A Test-Based Scheduling Protocol (TBSP) for Periodic Data Gathering in Wireless Sensor Networks

Mario Orne Díaz-Anadón and Kin K. Leung

Electrical Engineering Department
Imperial College, London SW7 2BT, United Kingdom
{orne.diaz06, kin.leung}@imperial.ac.uk

Abstract. We propose TBSP, a TDMA protocol for gathering information periodically from multiple data sources to a central location across multiple hops. TBSP incurs a lower overhead than the existing protocols because it only requires the sensor nodes to keep track of their own schedule, whereas in the existing protocols each node needs to know the schedule of its neighbors. In order to gain a transmission slot, the sensor nodes transmit in different slots until they find one with sufficiently low interference. TBSP provides mechanisms to reduce the probability that the sensor nodes steal each other's slots or thwart each other's efforts. Our simulation study reveals that TBSP provides great energy savings if the network changes slowly and the sensor nodes can wait a dozen TDMA frames before obtaining a transmission slot. TBSP also provides the advantage of being more likely to keep the sensor network connected in sparse networks.

Keywords: adaptive; wireless sensor networks; TDMA;

1 Introduction

Wireless Sensor Networks (WSNs) are used to collect signals such as temperature, acceleration or video from multiple locations within a certain geographic area. They can be deployed faster and at lower cost than their wired counterparts, which has led them to find applications in a range of environments, including water pipes, forests and battlefields [1]. A WSN consists of a large number of *sensor nodes*, which are devices equipped with a battery, a sensing module, a microcontroller and a radio transceiver. Usually, the sensor nodes have little energy and computational power.

We propose a Test-Based Scheduling Protocol (TBSP) that obtains and maintains a TDMA schedule for periodically transmitting packets from a set of sensor nodes referred to as *data sources* to a special node referred to as *data sink* across multiple hops. TBSP is designed for applications in which the set of data sources, their data rate, or the properties of the wireless links change infrequently. The main goal of TBSP is to adapt the transmission schedule to those infrequent changes in an energy-efficient way. The focus of TBSP is the adaptation of the schedule rather than its initial construction because the initial scheduling overhead is negligible in long-running monitoring applications.

TBSP is scalable because it is distributed and its buffering requirements do not grow with the network size. Our simulation studies reveal that TBSP consumes less energy than the existing protocols in networks that change relatively slowly. In addition, TBSP is more likely to obtain a collision-free schedule than the existing protocols because the existing protocols assign a given transmission slot to a set of nodes if the hop distance between them exceeds a certain number, whereas TBSP only assigns them the same slot if it has been proved empirically that they tolerate each other's interference.

The rest of the paper is organized as follows. Section 2 reviews some related work, Section 3 describes our protocol, Section 4 presents our simulation methodology and results, and Section 5 concludes the paper.

2 Related work

MAC protocols for wireless sensor networks can be classified as contention-based or TDMA-based [4]. The contention-based protocols respond quickly to traffic demand variations but waste bandwidth and energy due to back-off periods, packet collisions, idle listening and overhearing. The TDMA-based protocols avoid these problems during the data transmission phase but incur in scheduling and synchronization overhead. Overall, if the data are transmitted frequently and periodically, which is the case as in the applications that we are considering, the TDMA-based protocols outperform the contention-based protocols. Some hybrid protocols seeking to combine the benefits of contention and TDMA have been proposed [4], but for stationary networks they are less efficient than TDMA [5]. Therefore, we focus our attention in TDMA protocols, and more specifically in distributed TDMA protocols since centralized protocols such as [3] incur significant maintenance overhead in large networks.

