TDMA Scheduling for Event-Triggered Data Aggregation in Irregular Wireless Sensor Networks

Mario Orne Díaz-Anadón
Kin K. Leung
Electrical & Electronic Engineering Department, Imperial College, London, UK
ornediaz@gmail.com, kin.leung@imperial.ac.uk

28 May 2011
The final version of this paper is published in Elsevier Computer Communications Journal, 2011,
http://dx.doi.org/10.1016/j.comcom.2011.05.005

This work is funded by UK EPSRC Research Grant EP/D076838/1, entitled: "Smart Infrastructure: Wireless Sensor Network System for Condition Assessment and Monitoring of Infrastructure".

Multi-hop wireless sensor networks often use a TDMA schedule to collect data periodically from multiple locations within a large area. If the measurements from neighboring sensors are cross-correlated, they can be aggregated and compressed as they travel to the data sink. In order for data aggregation to occur quickly, the TDMA schedule must arrange time slot assignments in a certain order. The existing scheduling protocols cannot quickly obtain a schedule with this order, high concurrency, and no collisions. We propose a distributed TDMA scheduling protocol for data aggregation called DATP. In DATP, the sensor nodes transmit dummy packets in order to determine whether they can tolerate the interference from the other nodes that are assigned the same time slot. In this way, time slot allocations are empirically verified to be collision-free. In contrast, the existing protocols obtain schedules with collisions because they use unrealistic interference models such as neglecting interference generated more than two hops away. Furthermore, our simulations reveal that DATP achieves similar concurrency and lower execution time than comparable protocols. These simulations are executed for different network sizes, node densities, and data compression models. In addition, we show that, in networks with fluctuating links, DATP's main advantage is its execution speed.

1 Introduction

Wireless Sensor Networks (WSNs) are used to monitor a variety of natural and man-made environments [1]. They consist of devices called *sensor nodes* that communicate with each other through radio links. The sensor nodes collect measurements from their environment and transmit them to a special node called *data sink*, possibly across multiple hops.

The reliability of the network can be improved by deploying dense networks in which neighboring nodes generate correlated data. Such data can be compressed locally before being relayed to the data sink, which is a process referred to as *in-network data aggregation* [2]. This process reduces the amount of wireless communication and thus the energy consumption [3].

A network is said to be *event-triggered* if no data needs to be transmitted to the data sink until a relevant, unpredicted event occurs. This paper considers the problem of quickly and reliably obtaining a TDMA schedule for data aggregation in event-triggered applications. TDMA avoids packet collisions and overhearing, but introduces a scheduling overhead that we seek to minimize.

Many distributed TDMA protocols [4, 5, 6] are unsuitable for event-triggered data aggregation because they do not arrange transmissions so as to minimize the latency towards the data sink. Other protocols [7, 8] provide low latency but are centralized and scale poorly with the network size. Furthermore, all the existing protocols use interference models that can fail due to the unpredictability of wireless channels [9]. As a result, the existing protocols may assign time slots with excessive interference.

This paper proposes the Data Aggregation TDMA Protocol (DATP) in order to overcome the limitations of the existing protocols. In DATP, the sensor nodes contend for time slots without knowing their neighbors' schedules. A node only obtains a time slot if it has proved empirically its tolerance to the interference from all the current users of that time slot. This approach incurs little overhead and makes no assumptions about a node's interference range.

Our simulations reveal that DATP obtains a low-latency schedule with a high degree of concurrency. The schedule is very likely to be collision free because every time slot allocation is tested before being confirmed. DATP is scalable and fast because it operates distributedly and spares the sensor nodes from the burden of discovering the identity and schedule of their neighbors. Furthermore, DATP can operate in a network with fluctuating links, in which case its main advantage is its execution speed.

The rest of the paper is organized as follows. Section 2 presents our system model and formulates our scheduling problem. Section 3 describes the limitations of existing protocols. Section 4 presents the basic version of DATP. Section 5 proves theoretically that DATP always obtains a collision-free schedule in reliable networks. Section 6 extends DATP to make it suitable for unreliable networks. Section 7 presents simulation results for both reliable and unreliable networks. Finally, Section 8 draws some conclusions.

2 System model and problem formulation

2.1 Overview of the network phases

We consider a multi-hop network with a single a data sink. The network spends most of the time in a *quiet phase*. When an event occurs, a *TDMA scheduling phase* is executed, followed

by a *data transmission phase*. After all the data related to the event is transmitted, the network switches back to the quiet phase and remains there until the next event.

2.2 Quiet phase

We assume that the duration of the quiet phase is in the order of hours or days. During this time, nodes may be added or removed, and the propagation environment may change. The goals of the quiet phase are to consume little power, maintain a routing tree and time synchronization, and be ready to respond to events quickly.

An efficient *time synchronization* method is to propagate synchronization information from the data sink to every sensor node using a *routing tree* rooted in the data sink. Every node receives synchronization information from its parent node and relays that information to its children nodes [10, 4]. If a node repeatedly fails to receive synchronization information from its parent node, it finds a new parent node. However, in order to save energy, a node does not keep track of its other neighbors.

2.3 Number of time slots assigned to each node

We assume that, during the data transmission phase, time is divided into *reporting intervals* with duration equal to the interval between the start of consecutive TDMA frames. Every sensor node generates exactly one packet about every reporting interval. Packets about different reporting intervals cannot be aggregated, but packets from different nodes about the same reporting interval can be aggregated and compressed before being relayed. The number of time slots that have to be assigned to a node depends on its number of descendants, the time slot duration, and the amount of data compression.

The amount of data compression is specified by a compression model. The compression models in [11, 12] are expressed in terms of bits rather than packets, and thus require at least two parameters: the packet size and a compression coefficient. In order to have a model with a single parameter, we use the following compression model. Every node X has to transmit

$$n_X = \left\lceil \frac{1 + d_X}{1 + \gamma} \right\rceil \tag{1}$$

packets per reporting interval, and thus also per TDMA frame. Here, d_X is the number of descendants of X in the tree, and γ is a *compression coefficient* that takes values between zero, which indicates no compression, and infinity, which indicates that any number of packets from the same reporting interval can be compressed into a single packet.

The maximum and the histogram are examples of practical aggregation functions that verify $\gamma \to \infty$ [13]. The maximum verifies $\gamma \to \infty$ because the maximum of any number of measurements is simply a number. The histogram verifies $\gamma \to \infty$ because the histogram of any number of measurements can be stored with a fixed number of bits. Therefore, the information that needs to be transmitted to the data sink when using these functions is independent of the network size.

Sometimes the compression coefficient γ is lower than ∞, for example if location information is required. A method to provide coarse location information is to divide the monitored area into different regions and deliver summaries from each of these regions to the data sink. Measurements from the same region are aggregated, but not from different regions. Therefore, the amount of information that the data sink needs to receive grows with the number of regions, and thus with the network size.

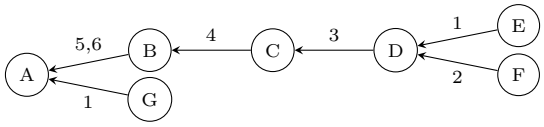


Figure 1: A routing tree and a schedule for data aggregation verifying the precedence requirement.

