results in decreasing series of $D_{\min}(\mathcal{O})$ integer values, thus $D_{\min}(\mathcal{O})$ eventually reaches 0, where necessarily $E_R \in \mathcal{O}$ (and also $E_R \in \widehat{\mathcal{G}}$; see lines (21)-(22)). The iteration stops in line (15) and the algorithm terminates after executing lines (23)–(30). □

Theorem 17. *The list of states, generated by algorithm FSS corresponds to a shortest schedule for the given segmentation.*

Proof. Let $\widehat{A}_R = (\widehat{\mathcal{G}}, D)$ be the schedule connection graph generated by the FSS algorithm. (Note that edges in D are implicitly represented by the *parent* mapping.)

The algorithm starts with the node $(\otimes)_k$ (see line (13)); therefore, $\widehat{A}_R$ contains the start-state $(\otimes)_k$. According to Lemma 16, $\widehat{A}_R$ also contains the endstate E_R.

The search mechanism of the algorithm (breadth-first search) provides the shortest path from the start-state to the endstate. Since the algorithm applies the tracing rules (see line (19)), therefore, $\widehat{A}_R \subseteq A_R$; thus the $\widehat{A}_R$ component of A_R, built by the algorithm, contains the shortest path in A_R from $(\otimes)_k$ to E_R. According to Lemma 12 this path corresponds to one of the shortest paths for the given segmentation. □

Theorem 18. *The OMTS and OMTS-A algorithms find an optimal schedule for the MWCTT for a given node set, link set, communication link set triplet (N, L, L_C), and a maximum width k.*

Proof. The OMTS algorithm tries every possible segmentation for the given communication graph. According to Theorem 17 the FSS algorithm finds the shortest schedule for each given segmentation. Therefore, OMTS finds the global optimum.

The OMTS-A algorithm also enumerates every possible segmentations but skips the cases where the result would trivially be worse than the current best. Thus OMTS-A also finds an optimal solution. □

Theorem 19. *If the number of nodes in the communication graph is n then the computational complexity of the OMTS algorithm is a polynomial function of n for a given maximum width k_{max}.*

Proof. The FSS algorithm traces each state in lines (15)–(22). The *dequeue* operation can run in constant time. The cardinality of $\mathcal{P}(\mathcal{R}) \setminus \emptyset$ is constant for a given k; thus the number of iterations in the loop of lines (17)–(22) is also constant. The set operations, checking of tracing rules, queue, and map updates in the body of the loop in lines (18)–(22) have at most $O(n)$ complexity for a given k. Therefore, the complexity of tracing any state is $O(n)$.

Let the number of states traced by the FSS algorithm (i.e., the number of iterations performed by the loop in lines (15)–(22)) be denoted by I_{FSS}. I_{FSS} clearly cannot be more than the number of all possible states. Let us consider a k-wide segmentation R with segment lengths of $l_1, l_2, \ldots, l_k$. Then the ith element of a state can be either $R_{i,1}$, or $R_{i,2}, \ldots,$ or R_{i,l_i} or $\otimes$ ($l_i + 1$ possibilities). Therefore the I_{FSS} number of possible states for segmentation R is

$$I_{\text{FSS},R} = (l_1 + 1) \cdot (l_2 + 1), \ldots, (l_k + 1). \tag{9}$$

Using the inequality of arithmetic and geometric means

$$I_{\text{FSS},R} \le \left(\frac{l_1 + l_2 + \cdots + l_k + k}{k}\right)^k = \left(\frac{n+k}{k}\right)^k, \tag{10}$$

which has the complexity of $O(n^k)$. Taking into consideration the complexity of tracing one state ($O(n)$), the complexity of the FSS algorithm is

$$C_{\text{FSS}}(n) = O\left(n^k \cdot n\right) = O\left(n^{k+1}\right). \tag{11}$$

The I_{OMTS} number of iterations when FSS is called in the OMTS algorithm is equal to the sum of the numbers of m-wide segmentations, where $m = 1, 2, \ldots, k$:

$$\begin{aligned} I_{\text{OMTS}}(n) &= \binom{n}{k} + \binom{n}{k-1} + \cdots + \binom{n}{1} \\ &< n^k + n^{k-1} + \cdots + n^1 < k \cdot n^k. \end{aligned} \tag{12}$$

Therefore,

$$I_{\text{OMTS}}(n) = O\left(n^k\right), \tag{13}$$

and the complexity of the OMTS algorithm can be computed from I_{OMTS} and C_{FSS} as

$$C_{\text{OMTS}}(n) = O\left(n^k \cdot n^{k+1}\right) = O\left(n^{2k+1}\right). \tag{14}$$

□

4. Analytical Results

To compare the scheduling methods, measurement metrics are defined.