Let us make some general definitions about distributed TDMA protocols. Since we are interested in periodic data-gathering in multihop networks, we assume that a *routing tree* rooted at the data sink has been established. In this tree, each node receives packets from its child nodes and relays these packets to its parent node. Time is divided in TDMA frames, and each TDMA frame consists of a number of transmission slots. The schedule is relatively periodic in the sense that, if a node is allocated a certain transmission slot, it is allocated that transmission slot in several consecutive TDMA frames. Let X be a sensor node and let Y be its parent node. We say that a transmission slot allocated to X is *unfeasible* if the joint interference from other nodes during that slot is so large that Y cannot receive packets from X or X cannot receive ACKs from Y. Feasible slots may become unfeasible, forcing their owners to seek new transmission slots. If a node's slot becomes unfeasible because of the interference of some nodes that recently started using that slot, we say that the node has been *expelled* from its slot or that the node has suffered an *expulsion*.

TRAMA [8] is a distributed TDMA scheduling protocol that adapts the schedule to traffic changes very quickly but also consumes much energy. FLAMA [7] greatly reduces the energy consumption by taking advantage of the fact that the adaptivity requirements in many monitoring applications are modest. However, the nodes in FLAMA incur significant overhead in either keeping track of the priorities of all their two-hop neighbors or listening in slots in which they do not receive packets. In addition, FLAMA does not minimize the buffering requirements or the *traversal time*, which we define as the maximum number of TDMA frames needed by a packet from the data sources to reach the data sink. FlexiTP [5] solves these problems by considering latency and memory requirements in its slot-selection algorithm. In Section 4, we present simulation results comparing the performance of FlexiTP with that of our protocol.

To our knowledge, every existing distributed TDMA protocol for periodic data gathering in WSNs uses a certain *interference model*, which is a model to decide when a node's interference is negligible. The most common interference model is to neglect the interference originated more than $k = 2$ hops away, which is a model referred to as the k-hop interference model. Unlike the existing algorithms, TBSP does not use an interference model. We say that TBSP is a *test-based* protocol because a node is allocated a given slot only if that slot has been proved feasible in an empirical *test* that consists in transmitting a dummy packet in that slot and checking whether the reception of this packet is acknowledged with an ACK packet.

The existing protocols obtain fewer feasible slots than TBSP does because their interference models may fail whereas TBSP bases its scheduling decisions in empirical tests. When the existing protocols assign an unfeasible slot to a certain node X, X incorrectly assumes that its parent node is unreachable, which makes X disconnected from the network if X does not have other neighbors. TBSP is less likely to suffer this connectivity problem because, up to a maximum number of times, if X cannot communicate with its parent node during a slot, X tries to change its slot, not its parent. Another disadvantage of the existing protocols is that they require that every node listens to its neighbors in case it receives messages from them indicating changes in their schedules. In contrast, TBSP does not incur this idle-listening overhead because all scheduling decisions are taken distributedly and without knowledge of other nodes' schedules.

3 TBSP

The purpose of the Test-Based Scheduling Protocol (TBSP) is to obtain and maintain a TDMA transmission schedule to transfer the data from the data sources to the data sink periodically. The minimum unit that can be scheduled to a node per TDMA frame is a Data Subframe (DS), which consists of two data transmission slots. Since each DS should be used during at least 20 consecutive TDMA frames, TBSP is only suitable for relatively static networks. Although we assume infrequent network changes, we regard those changes as very important, and the goal of TBSP is to adapt the schedule to them in an energy-efficient way.

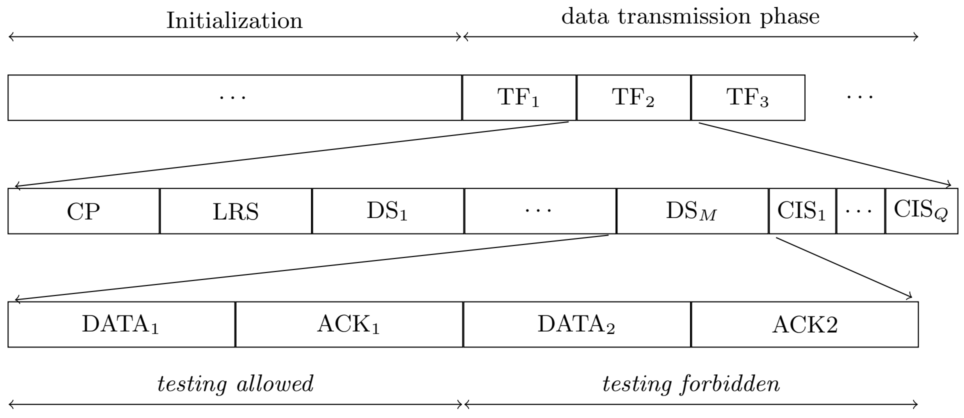

Fig. 1. Timing diagram of TBSP.