2.4 Precedence property

During the data transmission phase, we define *latency* as the time elapsed from the instant when the data are generated until the instant when they reach the data sink. The latency can be roughly expressed as a number n_f of TDMA frames. This number depends on the relationship between the indices of the time slots assigned to a node and its children.

In order to minimize the latency, we impose the *precedence property*, which we define as follows: the indices of the time slots assigned to a node should be bigger than the indices of the time slots assigned to its children nodes. This property ensures $n_f = 1$ because it allows every node to have all the data it needs before its transmission time slot.

Figure 1 shows a schedule verifying the precedence property for a routing tree rooted at node A. The schedule is obtained for a compression coefficient of $\gamma = 3$, which means that only B needs more than one time slot because only B has more than 3 descendants. According to this schedule, nodes E and G transmit in the first time slot. Node F transmits in the second time slot of the same frame. Node D transmits in the the third time slot of the same frame because it has already received packets from its children E and F. This process continues and at the end of the TDMA frame the data sink has received all the packets, and thus $n_f = 1$.

An example of a schedule that does not satisfy the precedence property is the schedule of Figure 1 modified by assigning slot 5 instead of slot 1 to E. With this schedule, node D is unable to transmit in the first TDMA frame because it has not received the packet from its child E yet, and thus $n_f = 2$.

2.5 Problem formulation

This paper addresses the problem of designing a protocol to obtain a TDMA schedule during the scheduling phase. The protocol should be fast in large and dense networks, and obtain a schedule with the following properties. First, every node should be allocated the number of slots indicated by (1). Second, the schedule should verify the precedence property in order to obtain a low latency. Third, the schedule should enjoy a low *failure probability* p_f, defined as the probability that a transmission during the data transmission fails due to excessive interference. Fourth, the schedule should enjoy a high *concurrency*, defined as the quotient between the number of transmissions per TDMA frame and the number of time slots.

3 Related work

For short events, the quality of the routing tree is unimportant because it is hardly used. As the event duration grows, the quality of the routing tree becomes more important. Therefore, for long events, a new, event-specific routing tree should be constructed after each event. This tree construction approach is followed by the algorithms in [14, 15, 2]. However, these algorithms are slow because they are centralized and ignore their interaction with the MAC layer [11].

The FAT protocol [11] is a cross-layer protocol that handles the MAC layer during the quiet phase and performs the tree

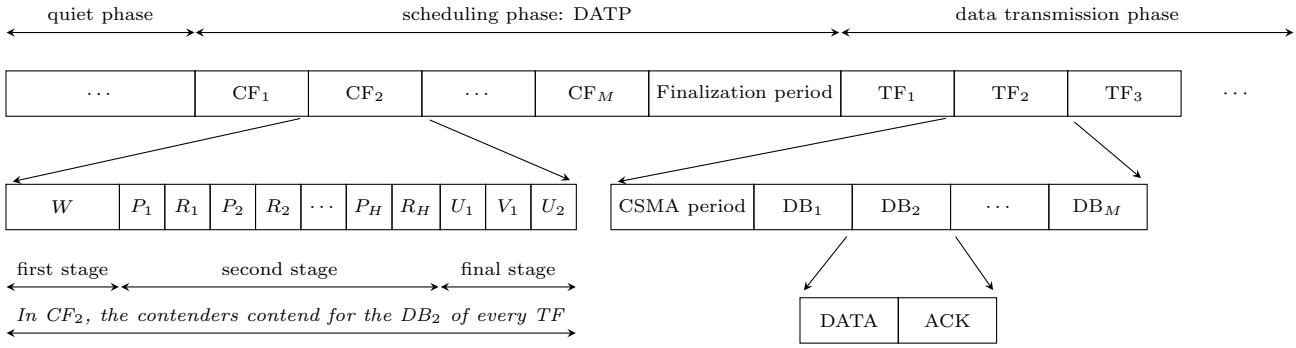


Figure 2: Timing diagram of the quiet phase, DATP and the data transmission phase.

construction after the occurrence of every event. By arranging the active periods of the sensor nodes according to the expected data pattern, FAT responds quickly to events. FAT is not directly applicable to our problem formulation because it requires keeping this information updated during the quiet phase, which consumes significant energy in time-varying networks. However, FAT can be adapted to our problem formulation by making every node claim the same parent as it had during the quiet phase.

During the data transmission phase, contention-based protocols are an alternative to schedule-based protocols [16, 17, 3]. Contention based protocols incur no scheduling overhead, but require contention before every packet transmission, and suffer frequent packet losses under high traffic loads [3, 18, 19]. Therefore, contention-based protocols are suitable for unpredictable traffic, whereas schedule-based protocols are suitable for long-running, periodic traffic. Hybrid protocols with features from the two types of protocols have been developed [20, 21, 22]. However, they protocols often perform significantly worse than schedule-based protocols for periodic traffic [4]. A key goal of our paper is to provide a schedule-based protocol with lower overhead than the existing schedule-based protocols.

A schedule-based protocol is said to be *adaptive* if it can modify the schedule during the data transmission phase. This ability enables nodes that were assigned time slots with excessive interference during the scheduling phase to replace their time slots during the data transmission phase. However, the existing adaptive schedule-based protocols [23, 4, 6, 5, 24] do not preserve the precedence property, and doing so is is difficult. To illustrate this difficulty, suppose that in Figure 1 E must be assigned a new time slot because it suffers excessive interference from G in time slot 1. Node E cannot be assigned a new time slot without changing the time slot of D because the index of the time slot assigned to E must be smaller than that assigned to D, which is 3. A solution to this problem is to change the algorithm used to obtain the initial schedule so that some unused time slots are left between those of a node and its children, but this solution deteriorates the concurrency of the schedule.

Due to the difficulty of developing an adaptive scheduling protocol that preserves the precedence property, we focus our attention in obtaining a correct schedule in the scheduling phase, i.e. a schedule with a low failure probability p_f. Obtaining low p_f and high concurrency c simultaneously requires a realistic interference model such as the *physical interference model*. This model uses the SINR to determine the success of a transmission, and is used in [25, 26]. An enhancement of this model that considers Rayleigh fading is used by the scheduling algorithm in [27]. However, the physical interference model incurs significant overhead in collecting link quality information about every wireless link and transmitting this information to the data sink.

A much more common interference model is the *k-hop interference model*, which consists in neglecting interference sources more than k hops away from a receiver. Obtaining the shortest possible schedule with this model is an NP-complete problem [8], so approximations are used instead.

The distributed TDMA protocols that use the k-hop interference model [23, 4, 6, 5] do not verify the precedence property. The EMAC protocol [28] is distributed and verifies the precedence property. However, instead of using the k-hop interference model, it neglects interferers more than 2 hops away in the routing tree. This interference model is very unrealistic because it may ignore the interference from a node within transmission range.

The algorithms in [7, 8] are very similar and we refer to both of them as BF_k because they traverse the routing tree in Breadth-First (BF) order and use the k-hop interference model. These protocols are centralized and assign time slots as they traverse the tree. We simulate BF_k and DATP in Section 7.