4.1. Turnaround Time. Turnaround time, denoted with T_T is the time required to deliver a packet from a node to all other nodes and get it back to the source node.

In linear TDMA scheduling the worst case delivery time can easily be calculated for an n-node network, as follows.

In the network asynchronous events occur (e.g., measurements), for which events the network reacts by sending messages (e.g., alerts). After this event, the node first has to wait for its transmission time slice. The worst case is when the node transmitted just before the measurement event, thus it has to wait a full period (n time slices) for the next transmission time. Second, at most n time slices are required for the message to turn around. Therefore, in the worst case the delivery time is

$$T_{T,\text{linear}} = 2n. \tag{15}$$

For a k-wide multi-TDMA schedule a lower bound can be calculated. Let c be the length of the period. In the worst case a node has to wait for c time slices to transmit and $k \cdot c$ time slices are required for the message to turn around; thus

$$T_{T,\text{multi}} = c + k \cdot c = (c + 1) \cdot c. \tag{16}$$

The length of the period must be at least the size of the largest segment in the segmentation; therefore, the following lower bound can be given for c:

$$c \geq \frac{n}{k}. \tag{17}$$

Therefore,

$$T_{T,\text{multi}} \geq \frac{k+1}{k} n > n = \frac{T_{D,\text{linear}}}{2}. \tag{18}$$

Note that the maximal gain in delivery time is approximately 2, compared to linear TDMA scheduling. The lower bound is a good approximation, when

(i) the segmentation contains segments with equal size,

(ii) in all time slices one node transmits from each segment,

(iii) k is large.

4.2. Power Consumption. Power consumption can be considered in two different ways. The energy, which is necessary to deliver a packet from a node to all other nodes is denoted by $E_{\text{packet,linear}}$ for the linear TDMA and $E_{\text{packet,multi}}$ for the multi-TDMA scheduling. The average energy consumed in a time slice is denoted by $E_{\text{avg,linear}}$ and $E_{\text{avg,multi}}$ for linear and multi-TDMA, respectively.

In our model energy E_{tx} is necessary to transmit and E_{rx} to receive a packet, regardless of other parameters (e.g., link quality).

In both linear and multi-TDMA scheduling each measurement is transmitted by each node exactly once. In each time slice E_{tx} is required for the transmission and E_{rx} is required for the reception of the message containing the measurement. Thus the energy required to broadcast a measurement can be calculated with the following equation for both the linear and the multi-TDMA scheduling:

$$E_{\text{packet,linear}} = E_{\text{packet,multi}} = n \cdot (E_{tx} + E_{rx}). \tag{19}$$

In linear TDMA scheduling in each time slice exactly one node transmits and one node receives the message; thus the average energy consumption is

$$E_{\text{avg,linear}} = E_{tx} + E_{rx}. \tag{20}$$

In multi-TDMA scheduling each node will receive and transmit once in a period; therefore, the multi-TDMA requires the same energy amount in a period with length c as the linear TDMA for a longer period with length n. Thus the average energy consumption for the multi-TDMA scheduling can be calculated with the following equation:

$$E_{\text{avg,multi}} = \frac{n}{c} \cdot (E_{tx} + E_{rx}) = \frac{n}{c} \cdot E_{\text{avg,linear}}. \tag{21}$$

Thus the price of faster delivery time is the increase in energy consumption. Since $c \geq n/k$,

$$E_{\text{avg,multi}} \leq k \cdot E_{\text{avg,linear}}. \tag{22}$$

The average energy consumption is largest when all the segments are equal and there is one transmitter from each segment in each time slice; in this case the multischeduler algorithm consumes k times more energy than the linear one.