Every node seeking to obtain a DS is referred to as a *contender*, and the DS that it seeks to obtain is referred to as its *target DS*. A simplified description of the mechanism whereby a contender gains a DS is as follows. First, the contender listens during a TDMA frame in order to identify DSs with low interference. Then, the contender selects one of those DSs as its target DS. The target DS may be unfeasible due to the hidden terminal problem, and the contender determines whether that is the case by testing its target DS. The test consists in transmitting a packet referred to as *Testing Packet* during its target DS and checking whether it receives an ACK in response. The contender repeats the whole process with different target DSs until the test is positive, in which case it has gained its target DS and starts using it.

3.1 Timing diagram

Figure 1 shows that TBSP consists of an initialization phase followed by a data transmission phase. The goals of the initialization phase are to obtain a routing tree and to synchronize the sensor nodes for the first time. These two goals are achieved by executing the distributed Bellman-Ford algorithm and the FTSP protocol [6], respectively. The data transmission phase extends during the rest of the network lifetime and consists of TDMA Frames (TFs), each of which is composed of a CSMA Period (CP), a Listening Request Slot (LRS), M Data Subframes (DSs) and Q Collision-Indication Slots (CISs). Every node keeps its transceiver active during the CP, the LRS, any DS during which it is scheduled to transmit or receive, and the CISs. We explain each of these slots as follows.

The CP is a CSMA period with two purposes. First, it is used to request and receive neighbor information by nodes that need to change their parent node in the routing tree. Second, it is used to request and receive synchronization information by nodes that are not engaged in periodic data transmissions. Other nodes do not use this synchronization method because the ACKs that they receive from their parents as a response to their periodic packet transmissions already contain synchronization information.

The Listening Request Slot (LRS) is a CSMA period used to transmit at most one Listening Request Packet (LRP), which is a packet whereby the contenders request their parents to listen during their target DSs.

The DSs are the Data Subframes that the sensor nodes use for their periodic packet. A DS is the minimum time unit that can be assigned to a node. It consists of two DATA slots with their respective ACK slots. We refer to these slots as DATA_1, ACK_1, DATA_2 and ACK_2. The ACK packets include synchronization information. During the DATA_1 slot the contenders can transmit Testing Packets, but not during the DATA_2 slot. If a node X receives a packet during the DATA_2 slot but not during the DATA_1 slot, the first packet was probably lost because another node, which we call Y, generated a collision by transmitting a Testing Packet. In this case, we say that Y is the *originator* of a *contender-induced collision* and X is its *victim*. If Y receives an ACK during ACK_1, Y seizes its target DS, thereby expelling X, which is undesirable.

The Collision-Indication Slots (CISs) are Q very short slots that are used to reduce the number of expelled nodes by propagating a *collision indication*, which is an indication that a contender-induced collision has occurred. We say that a node receives a collision indication if it senses the channel busy during any of the CISs. Collision indications are propagated as follows. In CIS_1, the victim transmits a dummy packet in order to make the channel busy. When the victim's one-hop neighbors sense the collision indication, they propagate it to their own one-hop neighbors during CIS_2. This process is repeated until CIS_Q so that the collision indication reaches all the nodes within Q hops of the victim. Hopefully, the originator of the collision also receives the indication, in which case it refrains from seizing its target slot in order to prevent an expulsion. This collision-indication mechanism is inappropriate in noisy environments because noise might be interpreted as a collision indication. In such noisy environments, we recommend not using the CISs at all (i.e. setting $Q = 0$), which according to our simulation results provides degraded but sufficient results. As an added benefit, the suppression of the CISs simplifies the protocol.