BF starts its execution after an event is detected and consists of four phases. First, the sensor nodes discover their neighbors, which is necessary because they do not keep track of them during the quiet phase. Second, the sensor nodes transmit their neighbor lists to the data sink. Third, the data sink uses these lists to compute the schedule. Fourth, the data sink informs the sensor nodes of their time slots.

BF incurs significant overhead because it is centralized and the nodes close to the data sink have to relay a large data volume. The data volume transmitted in the second phase is larger than that transmitted in the fourth phase, at least when the compression coefficient γ is large.

In conclusion, the existing scheduling algorithms for data aggregation do not obtain schedules with high concurrency and low failure probability in a scalable and efficient way. We solve these problems by proposing the Data Aggregation TDMA Protocol (DATP).

4 DATP

DATP is a protocol to obtain a TDMA schedule for data aggregation. It is fast and scalable because it operates distributedly and spares the sensor nodes from the burden of tracking their neighbors' schedules. In order to obtain a certain time slot, the sensor nodes transmit dummy packets at the same time as the current users of that time slot. If they receive an ACK in response, they have empirically proved their tolerance to the interference of those users. This test-based approach to scheduling is more reliable than that of the existing protocols, which uses an interference model.

4.1 Execution context

The timing diagram in Figure 2 shows the three phases of the network: a quiet phase in which time synchronization and a routing tree are maintained; an event-triggered scheduling phase in which DATP is executed; and a data transmission phase in which the event is reported.

The data transmission phase consists of transmission frames (TFs) labeled $\{TF_1, TF_2, \ldots\}$. There are as many TFs as necessary to report the event to the data sink. Each TF consists of a CSMA period and M data blocks (DBs). The CSMA period is used for unpredicted packet transmissions. The DBs are labeled $\{DB_1, \ldots, DB_M\}$ and are used for periodic packet transmissions. Each DB consists of a DATA slot that is used for a packet transmission, and an ACK slot that is used for the corresponding acknowledgment.

4.2 Phases of DATP

Figure 2 shows that DATP divides the scheduling phase into M contention frames (CFs), labeled $\{CF_1, CF_2, \ldots, CF_M\}$, and a finalization period. Each CF decides the set of nodes that transmit in a certain of DB. There are as many CFs as necessary until all the nodes obtain the DBs they need. Therefore, the number M of CFs is equal to the number of DBs per TF in the data transmission phase, which is unknown in advance.

4.3 Contention during a CF

The nodes that attempt to obtain DB_i in CF_i are referred to as the *contenders* of DB_i, and successful contenders are referred to as *winners*. Every sensor node becomes a contender after all its children have obtained all the DBs they need. By waiting for its children, the nodes ensure the satisfaction of the precedence property.

In order to obtain a schedule with high concurrency c, a large number of contenders should become winners in every CF. Furthermore, in order to obtain a low failure probability p_f, the winners should be mutually compatible. A node X is said to be *compatible* with another node Z if two conditions hold. First, X's parent receives X's packet successfully when X and Z transmit simultaneously. Second, X receives its parent's packet when X's parent and Z's parent transmit simultaneously. Note that X being compatible with Z does not imply the inverse.

The contention process during a CF consists of three stages. If a contender fails at any of these stages, it withdraws from the contention immediately and contends again in the next CF. The contenders that succeed in the three stages become winners and report their success to their parents, but not to any other nodes.

4.3.1 First stage of a CF

The *first stage of the contention* of a CF occurs during the W slot. Every contender backs off for a random period within this slot and then performs a clear channel assessment (CCA). If the wireless channel is clear, the contender has succeeded in this stage and transmits dummy data until the end of the W slot in order to thwart nearby contenders that might be incompatible with itself. If, on the other hand, the channel is busy, the contender has failed in this stage, withdraws from the contention until the next CF, and remains active during the rest of CF in case any of its children are contending to obtain a DB.

The nodes that are not contending for a DB in the current CF check the channel status at the end of the W slot. If the channel is busy, they listen during the rest of the CF in case their children are seeking to obtain a DB, otherwise they sleep until the next CF.

The purpose of this stage is to reduce the number of contenders. In particular, at the end of this stage, the probability that two contenders within transmission range of each other remain in the contention is very small. Even if no such contenders remain, the remaining contenders may be mutually incompatible because they measured the interference at the transmitters rather than at the receivers. This problem is called the *hidden terminal problem* and is addressed in the second and third stages of the CF.

4.3.2 Second stage of a CF

The *second stage of the contention* of a CF is divided into the slots $\{P_1, R_1, P_2, R_2, \ldots, P_H, R_H\}$. Every remaining contender X chooses a random integer h between 1 and H. To maximize the concurrency of the schedule, the number H of slot pairs should be so large that every contender chooses a different h.

In slot P_h, X transmits a packet indicating its identity to its parent node Y. If Y receives this packet, it replies with an acknowledgment in slot R_h and transmits dummy packets in slots $\{R_{h+1}, R_{h+1+2}, \ldots, R_H\}$ in order to interfere with contenders that might be incompatible with X. If X does not receive Y's acknowledgment in slot R_h, X has failed in the second stage of the contention. Otherwise, X has succeeded and transmits dummy packets in the slots $\{P_{h+1}, P_{h+1+2}, \ldots, P_H\}$ in order to provoke the failure of contenders that may be incompatible with itself.

Our simulations in Section 7.1.3 show that $H = 4$ provides a good tradeoff between schedule concurrency and scheduling speed. A higher H would be needed if the first stage of the CF were suppressed. The purpose of the first stage is to reduce the overhead in this way.

A contender that chooses a small h is very likely to succeed in the second stage, whereas a contender that chooses a big h is unlikely to succeed in the second stage. A contender that chooses a large h only succeeds if it is compatible with the nodes that succeeded earlier. However, the success of a node with large h may create intolerable interference for some of the nodes that succeeded earlier. Such nodes withdraw from the contention during the third stage.

4.3.3 Third stage of a CF

The *third stage of the contention* of a CF consists of slots $\{U_1, V_1, U_2\}$, as shown in Figure 2. In slot U_1, every remaining contender X transmits a packet indicating its identity to its parent node Y. If Y receives this packet, it replies with an ACK packet in slot V_1. If X does not receives this ACK, X has failed in the last stage of the contention. Otherwise, X has succeeded and reports its victory to Y in slot U_2.

4.4 Finalization period

The finalization period begins when the data sink has received victory declarations from all its children nodes. The purpose of the finalization period is to distribute a packet called *finalization packet* from the data sink to every node in the network. This packet is created by the data sink, and it contains two parameters that are decided by the data sink: the start of the finalization period and the TDMA frame period. The data sink transmits the finalization packet to its children nodes, and each of them relays this packet to its own children. Every node transmits the finalization packet using CSMA, and it retransmits the packet until it receives an ACK. In this way, the finalization packet is propagated from parent to child along the routing tree.

5 Failure probability in reliable networks

This section proves some results about DATP under the following assumptions: wireless links are constant, there is no external inference, and there are no node failures. These assumptions are abandoned in Sections 6 and 7. We use the definition of compatibility from Section 4.3. We define a *collision-free schedule* as a schedule where the transmitters in every DB are mutually compatible.