5. Test Results

5.1. Computation Time. To test the computational complexity of the scheduling algorithms, both the OMTS and OMTS-A algorithms were implemented in Perl. For reference a brute-force algorithm was also implemented, which did not use the proposed states. For test cases random graphs with controlled numbers of node (n) and link (b) were generated, as follows. n nodes were placed randomly in a square-shaped area and n links for the Hamiltonian cycle were added. Afterwards $b - n$ additional links were added iteratively between the two nodes, which had the shortest distance and no existing link yet. In the test c was chosen as 15% of all the possible number of links; that is, $b = 0.15 \cdot (n \cdot (n-1)/2)$. The value of n varied from 10 to 100. For each n, 10 different random topologies were generated and the average and standard deviation of the runtimes of the three algorithms were measured, where it was feasible. The test platform was a Lenovo Thinkpad T430 with an Intel Core i5-3210M CPU at 2.5 GHz and 8 GB RAM. In the tests $k = 3$ was used. The results are shown in Table 1, where the average and standard deviation values are listed for each experiment. The results are also plotted in Figure 8 in a logarithmic scale, showing the results of all the experiments.

The brute-force algorithm quickly reached the feasibility limit of one day at $n = 15$, the OMTS algorithm could compute schedules for $n = 40$, while OMTS-A was usable even for $n = 100$. The speed differences are clearly shown in Figure 8: the computational time of the brute-force algorithm increases with the number of nodes much faster than in the case of the proposed algorithms. The behavior of algorithms OMTS and OMTS-A are similar, but due to the early cuts, the complexity of OMTS-A is two orders of magnitude lower. Notice the higher variance of OMTS-A, which is due to the fact that the efficiency of early cuts depends largely on the actual topology.

Table 1: Computation times of algorithms OMTS, OMTS-A, and brute-force (bf).

n	t_{OMTS} (s)		$t_{\text{OMTS-A}}$ (s)		t_{bf} (s)
	Avg.	Std. dev.	Avg.	Std. dev.	
10	0.3	0.01	0.005	0.005	4.7
11	0.5	0.004	0.014	0.007	28.5
12	0.9	0.02	0.033	0.01	161.4
13	1.5	0.02	0.023	0.01	1578
14	2.3	0.03	0.07	0.05	21 182
15	3.6	0.03	0.08	0.03	—
20	23	0.2	0.22	0.3	—
25	101	0.8	0.6	1.1	—
30	383	33	2.6	0.7	—
40	2380	71	19	25	—
50	12 434	280	107	52	—
60	—	—	228	122	—
70	—	—	1736	261	—
80	—	—	7404	3091	—
90	—	—	11 029	2462	—
100	—	—	64 066	23 836	—

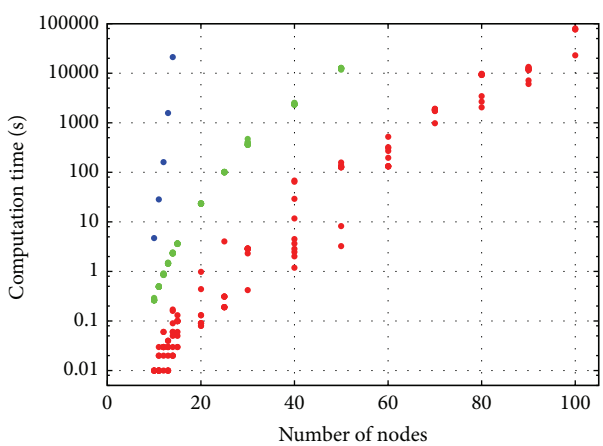

Figure 8: Computation times of algorithms OMTS (green), OMTS-A (red), and brute-force (blue).

For $k = 3$, used in the tests, the theoretical complexity of the OMTS and OMTS-A algorithms is $O(n^7)$, which corresponds well to the measurement results for $n \geq 20$.

5.2. Performance Comparison. The proposed method is compared to a graph coloring based reference algorithm described in [11]. An example network with 16 nodes was constructed, where 15 nodes form a cycle with symmetrical connections between the neighbors and all nodes in the cycle have symmetrical connection to the central node; see Figure 9(a). The connectivity graph with a Hamiltonian cycle is shown in Figure 9(a) with straight and dashed lines, respectively. From the connectivity graph and the Hamiltonian cycle the constraint graph is constructed for the method described in [11] with constraints C_1 and C_2, as shown in Figure 9(b).