3.2 Algorithm to gain a DS

Every contender uses a *DS-selection algorithm* to select its target DS. Then, each contender uses a *wait-selection algorithm* to decide the number of TDMA frames it waits before testing its target slot. This wait aims to address the hidden terminal problem by reducing the probability that two contenders' attempt to gain the same slot simultaneously, which would reduce the probability of obtaining a DS for both nodes. After the wait, the contender contends to transmit a Listening Request Packet in every TF until it succeeds. When it succeeds, the contender transmits a Testing Packet during the DATA_1 slot of its target DS. If it receives an ACK, it has obtained its target DS. Otherwise, the contender repeats the whole process until it obtains a DS. We detail the DS-selection algorithm and the backoff algorithm as follows.

The DS-selection algorithm Each contender selects its target DS among the DSs that verify the following properties. First, the signal level detected during that DS by the contender must be low. Second, the contender should not have tested that DS yet. Third, if the contender seeks the DS in order to relay a packet from a child node, the number of the target DS must exceed the number of the DS in which it is scheduled to receive packets from its child. This third property is discussed in [5] and ensures small traversal time. We refer to the DS with the smallest number that verifies the previous properties as $\mathrm{DS}_{\min}$. If the contender has made fewer than a certain number of attempts (in our simulations this number is four), the contender selects $\mathrm{DS}_{\min}$ as its target DS. Otherwise, the contender selects the first DS that verifies the above properties and whose number exceeds the number of $\mathrm{DS}_{\min}$ by a small random integer.

The wait-selection algorithm Every contender X keeps track of the number n_c of consecutive TFs without collision indications. Node X also selects a *wait parameter* n_b as a random integer between 0 and $N_b - 1$ (in our simulations, $N_b = 8$). In each TF, the algorithm executed by X depends on the value of $n_c - n_b$:

- If $n_c - n_b$ is negative, X does not contend during the LRS of the current TF. If X senses a collision indication, there is probably a contender nearby. In order to avoid ruining the efforts of that contender, X resets n_b as a random integer between N_b and $2N_b - 1$.
- If $n_c - n_b$ is non-negative, X contends to transmit a Listening Request Packet (LRP) in the LRS using CSMA. The backoff period to transmit the LPR is a random quantity that is shorter for contenders closer to the data sink or with full buffers. After X transmits the LRP, it transmits its Testing Packet in the DATA_1 slot of its target DS. If X receives an ACK from its parent Y during slot ACK_1, X has obtained its target DS, unless the ACK packet from Y indicates that Y's buffer is full, in which case X resets its backoff parameter n_b as a random integer between $3N_b$ and $4N_b - 1$. If, on the other hand, X receives no response from Y, X sets $n_c = 0$ and n_b with a random integer between 0 and $N_b - 1$, and reruns the DS-selection algorithm.

4 Performance evaluation

In order to evaluate the performance of TBSP, we develop a simulator using the Python programming language. The code of our simulator and all our simulation parameters are available in [2]. We compare our protocol with FlexiTP [5] because FlexiTP is the only existing distributed and adaptive TDMA protocol for periodic data gathering that shares our goal of minimizing the buffering requirements and the traversal time. The major difference between TBSP and FlexiTP is that

FlexiTP uses the traditional 2-hop interference model whereas TBSP follows our test-based approach of not keeping track of other nodes' schedules and assigning a given DS to a set of nodes if they have been proved empirically to be able to tolerate each other's interference.

4.1 Simulation scenario

The transmit and receive power are 63 mW, which are common in current hardware. The noise figure of the receiver is 4.8 dB. The wireless channel has attenuation exponent of 3.5, path loss at 100 m of 80 dB, and standard deviation of the log-normal fading of $\sigma_f = 8\,\mathrm{dB}$. A transmission succeeds if the SINR at the receiver exceeds 20 dB. Although there is no common transmission range for all the network, we define t as a node's hypothetical transmission range if there were no fading and interference; in our case, t is 48 m.