Lemma 1. *If DATP finds a schedule, it is collision free*

Proof. DATP only schedules simultaneously the winners of the same CF, and thus we have to prove that the winners of every CF are mutually compatible. Observe that the winners were able to communicate with their parents in slots U_1 and V_2. Since these transmissions succeeded concurrently under the interference from unsuccessful contenders, the winners can also communicate successfully without the unsuccessful contenders' interference, and thus are mutually compatible. □

Theorem 2. *If $L \geq 2$ and a collision-free schedule exists, DATP always obtains a collision-free schedule.*

Proof. Because winners are mutually compatible, victory declarations in slot U_2 are always received correctly, and no node waits unnecessarily for its children nodes. Since we also have $L \geq 2$, there is a nonzero probability that exactly one contender X will transmit a REQ packet in slot P_1. When this happens, X receives an ACK because there are no other interferers and a schedule exists. Then, X transmits dummy packets in slots P_2 to P_L, and reaches the third stage. If in slot P_2 a contender Y transmits a packet, Y only reaches the third stage if it is compatible with X and other contenders in slot P_2. It is possible that Y is compatible with X but not the other way around. In this case, although both X and Y reach the third stage, only Y has a chance of becoming a winner. Therefore, at least one node still has a chance of becoming a winner. Repeating this argument in every P_i slot for $i \in \{3, \dots L\}$, we obtain that at least one node becomes a winner under our assumption that X is the only node to transmit in P_1. Therefore, the probability of there being some winners in every CF is strictly positive. Since there are as many CFs as necessary, a schedule is found eventually, and according to the previous lemma, it is collision free. □

Note that the existing TDMA protocols do not verify the above properties under the same assumptions because they rely on interference models that can fail.

6 Extensions for unreliable networks

6.1 Extension for networks with fluctuating links

We say that a wireless link is *fluctuating* if its attenuation varies quickly around a certain average. If link fluctuations are sufficiently large, no schedule can guarantee zero failure probability p_f, even if no time slots are shared. However, the schedule-based protocols are still useful because they enjoy lower p_f than the contention-based protocols.

In a network with fluctuating links, DATP obtains few winners in every CF. This is because winning in a CF requires five successful consecutive transmissions and the success probability of each transmission decreases with the intensity of the fluctuations. A small number of winners leads to a schedule with poor concurrency. In order to improve the concurrency of the schedule in networks with fluctuating links, we extend DATP as follows.

Every time slot in Figure 2 is replaced by a *superslot* consisting of A time slots, where A is a small network-wide parameter. Every node that in the original version of DATP had to transmit one packet in one slot, in the modified version transmits one packet in each slot of the associated superslot. A packet reception during a superslot is considered successful if the packet was received correctly in at least S slots of the superslot, where S is a small network-wide parameter that verifies $S \leq A$. We investigate the choice of A and S in Section 7.3.

6.2 Extension for handling node failures

A failed node may cause its parent node to wait indefinitely. In order to avoid this problem, a node only waits for a maximum time for its children nodes. The maximum waiting time should be a function of the node failure probability and the number of descendants. We do not implement this extension because we do not simulate node failures.

7 Performance evaluation

We write a packet-level simulator in Python whose code is available in [29]. The simulator implements DATP and BF_k for different values k, where k indicates the minimum hop distance imposed by BF_k between a receiver and all its interferers. The simulator compares DATP and BF_k in terms of three metrics. The first metric is the concurrency c of the computed schedule, which is average number of transmissions per time slot; it is equal to one if every node is assigned a different DB and all the DBs are used. The second metric is the transmission failure probability p_f, which indicates the fraction of unsuccessful packet transmissions in the data transmission phase when using the schedule computed in the scheduling phase. The third metric is the execution time of the scheduling phase.

7.1 Simulator details

7.1.1 Communication and channel model

Every node's transmission power is 10^{-6} W and the noise figure of every receiver is 7 dB. A packet transmission is successful if the SINR at the receiver exceeds 20 dB. The computation of the SINR considers the interference from all the nodes in the network.

The wireless channel has an attenuation exponent of 3.5 and a path loss at 100 m of 80 dB. The wireless links suffer log-normal shadow fading with a standard deviation of 8 dB. We define t as the maximum distance across which two nodes can communicate if there is no fading and interference; the parameters presented above yield $t = 48$ m.

Section 7.2 simulates constant links, whereas Section 7.3 simulates fluctuating links. Link fluctuations are simulated by multiplying the attenuation of every link, including interference links, by a different random number in each time slot. The random number is taken from a log-normal distribution with standard deviation σ_f.

7.1.2 Topology

All the sensor nodes are data sources and are deployed randomly within a square with side $\bar{x}t$, where $\bar{x}$ is referred to as the *normalized network size*. The data sink is in the middle of the left border of the square. Although we also simulated networks where the data sink lies in the middle of the square, we omit the corresponding results because they resemble those obtained in smaller networks with the data sink in the middle of the left side of the square.

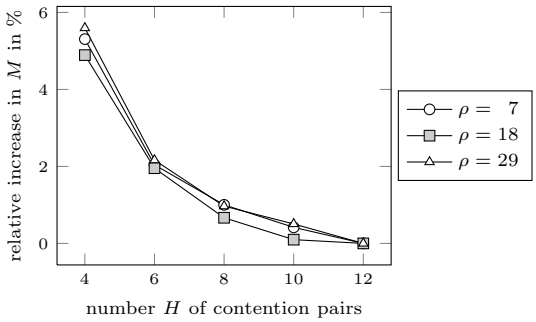


Figure 3: Length M of the computed schedule as a function of H and ρ. The y-axis shows $(M - M_0)/M_0$, where M_0 is the value of M obtained for $H = 12$.

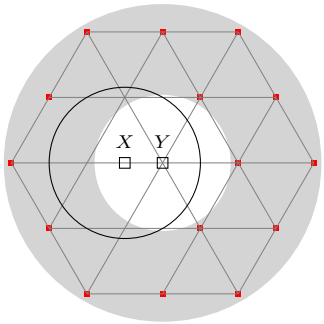


Figure 4: Simplified model to justify that the number of hidden terminals does not increase indefinitely with the node density ρ.

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. The node density represents the average number of nodes per neighbor if there were no shadow fading. Our default parameters are $\bar{x} = 8$ and $\rho = 24$, which yield a total of $N = 484$ nodes. The routing tree is computed with Dijkstra's algorithm using the hop distance as the cost metric. We discard every random deployment where over 10 % of the nodes cannot reach the gateway directly or through multiple hops, which is rare for $\rho \geq 7$. All the results are the average of 80 simulation runs.

7.1.3 Choice of H in DATP

The number H of slot pairs in the second stage of a CF is an important parameter whose meaning is shown Figure 2. This parameter has a complex influence on the execution speed. On one hand, increasing H prolongs the CFs, which contributes to deteriorating the execution speed. On the other hand, increasing H reduces the number M of CFs, which contributes to improving the execution speed.