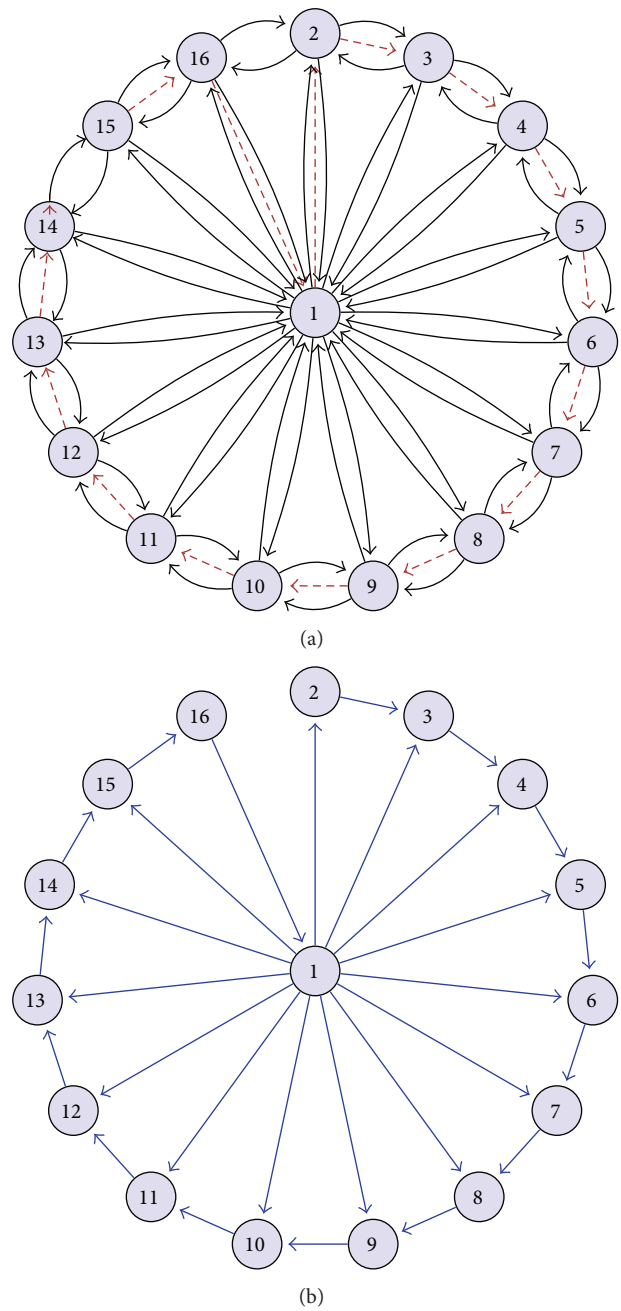

Figure 9: (a) The connectivity graph of the test network (straight lines) and the Hamiltonian cycle (dashed lines). (b) The constraint graph of the network according to the model described in [11].

Although the method described in [11] contains heuristics to solve the graph coloring problem, in the example of Figure 9 the optimal coloring was found. In Figure 10(a) the constraint graph is colored with 3 colors, which is indeed the minimum, because the graph contains triangles. The sequence of the colors was chosen to be purple $\rightarrow$ green $\rightarrow$ blue; that is, purple nodes transmit in time slices $t = 3m + 1$, green nodes transmit in time slices $t = 3m + 2$ and blue nodes transmit in time slices $t = 3m$ $(m \in \mathbb{N})$.