The sensor nodes are randomly deployed within a square of side $3t$ and the data sink lies in the middle of one of the sides of the square. We define the *node density* ρ as $N(\pi t^2)/A$, where N is the number of nodes and A is the area of the square. Therefore, ρ is the average number of one-hop neighbors per node. We discard any deployments where more than 10 % cannot reach the data sink, which is rare for $\rho \geq 7$. All the nodes are data sources and the routing tree is the shortest path tree. Our results are computed as the average of 700 simulation runs.

In order to evaluate the adaptivity of the two protocols, we simulate a network that changes periodically at discrete points in time. We refer to each interval during which no changes occur as a *network cycle*. At the beginning of each network cycle, we remove a number $S = 3$ of nodes from the network and add an identical number of new nodes at random locations within the monitored area. These removals and additions force some sensor nodes to seek new parents and transmission slots. We compare the way in which FlexiTP and TBSP enable those nodes to obtain new slots as follows.

4.2 Total energy consumption E_t

We now define three energy metrics, all of which exclude the energy consumed in data transmissions, time synchronization and parent node selection because these components are the same for TBSP and FlexiTP. First, we define the *fixed energy consumption* E_f as the energy consumed per TDMA frame and per node when no contenders are present in the network. Second, we define the *variable energy consumption* E_v as the sum of extra energy consumed by all the nodes each time that a transmission slot is acquired. Third and most importantly, E_t is the total energy consumed per node and per TDMA frame and is given by

$$E_t = E_f + \frac{E_v f_s}{N}, \qquad (1)$$

where N is the total number of nodes in the network and f_s is the average number of slots that have to be gained in the network per TDMA frame.

We also define n_n as the number of contenders forced to seek a slot by a topology change, and n_e as the number of contenders forced to seek a slot by an expulsion. We define the *adaptation latency* as $l = n_w/n_n$, where n_w is the total number of TDMA frames waited by all the contenders before obtaining a slot. In other words, the adaptation latency is the average number of TDMA frames to required to gain a slot multiplied by $1 + \epsilon$, where $\epsilon = n_e/n_n$ is the expected number of expelled nodes each time that a node obtains a slot. We make this definition of adaptation latency in order to consider the negative influence of expulsions.

In FlexiTP, E_t is equal to E_v and proportional to T_f, which is the duration of the FTS. The FTS is a CSMA slot used by the FlexiTP nodes to notify their 2-hop neighbors of the slots that they are going to use. The adaptation latency l decreases linearly with T_f until a certain point. We refer to the l, T_f and E_t at this point as l_0, T_f^0 and E_t^0 respectively. In our simulations, we use set $T_f = T_f^0$ because it allows us to compute the energy consumption for any adaptation latency l_1 with the expression $E_t = E_t^0 l_0/l_1$.

In order to compare the energy consumption of FlexiTP with that of TBSP in a fair way, we have to consider the adaptation latency l simultaneously. Figure 2 presents the adaption latency as a function of the node density. FlexiTP's latency is much smaller than that of TBSP for the simulated parameters, and we take this into account when we compare E_t for the two protocols. Figure 2 shows that the use of CISs (which are the slots shown in Figure 1) in TBSP reduces the latency, but only for high node densities. For high node densities the CISs reduce the latency because they reduce the number n_e of expulsions. However, for low densities the CISs do not reduce the latency because n_e is already very small without using any CISs.

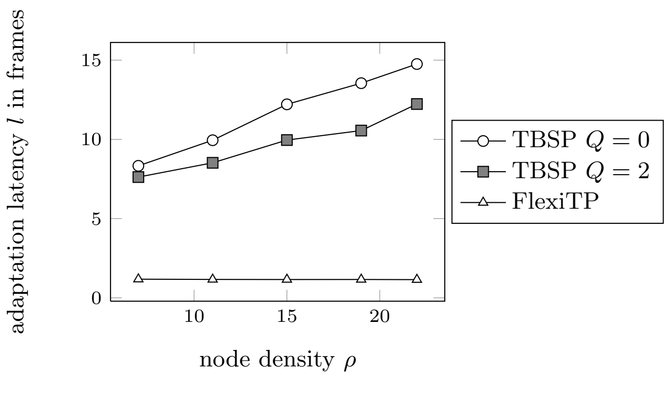

Fig. 2. Adaptation latency l. Note that although the latency of FlexiTP is up to 15 times smaller than that of TBSP, FlexiTP consumes at least 47 times more energy than TBSP does according to Table 1.