In order illustrate why M decreases with H, suppose that only two mutually incompatible contenders, which we denote by X and Z, reach the second stage of a CF. If X and Z choose the same h, they transmit simultaneously in P_h, suffer each other's interference, and fail to receive a reply. As a result, none of them becomes a winner. In contrast, if X chooses a smaller h than Z does, X suffers no interference in its first transmission of the second phase and reaches the third stage. When Z transmits its packet, it receives no reply due to the interference from X or X's parent, and withdraws from the contention. When X transmits in the third stage, it suffers no interference and becomes a winner. Therefore, the number of winners is highest when X and Z choose different h. Consequently, as we wanted to show, M decreases with H.

However, the decrease of M with H is bounded. After M is sufficiently large, all the contenders are likely to choose different h, so further increasing H barely reduces M. This can be seen in Figure 3, which shows that increasing H from $H = 4$ to $H = 12$ only reduces the schedule length by 5.5 % independently of the node density ρ. Since $H = 4$ obtains good execution speed (H and M are small) and concurrency (M is small), we use it in the rest of the simulations.

We illustrate the independence of M with H with an example based on Figure 4. For simplicity, we assume a constant transmission range r_t, defined as the maximum distance within which two nodes can communicate with each other or detect each other's transmissions. We also assume a constant interference range $r_i = 2r_t$, defined as the maximum distance over which a transmitter's interference is significant.

In Figure 4, node X is contending for a DB to communicate with its parent Y. Node Y is the center of a ring with internal radius r_t and external radius $r_i = 2r_t$. Therefore, the nodes within the white area around Y can communicate with Y, and the nodes within the shaded area can interfere with Y, but not communicate with Y. Node X is the center of a circle with radius r_t.

Suppose that X reaches the second stage. We say that a node is an *incompatible contender* of X if it reaches the second stage and generates significant interference upon Y. The incompatible contenders of X are inside the circle of radius $2r_t$ centered at Y and outside the circle of radius r_t centered at X. Since this area is finite and the minimum distance between any two incompatible contenders is r_t, the maximum number of incompatible contenders is bounded. Therefore, H does not need to increase with ρ.

An example of a distribution of incompatible contenders of X that contains almost as many contenders as possible is shown in Figure 4. Contenders are represented by small squares and arranged in an hexagonal grid. Every contender has six neighboring contenders at a distance r_t.

7.1.4 Estimation of BF's execution time

The implementation details of the first, second, and fourth phases of BF are omitted in [7, 8], whose focus is the third phase. Designing an efficient implementation of the first phase is nontrivial because the back-off periods that precede every packet transmission must be chosen carefully. Making our own design would affect the validity of our conclusions because our design may be suboptimal. Instead, we derive an optimistic estimate of the execution time of the first phase and assume that this phase provides perfect information to the following phases. This section contains our derivation of the execution time of the first phase and describes our implementation of the second and fourth phases. We assume that the third phase is infinitely fast.

Here, but not in our simulations, we assume the existence of a constant transmission range r_t, which is equal to t, and a constant interference range r_i, which is equal to $2r_t$. BF's first phase requires that every sensor node announces itself to all its one-hop neighbors by transmitting a packet that we call *identification packet*. For efficiency's sake, we assume that each node only transmits one identification packet and that this packet is received by all its neighbors. In order for all the receivers to receive this packet, no transmissions should occur within their interference range. Therefore, the minimum distance between concurrent transmitters is $r_t + r_i = 3r_t$.

The maximum density of concurrent transmitters is achieved when the transmitters are deployed over an hexagonal grid as the one in Figure 4, with the difference that now the distance

between transmitters is $3r_t$. This grid divides the plane in non-overlapping hexagons with side $1.5r_t$ and a transmitter in the center of the hexagon. Every hexagon has an area of $5.84r_t^2$ and contains 5.84ρ nodes. Since only one in every 5.84ρ nodes transmits an identification packet at a time, the neighbor discovery phase lasts at least 5.84ρ time slots. This estimate is optimistic because in practice there are collisions and unused intervals during the first phase. In our simulations, the time slots used in both BF's neighbor discovery phase and DATP last for 20 ms.

BF's second phase serves to transmit neighbor information, and BF's fourth phase serves to transmit schedule information. The neighbor information consists of lists of node IDs, each of which occupies 16 bits. The schedule information contains time slot indices, each of which also occupies 16 bits. The packet size is 448 bits. Therefore, if every node has g neighbors, a node with d descendants in the routing tree has to transmit $\lceil 16(g+1)(d+1)/448 \rceil$ packets with neighbor information in the second phase. In addition, in the fourth phase, this node has to receive $\lceil 16 \cdot 2(d+1)/448 \rceil$ packets with schedule information if the compression coefficient γ is infinity.

The second and fourth phases are divided into pairs of time slots that are used to transmit neighbor and schedule information. The first slot in each slot pair has a duration of 30 ms, and is used to contend briefly and transmit a 448-bit data packet. This data packet can be used to transmit neighbor or schedule information. The second slot in each pair has a duration of 15 ms and is used to contend briefly and to transmit an acknowledgment packet in response to the preceding packet.

Every node X seeking to transmit a packet to a node Y keeps track of the number n_f of consecutive times that it failed to receive an acknowledgment in response to a packet. During the first slot of a slot pair, X contends with probability 2^{-z}, where z is the minimum of n_f and a network-wide constant that we denote by n_m. The contention process consists in backing off for a random time and transmitting a packet if at the end of the back-off period the channel is idle. If Y receives X's packet, it backs off for random period, and only replies with an acknowledgment if at the end of the back-off period the channel is idle.

The value of the constant n_m that minimizes the execution time of BF_k decreases as the standard deviation σ_f of the channel fluctuations increases, which can be explained as follows. For low σ_f, the most likely cause for not receiving acknowledgments repeatedly is the interference from hidden terminals, which can be reduced by increasing n_m. In contrast, for a high σ_f, most unsuccessful transmissions are caused by link fluctuations, and thus reducing n_m increases the number of transmission attempts and the number of successes. Our simulations reveal that the optimal values of n_m are the following: 4 for $\sigma_f = 0\,\mathrm{dB}$; 3 for $\sigma_f = 0.5\,\mathrm{dB}$; and 2 for $\sigma_f = 1\,\mathrm{dB}$. We use these optimal values in our simulations.

7.2 Evaluation in networks with stable links

Figure 5 is obtained for a compression coefficient $\gamma = 10^4$ and is the result of averaging 100 simulations. The figure consists of a matrix of graphs. Each matrix row shows one of the three performance metrics, and each matrix column refers to a different node density ρ.

The first row of Figure 5 shows that the failure probability p_f of BF p_f increases with the normalized network size $\bar{x}$. This is because in small networks most nodes are near the border of the monitored area and thus are surrounded by few neighbors and few interferers. The first row of Figure 5 also shows that the failure probability p_f decreases with the node density ρ. One reason for this is that, as ρ increases, the number of hops between nodes becomes more correlated with the physical distance between them. Another reason is that, as ρ increases, the number of conditions used by BF to determine whether two concurrent transmissions interfere with each other increases, which reduces the number of concurrent transmissions and the failure probability p_f.