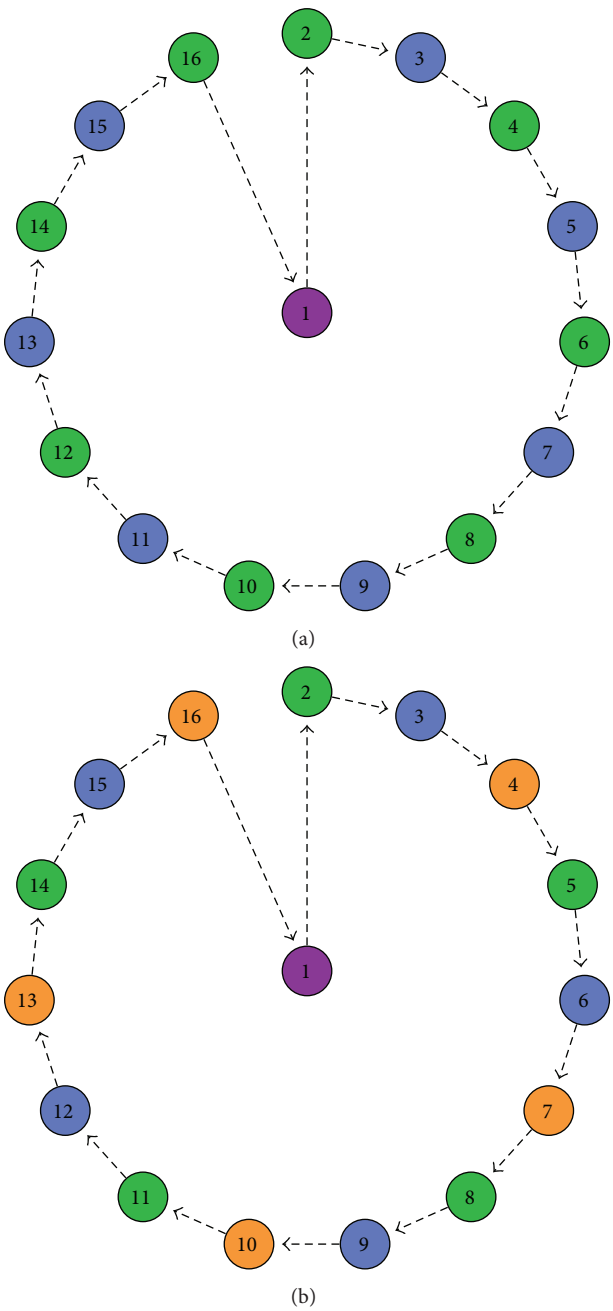

Figure 10: The coloring of the graph according to the method described in [11] (a) and the proposed method (b).

The optimal schedule found by the proposed OMTS algorithm for the same network is shown in Figure 10(b). The optimum was found at $k = 3$ and the graph is colored with 4 colors, with sequence of purple → green → blue → orange.

The timing diagrams of the reference and the proposed algorithms are shown in Figure 11. The arrows denote the transmission of the information originating from node 1; the dashed vertical lines denote the point, where this information reaches all nodes and is returned back to node 1.

In Figure 11(a) the timing diagram for the reference algorithm is shown. After the information to be disseminated first appears at node 1, the node has to wait for its transmission time slice, which requires 3 time slices in the worst case. From that point the information reaches node 1 again in the 23rd time slice. Thus the propagation of the information requires $3 + 23 = 26$ time slices.