Table 1. Comparison of energy metrics. The last column shows $\bar{G}$, which is the number of times that TBSP outperforms FlexiTP in terms of E_t when we set the same maximum adaptation latency for the two protocols.

Simulation parameters		Energy metrics in mJ				
Protocol	ρ	E_f	E_v	E_t	G	$\bar{G}$
FlexiTP	7	18.425	0.0	18.43		
	14	31.017	0.0	31.02		
	21	38.916	0.0	38.92		
TBSP with $Q = 0$	7	0.031	65.9	0.36	51.2	6.2
	14	0.031	193.1	0.51	60.8	5.0
	21	0.031	335.6	0.59	66.0	4.5
TBSP with $Q = 2$	7	0.094	59.4	0.39	47.3	6.2
	14	0.094	157.0	0.49	63.3	6.4
	21	0.094	276.0	0.55	70.8	5.8

Table 1 compares FlexiTP and TBSP in terms of E_f, E_v and E_t for different node densities. Let us first examine E_f and E_v for the two protocols independently. The two protocols suffer an increase in the energy consumption with the node density. In the case of FlexiTP, this is because a larger T_f^0 is needed for a node to communicate its list of transmission slots to its increased number of neighbors. In the case of TBSP, this is because more DSs are used for periodic transmissions around each contender, which increases the number of failed attempts to gain a DS and the number of expulsions. The node density is also important because it determines the usefulness of the CIS mechanism. The CISs are only beneficial for high node densities, since for low node densities the slight reduction of the number of expelled nodes does not warrant the overhead it incurs.

Now let us examine in Table 1 the value of the total energy consumption E_t. This is the most important energy metric that we consider since it considers both E_f and E_v using (1). Equation (1) shows that the speed of change of the wireless links, which is controlled by f_s, greatly affects E_t. Table 1 uses $f_s = 1/20$, which implies that the network varies relatively slowly. For this value, the quotient between E_t for FlexiTP and TBSP, which we define as G, ranges between 47.3 and 70.8. However, in order to provide a fair comparison, we have to compare the two protocols for the same adaptation latency. We provide this fair comparison by defining $\bar{G}$. We define $\bar{G}$ as the number of times that TBSP outperforms FlexiTP when FlexiTP uses the shortest FTS that achieves the same adaptation latency as TBSP does. Table 1 shows that TBSP is approximately 6 times more energy efficient than FlexiTP, but the results would vary for other values of f_s. If we reduce f_s, the advantage of TBSP over FlexiTP grows, and if we increase f_s over 1/3, FlexiTP becomes more efficient than TBSP. However, in environments with a large f_s, both TBSP and FlexiTP are outperformed by contention-based MAC protocols and TRAMA [8].

4.3 Probability that a node is assigned an unfeasible slot

Figure 3 shows the probability p_u that a node is assigned an unfeasible slot. For FlexiTP, this probability decreases with the node density because a greater node density increases the number of conditions that are required for a pair of nodes to be assigned the same transmission slot. Figure 3 presents two versions of FlexiTP. The first one is the original version, which neglects the interference from a node if it lies at least $k = 2$ hops away, and the second one is a modified version that uses $k = 3$. The modified version consumes twice as much energy as the original version because it requires that each node reports its schedule to a greater number of neighbors. Although this modified version of FlexiTP reduces p_u, the obtained p_u is non-negligible. In FlexiTP, when a node obtains an unfeasible slot, it unduly concludes that its parent node is unreachable since according to its schedule information there is no interference in that slot. Therefore, the node seeks a new parent, but in sparse networks, the node may not have any other neighbor, in which case the node cannot reach the data sink. In contrast, TBSP guarantees $p_u = 0$ because it only assigns a slot to a node if the node's ability to communicate during that slot has been proved with a Testing Packet.