The first row of Figure 5 also highlights the high failure probability p_f suffered by BF_2 and BF_3, particularly in networks that are large and sparse. For $\rho = 9.6$ and $\bar{x} = 8$, p_f is 0.35 in BF_2, which is clearly intolerable. BF_3 performs much better, obtaining $p_f = 0.06$ for the same parameters. However, even $p_f = 0.06$ may be too high, particularly in networks with many hops. For example, if a node lies 10 hops away from the data sink, its packets reach the data sink with a probability as small as $(1 - p_f)^{10} = 0.53$. In contrast, DATP achieves $p_f = 0$ because it only assigns the same DBs to nodes that have been proved mutually compatible empirically. However, DATP cannot guarantee $p_f = 0$ for fluctuating links, as discussed in Section 7.3.

The second row of Figure 5 shows that DATP achieves as much concurrency as BF_2 and much more than BF_3. In small networks, the highest possible concurrency is 1 because no two nodes are far enough from each other to share a time slot; in larger networks, higher concurrency is possible. BF_3 obtains a much lower concurrency than BF_2 does because it imposes a larger distance between concurrent transmitters.

The third row of Figure 5 shows that DATP is much faster than BF_k, particularly for a large node density ρ and a large network size $\bar{x}$. This is because DATP is distributed and BF_k is centralized. The execution time of BF_k depends on the data volume to be transmitted in the second phase. This data volume increases with ρ and $\bar{x}$, but is independent of k. Therefore, the execution time is also independent of k.

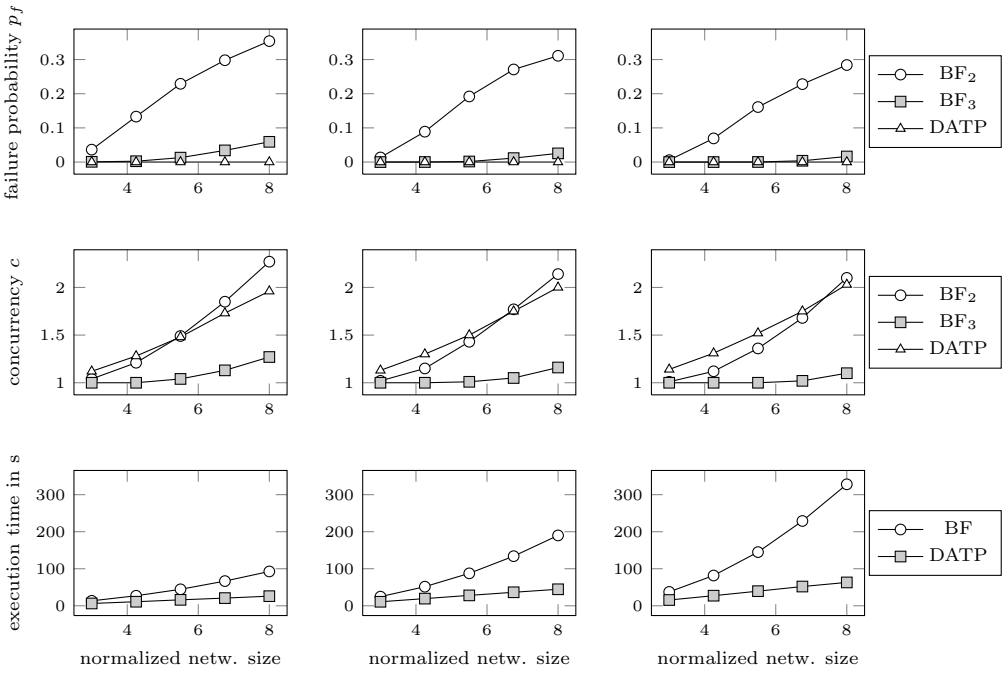
Figure 6 studies the influence of the compression coefficient γ on the three performance metrics. Figure 6 is organized as a matrix where each row presents one performance metric and each column is obtained for a different γ. The compression coefficient γ is defined in (1) and controls the number of time slots that have to be allocated to the nodes. For very high γ, every node has to be assigned exactly one time slot per TDMA frame. For low γ, the number of time slots assigned to a node depends on its number of descendants.

The first row of Figure 6 shows that reducing the compression coefficient γ reduces the failure probability p_f. This is because, for small γ, concurrency is low, as shown in the second row of Figure 6, and thus interference is low too. Concurrency is low because most of the time slots have to be assigned to the nodes close to the data sink, which are too close to each other to share a time slot.

The third row of Figure 6 shows that DATP's speed advantage over BF_k increases with γ. The execution time of BF_k barely decreases as γ increases despite the fact that a higher γ reduces the number of packets to be transmitted in the fourth phase. This is because, for low γ, most of these packets have to travel a short distance and BF_k's execution time is dominated by its second phase, whose duration is independent of γ. In contrast, the execution time of DATP greatly decreases with γ because γ reduces the number of time slot allocations and thus the number of CFs. Furthermore, increasing γ increases the concurrency and thus the number of winners per CF.

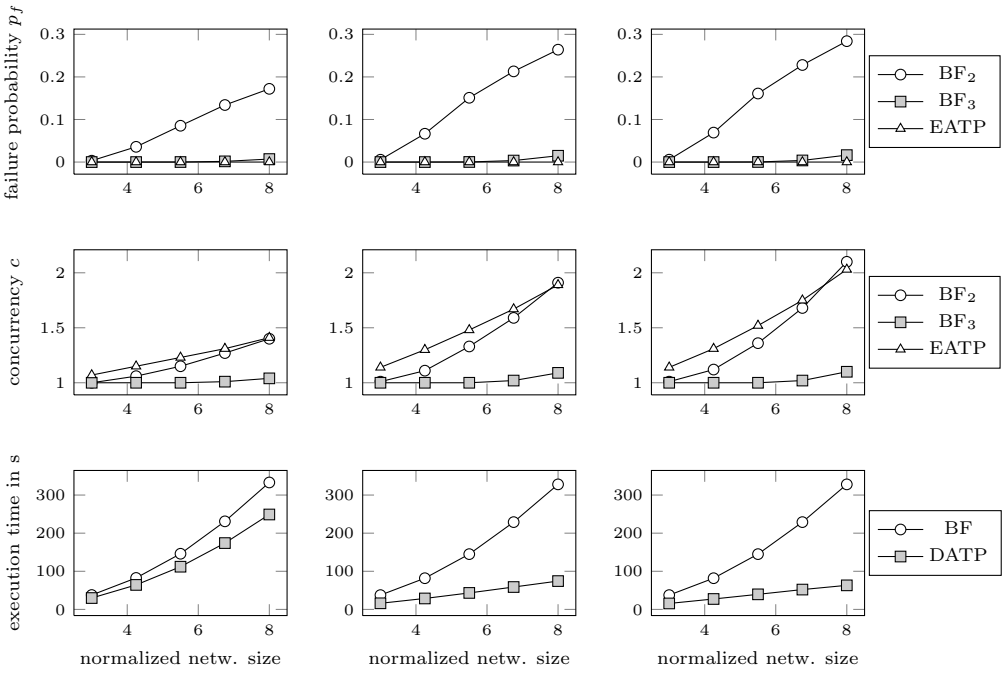
7.3 Evaluation in a network with fluctuating links