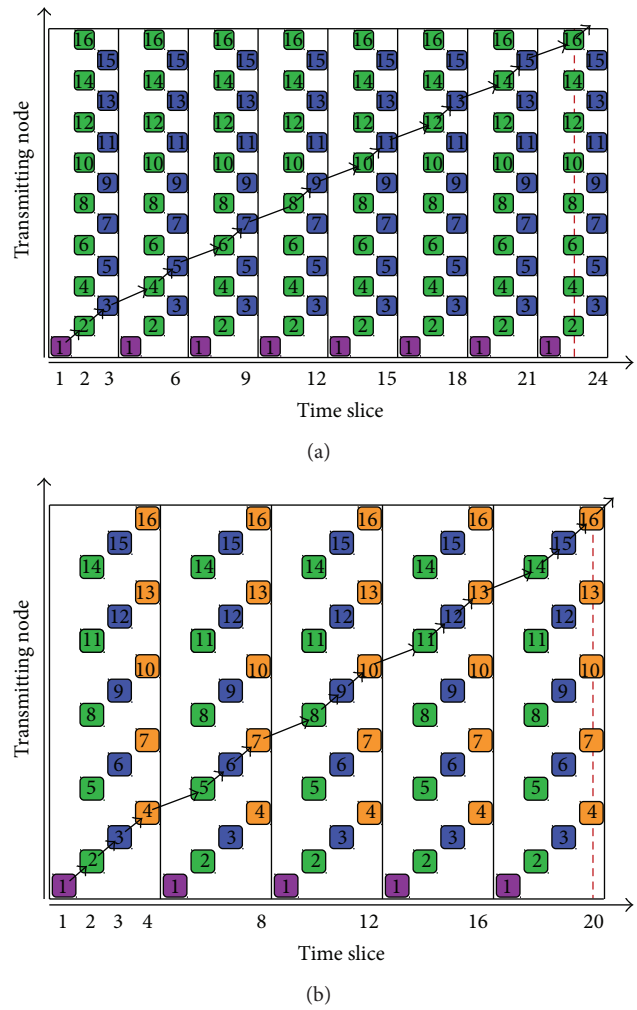

Figure 11: The timing diagram of the schedule generated by the reference algorithm (a) and the proposed algorithm (b).

In Figure 11(b) the timing diagram for the proposed OMTS algorithm is shown. In worst case, node 1 has to wait 4 time slices to transmit, and the information reaches the last node in the 20th time slice. Thus with this schedule $4 + 20 = 24$ time slices are required to disseminate the information originating from node 1. The example clearly illustrates the ability of the OMTS algorithm to find the optimal solution in ring topology networks, where messages must be propagated to all other nodes.

6. Conclusion

The Minimum Worst Case Turnaround Time problem was defined. A novel algorithm (OMTS) was proposed for ring-topology networks to compute multi-TDMA schedules for

the MWCTT problem. The algorithm was proven to find the optimal schedule (with the shortest turnaround time), which, in worst case, provides the fastest means to distribute information from any node to all other nodes and back to the source node in the network. It was proven that the multi-TDMA scheduling can provide at most twofold acceleration with respect to linear scheduling, at a price of increased energy consumption.

An accelerated version of the algorithm (OMTS-A) was also proposed, which narrows the search space during run-time using heuristics but still guarantees the optimal solution.

In addition to the mathematical analysis of the proposed algorithms, extensive tests were performed to compare the performances of the proposed algorithms and the conventional (brute-force) search method. While the brute-force algorithm can be used for small networks only (less than 15 nodes), the proposed algorithm can solve problems with even 100 nodes in reasonable time.

The performance of the designed schedule was also compared to that of another algorithm. In case of the MWCTT problem the proposed solution was able to find better (optimal) schedule.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the Hungarian Government and the European Union and cofinanced by the European Social Fund under the project TÁMOP-4.2.2.C-11/1/KONV-2012-0004. Gergely Vakulya was supported by the European Union and cofinanced by the European Social Fund in the framework of TÁMOP 4.2.4. A/2-11-1-2012-0001 "National Excellence Program."

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] G. Vakulya and G. Simon, "Design of a sensor network based security system," in *Proceedings of the 7th IEEE International Symposium on Intelligent Signal Processing (WISP '11)*, pp. 15–19, Floriana, Malta, September 2011.

[3] K. Kredo II and P. Mohapatra, "Medium access control in wireless sensor networks," *Computer Networks*, vol. 51, no. 4, pp. 961–994, 2007.

[4] V. Paruchuri, S. Basavaraju, A. Durresi, and R. Kannan, "Random asynchronous wakeup protocol for sensor networks," in *Proceedings of the 1st International Conference on Broadband Networks (BroadNets '04)*, pp. 710–717, October 2004.

[5] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1567–1576, June 2002.

[6] A. Barroso, U. Roedig, and C. Sreenan, "μ-MAC: an energy-efficient medium access control for wireless sensor networks," in *Proceedings of the 2nd European Workshop on Wireless Sensor Networks (EWSN '05)*, pp. 70–80, February 2005.

[7] P. K. Pal and P. Chatterjee, "A survey on TDMA-based MAC protocols for wireless sensor network," *International Journal of Emerging Technology and Advanced Engineering*, vol. 4, no. 6, pp. 219–230, 2014.

[8] O. D. Incel, A. Ghosh, and B. Krishnamachari, "Scheduling algorithms for tree-based data collection in wireless sensor networks," in *Theoretical Aspects of Distributed Computing in Sensor Networks*, pp. 407–445, Springer, Berlin, Germany, 2011.

[9] S. C. Ergen and P. Varaiya, "TDMA scheduling algorithms for wireless sensor networks," *Wireless Networks*, vol. 16, no. 4, pp. 985–997, 2010.