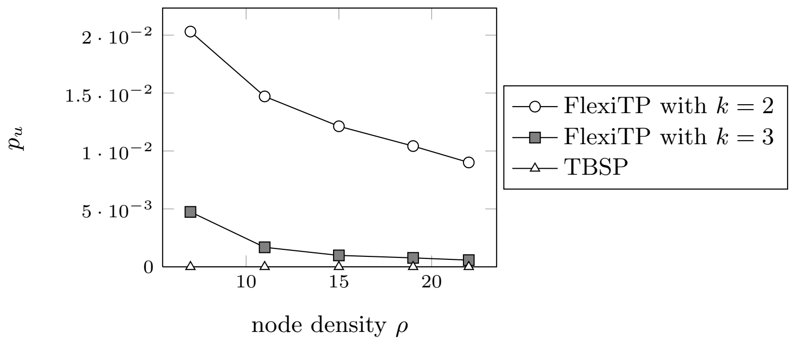

Fig. 3. Probability p_u that a node is allocated an unfeasible slot.

4.4 Other properties of the computed schedule

Our DS-selection algorithm seeks to obtain a schedule with the following properties. First, the schedule is short, which means that many nodes share each DS. A short schedule requires a smaller number M of DSs per TF (shown in Figure 1), thereby decreasing the frame rate and increasing the throughput. Second, the schedule enjoys a small traversal time (i.e. the packets from the data sources reach the data sink within a small number of TDMA frames). Third, the schedule imposes buffering requirements that do not grow with the network size. FlexiTP also seeks a schedule with these properties. Our simulation results show that TBSP performs similarly in these three aspects but we do not show these results due to space limitations.

5 Conclusions

The energy consumption of different scheduling protocols varies widely with the maximum tolerable value of the adaptation latency, which is approximately the average number of TDMA frames until a node obtains a transmission slot. If the network changes slowly and an adaptation latency of 15 frames is tolerable, for our simulation parameters, TBSP consumes six times less energy than FlexiTP. TBSP is also likely to outperform the other existing distributed TDMA protocols because it spares the sensor nodes from the burden of keeping track of their neighbors' schedules. Additionally, since TBSP does not use an interference model, the sensor nodes are less likely to incorrectly assume that their parent nodes are unreachable, and thus they are also less likely to become disconnected.

References

1. Arampatzis, T., Lygeros, J., Manesis, S.: A survey of applications of wireless sensors and wireless sensor networks. In: Proc. IEEE Mediterranean Conf. Control and Automation. pp. 719–724 (2005)
2. Díaz-Anadón, M.O.: Randsched implementation. http://github.com/ornediaz/wsnpy.git (2009)
3. Gandham, S., Ying, Z., Qingfeng, H.: Distributed minimal time convergecast scheduling in wireless sensor networks. In: Proc. IEEE Int'l Conf. Distributed Computing Systems. pp. 50–50 (2006)
4. Langendoen, K.G.: Medium access control in wireless sensor networks. In: Wu, H., Pan, Y. (eds.) Medium Access Control in Wireless Networks, Volume II: Practice and Standards, pp. 535–560. Nova Science Publishers, Hauppage, New York (May 2008)
5. Lee, W.L., Datta, A., Cardell-Oliver, R.: FlexiTP: A flexible-schedule-based TDMA protocol for fault-tolerant and energy-efficient wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. 19(6), 851–864 (Jun 2008)
6. Maróti, M., Kusy, B., Simon, G., Lédeczi, Á.: Robust multi-hop time synchronization in wireless sensor networks. In: Proc. Int'l Conf. Wireless Networks (ICWN) (Jun 2004)
7. Rajendran, V., García-Luna-Aceves, J., Obraczka, K.: Energy-efficient, application-aware medium access for sensor networks. In: Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems Conf. (MASS). pp. 630–637 (Nov 2005)
8. Rajendran, V., Obraczka, K., García-Luna-Aceves, J.: Energy-efficient, collision-free medium access control for wireless sensor networks. Springer J. Wireless Networks 12(1), 63–78 (Feb 2006)