Figure 7 studies the influence of the link fluctuations on the three performance metrics. The figure is organized as a matrix where each row presents one performance metric and each column is obtained for a different value of the standard deviation σ_f



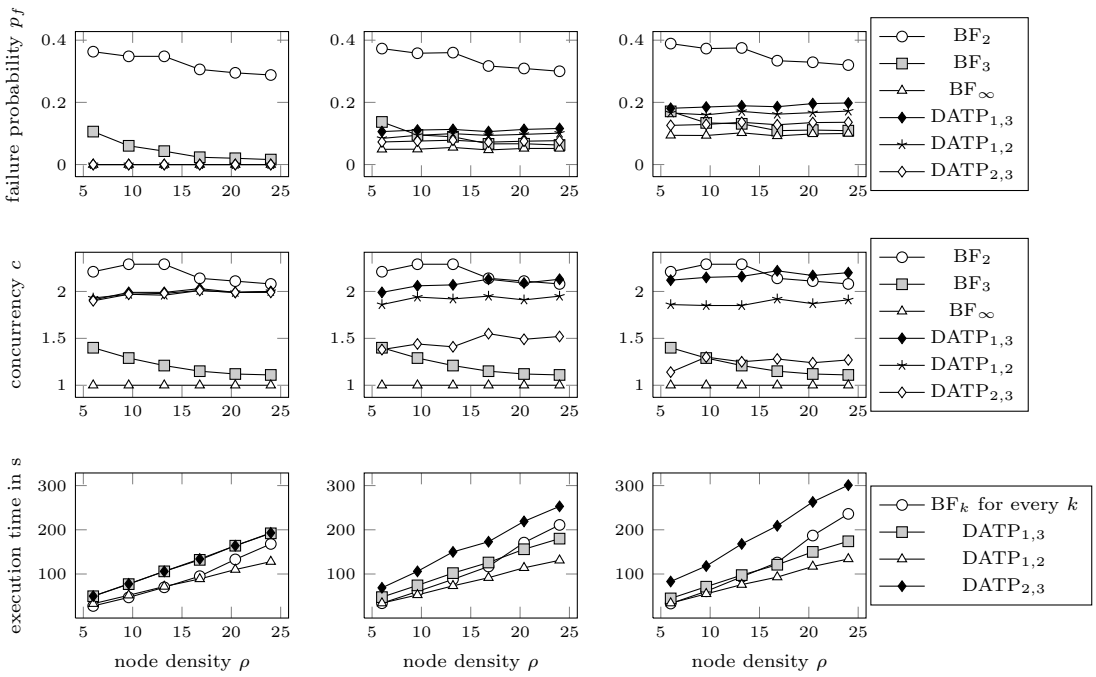
(a) $\rho = 9.6$ (b) $\rho = 16.8$ (c) $\rho = 24.0$

Figure 5: Scalability with both the normalized network size $\bar{x}$ and the node density ρ.



(a) $\gamma = 0$ (b) $\gamma = 10$ (c) $\gamma = 10000$

Figure 6: Influence of the compression coefficient γ.



(a) $\sigma_f = 0.0\,\mathrm{dB}$ (b) $\sigma_f = 0.5\,\mathrm{dB}$ (c) $\sigma_f = 1.0\,\mathrm{dB}$

Figure 7: Influence of link fluctuations, characterized by the standard deviation σ_f.

of the link fluctuation in dBs. This set of simulations used a compression coefficient γ of 10000.

For $\sigma_f > 0$, no schedule can guarantee zero failure probability p_f. The schedule with the lowest possible p_f is BF_∞ because BF_∞ assigns different time slots to different nodes and thus no packets are lost due to interference from other concurrent transmitters.

We simulate different versions of DATP, namely $DATP_{1,3}$, $DATP_{1,2}$ and $DATP_{2,3}$. In the notation $DATP_{S,A}$, the first subscript is the number S of successful receptions that are required in a superslot in order to consider a reception successful, and the second subscript represents the number of slots per superslot.

$DATP_{S,A}$ requires a success ratio of S/A in a superslot. As this ratio increases, fewer contenders succeed in a CF, fewer nodes are assigned the same transmission slot, and the concurrency c of the schedule decreases. A low concurrency c reduces the spectral efficiency, but reduces the amount of interference and the failure probability p_f.

Figure 7 shows the tradeoffs suffered by the three DATP versions. $DATP_{2,3}$ has the highest S/A ratio, and thus the lowest failure probability p_f and the lowest concurrency c. It also has the longest execution time because it uses the greatest number of CFs and these CFs are at least as long as those of the other two versions. $DATP_{1,3}$ has the lowest S/A ratio, and thus the highest failure probability p_f and the highest concurrency c. $DATP_{1,2}$ is faster than $DATP_{1,3}$ because its superslots are shorter.

The best protocol in a network with fluctuating links depends on the relative importance of the three performance metrics. If the execution speed is paramount and the network is large and dense, the best protocol is $DATP_{1,2}$. $DATP_{1,2}$ also enjoys higher concurrency than BF_3, but suffers a higher failure probability p_f than BF_3. If minimizing the failure probability p_f is much more important than maximizing the concurrency and the execution speed, the best protocol is BF_∞.

8 Conclusions and future work

Periodic low-latency data aggregation requires a TDMA transmission schedule verifying the precedence property. The existing protocols to obtain such a schedule are centralized, obtain low concurrency, or suffer a high failure probability p_f. They consist of assigning the earliest unused time slot within k hops to every node.

DATP, our proposed TDMA scheduling protocol, follows a different approach. It builds the transmission schedule distributedly and incrementally through a contention-based process. The contenders transmit dummy packets in order to determine empirically whether they tolerate the interference of other concurrent transmitters.

DATP offers the greatest advantages in large, dense networks with a high compression coefficient γ and stable wireless links. In this case, the failure probability of DATP is zero whereas that of BF_2 and BF_3 is significant. Furthermore, DATP is faster and achieves more concurrency.

If the compression coefficient γ decreases, the concurrency of the three protocols decreases because transmissions are concentrated in a small area. As a result, BF_2 and BF_3 suffer lower failure probability p_f, whereas DATP retains $p_f = 0$. DATP also enjoys a faster operation than BF_k, but the performance gap is reduced.

If the wireless links fluctuate, it is impossible to obtain $p_f = 0$. DATP has a p_f that is lower than that of BF_2 and slightly higher than that of BF_3. Link fluctuations slow down DATP because every set of concurrent transmissions must be tested multiple times. However, $DATP_{1,2}$ is faster and achieves more concurrency than BF_3.

In conclusion, under a variety of circumstances, some versions of DATP are faster and achieve more concurrency than BF_k. In addition, if links are stable, DATP guarantees zero failure probability p_f whereas BF_2 and BF_3 suffer high p_f. These results prove the usefulness of DATP's test-based scheduling approach.

As future work, DATP can be implemented in real hardware. The impact of synchronization errors can be assessed in order to decide suitable synchronization parameters. The choice of the maximum time that a node waits for its children to obtain a DB can be studied. The time and energy required to execute the data aggregation functions must be considered. The impact of nodes with unequal transceivers can be determined. The fluctuations of the wireless links can be measured to model them more accurately.