[10] C. Florens and R. McEliece, "Scheduling algorithms for wireless ad-hoc sensor networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '02)*, vol. 1, pp. 6–10, November 2002.

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

[12] J. Grönkvist and A. Hansson, "Comparison between graph-based and interference-based STDMA scheduling," in *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 255–258, October 2001.

[13] S. Cui, R. Madan, A. Goldsmith, and S. Lall, "Energy-delay tradeoffs for data collection in TDMA-based sensor networks," in *IEEE International Conference on Communications*, vol. 5, pp. 3278–3284, May 2005.

[14] E. J. Duarte-Melo and M. Liu, "Data-gathering wireless sensor networks: organization and capacity," *Computer Networks*, vol. 43, no. 4, pp. 519–537, 2003.

[15] A. Sridharan and B. Krishnamachari, "Max-min fair collision-free scheduling for wireless sensor networks," in *Proceedings of the IEEE International Conference on Performance, Computing, and Communications*, pp. 585–590, 2004.

[16] J. Mao, Z. Wu, and X. Wu, "A TDMA scheduling scheme for many-to-one communications in wireless sensor networks," *Computer Communications*, vol. 30, no. 4, pp. 863–872, 2007.

[17] K. Kalpakis, K. Dasgupta, and P. Namjosh, "Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks," *Computer Networks*, vol. 42, no. 6, pp. 697–716, 2003.

[18] Y. Revah and M. Segal, "Improved lower bounds for data-gathering time in sensor networks," in *Proceedings of the 3rd International Conference on Networking and Services (ICNS '07)*, p. 76, IEEE, June 2007.

[19] H. Choi, J. Wang, and E. A. Hughes, "Scheduling for information gathering on sensor network," *Wireless Networks*, vol. 15, no. 1, pp. 127–140, 2009.

[20] P. Djukic and S. Valaee, "Link scheduling for minimum delay in spatial re-use TDMA," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pp. 28–36, Anchorage, Alaska, USA, May 2007.

[21] S. Gandham, Y. Zhang, and Q. Huang, "Distributed time-optimal scheduling for convergecast in wireless sensor networks," *Computer Networks*, vol. 52, no. 3, pp. 610–629, 2008.

[22] N.-L. Lai, C.-T. King, and C.-H. Lin, "On maximizing the throughput of convergecast in wireless sensor networks," in *Advances in Grid and Pervasive Computing*, vol. 5036, pp. 396–408, Springer, Berlin, Germany, 2008.

[23] G. Vakulya and G. Simon, "Extended round-robin TDMA scheduling scheme for wireless sensor networks," in *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference: Instrumentation and Measurement for Life (I2MTC '13)*, pp. 253–258, Minneapolis, Minn, USA, May 2013.

[24] C. Bazgan, M. Santha, and Z. Tuza, "On the approximation of finding a(nother) Hamiltonian cycle in cubic Hamiltonian graphs," *Journal of Algorithms*, vol. 31, no. 1, pp. 249–268, 1999.

[25] L. Pósa, "Hamiltonian circuits in random graphs," *Discrete Mathematics*, vol. 14, no. 4, pp. 359–364, 1976.

[26] I. Shields, *Hamilton cycle heuristics in hard graphs [Ph.D. thesis]*, North Carolina State University, 2004.

[27] A. Dharwadker, "A new algorithm for finding Hamiltonian circuits," in *Proceedings of the Institute of Mathematics*, 2004.

[28] A. Bertrand and M. Moonen, "Seeing the bigger picture: how nodes can learn their place within a complex ad hoc network topology," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 71–82, 2013.

[29] Á. Orosz, G. Róth, and G. Simon, "TDMA scheduling in fault tolerant wireless sensor networks," in *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC '12)*, pp. 1169–1173, IEEE, May 2012.

[30] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "DESYNC: self-organizing desynchronization and TDMA on wireless sensor networks," in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN '07)*, pp. 11–20, April 2007.

[31] Á. Orosz, G. Róth, and G. Simon, "Efficient TDMA scheduling algorithms for sensor networks containing multiple rings," in *Proceedings of the IEEE 8th International Symposium on Intelligent Signal Processing (WISP '13)*, pp. 126–130, Funchal, Portugal, September 2013.