References

[1] T. Arampatzis, J. Lygeros, S. Manesis, A survey of applications of wireless sensors and wireless sensor networks, in: Proc. IEEE Mediterranean Conf. Control and Automation, Limassol, Cyprus, 2005, pp. 719–724. doi:10.1109/.2005.1467103.

[2] E. Fasolo, M. Rossi, J. Widmer, M. Zorzi, In-network aggregation techniques for wireless sensor networks: a survey, IEEE Wireless Communications Magazine 14 (2) (2007) 70–87.

[3] K. G. Langendoen, Medium access control in wireless sensor networks, in: H. Wu, Y. Pan (Eds.), Medium Access Control in Wireless Networks, Volume II: Practice and Standards, Nova Science Publishers, Hauppage, New York, 2008, pp. 535–560.

[4] W. L. Lee, A. Datta, R. Cardell-Oliver, FlexiTP: A flexible-schedule-based TDMA protocol for fault-tolerant and energy-efficient wireless sensor networks, IEEE Transactions on Parallel and Distributed Systems 19 (6) (2008) 851–864. doi:10.1109/TPDS.2007.70774.

[5] V. Rajendran, K. Obraczka, J. García-Luna-Aceves, Energy-efficient, collision-free medium access control for wireless sensor networks, Springer J. Wireless Networks 12 (1) (2006) 63–78.

[6] V. Rajendran, J. García-Luna-Aceves, K. Obraczka, Energy-efficient, application-aware medium access for sensor networks, in: Proc. IEEE Int'l Conf. Mobile Ad-Hoc and Sensor Systems Conf. (MASS), 2005, pp. 630–637. doi:10.1109/MAHSS.2005.1542852.

[7] V. Annamalai, S. K. S. Gupta, On tree-based convergecasting in wireless sensor networks, in: S. Gupta (Ed.), Proc. IEEE Wireless Comm. and Networking Conf. (WCNC), Vol. 3, 2003, pp. 1942–1947.

[8] R. Mangharam, A. Rowe, R. Rajkumar, FireFly: a cross-layer platform for real-time embedded wireless networks, Springer J. Real-Time Systems 37 (3) (2007) 183–231.

[9] G. Zhou, T. He, S. Krishnamurthy, J. A. Stankovic, Impact of radio irregularity on wireless sensor networks, in: Proc. ACM Int'l Conf. Mobile Systems, Applications, and Services (MobiSys), Boston, MA, USA, 2004, pp. 125–138. doi:http://doi.acm.org/10.1145/990064.990081.

[10] M. Maróti, B. Kusy, G. Simon, Á. Lédeczi, Robust multihop time synchronization in wireless sensor networks, in: Proc. Int'l Conf. Wireless Networks (ICWN), 2004, pp. 454–460.

[11] M. O. Díaz-Anadón, K. K. Leung, Efficient data aggregation and transport in wireless sensor networks, Wiley J. Wireless Comm. and Mobile Computing.

[12] S. Pattem, B. Krishnamachari, R. Govindan, The impact of spatial correlation on routing with compression in wireless sensor networks, Proc. ACM/IEEE Conf. Information Processing in Sensor Networks (IPSN) (2004) 28–35.

[13] N. Shrivastava, C. Buragohain, D. Agrawal, S. Suri, Medians and beyond: new aggregation techniques for sensor networks, in: Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys), 2004, pp. 239–249.

[14] R. Cristescu, B. Beferull-Lozano, M. Vetterli, R. Wattenhofer, Network correlated data gathering with explicit communication: NP-completeness and algorithms, IEEE/ACM Transactions on Networking 14 (1) (2006) 41–54.

[15] A. F. Harris III, R. Kravets, I. Gupta, Building trees based on aggregation efficiency in sensor networks, Elsevier J. Ad Hoc Networks 5 (8) (2007) 1317–1328.

[16] J. Polastre, J. L. Hill, D. Culler, Versatile low power media access for wireless sensor networks, in: Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys), 2004, pp. 95–107.

[17] W. Ye, J. Heidemann, D. Estrin, Medium access control with coordinated adaptive sleeping for wireless sensor networks, IEEE/ACM Transactions on Networking 12 (3) (2004) 493–506.

[18] I. Rhee, A. Warrier, M. Aia, J. Min, M. L. Sichitiu, Z-MAC: a hybrid MAC for wireless sensor networks, in: Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys), ACM Press New York, NY, USA, 2005, pp. 90–101.

[19] Y. Zhu, R. Vendantham, S.-J. Park, R. Sivakumar, A scalable correlation aware aggregation strategy for wireless sensor networks, in: Proceedings of the First Int'l Conf. Wireless Internet, 2005, pp. 122–129.

[20] I. Rhee, A. Warrier, M. Aia, J. Min, M. L. Sichitiu, Z-MAC: A hybrid MAC for wireless sensor networks, IEEE/ACM Transactions on Networking 16 (3) (2008) 511–524.

[21] G. P. Halkes, K. G. Langendoen, Crankshaft: An energy-efficient MAC-protocol for dense wireless sensor networks, in: Proc. European Conf. Wireless Sensor Networks (EWSN), Springer, 2007, pp. 228–244.

[22] T. Zheng, S. Radhakrishnan, V. Sarangan, PMAC: an adaptive energy-efficient MAC protocol for wireless sensor networks, in: Proc. IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS), 2005, pp. 65–72.

[23] L. van Hoesel, P. Havinga, A lightweight medium access control (LMAC) for wireless sensor networks, in: Proc. Int'l Workshop Networked Sensing Systems (INSS), 2004, pp. 1–4.

[24] M. O. Díaz-Anadón, K. K. Leung, A test-based scheduling protocol (TBSP) for periodic data gathering in wireless sensor networks, in: A. Vinel (Ed.), 3rd International Workshop on Multiple Access Communications (MACOM), Lecture Notes in Computer Science, Springer, Barcelona, 2010, pp. 25–35.

[25] S. Borbash, A. Ephremides, Wireless link scheduling with power control and SINR constraints, IEEE Transactions on Information Theory 52 (11) (2006) 5106–5111.

[26] G. Brar, D. M. Blough, P. Santi, Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks, in: Proc. ACM/IEEE Int'l Conf. Mobile Computing and Networking (Mobicom), Los Angeles, CA, USA, 2006, pp. 2–13.

[27] S. Kompella, H. Sherali, A. Ephremides, Optimal scheduling in interference limited fading wireless networks, in: Proc. IEEE Global Telecommunications Conf. (GLOBECOM), Honolulu, HI, USA, 2009, pp. 1–6.

[28] X. Liang, W. Li, T. A. Gulliver, An energy-efficient MAC protocol exploiting the tree structure in wireless sensor networks, in: W. Li (Ed.), Proc. IEEE Military Comm. Conf. , 2007, pp. 1–7.

[29] M. O. Díaz-Anadón, Implementation of TPDA, `https://github.com/ornediaz/wsnpy/blob/master/y3/ne3.py`.