Combined Time Synchronization and Efficient Data Gathering for Wireless Sensor Networks

Application to Micaz® motes

Jérôme Mathieu
IUT Rodez
University Toulouse 1
Rodez, France
jerome.mathieu@iut-rodez.fr

Vincent Boudet, Jérôme Palaysi
LIRMM
University Montpellier 2
Montpellier, France
vincent.boudet@lirmm.fr
jerome.palaysi@univ-montp2.fr

Sylvain Durand
LIRMM
University Montpellier 3
Montpellier, France
sylvain.durand@lirmm.fr

***Abstract*—This paper presents an algorithm for wireless sensor networks, which combines time synchronization and efficient data gathering. The synchronization part is novel, using multiple ways between the reference and the node to be synchronized to. The data gathering part resumes a previous work. It uses various connected dominating sets for increasing the lifetime. This algorithm is calibrated to Micaz® motes by performing some experiments with them. The synchronization part is then validated on a simple Micaz® network. That shows that our algorithm can effectively merge time synchronization and data gathering with no special message and minimal overhead for synchronization.**

Keywords-Synchronization; Efficient data gathering; Wireless sensor networks; Micaz

I. Introduction

The main specificity of Wireless Sensor Networks (WSN) is due to the limited energy and capabilities (RAM/ROM, processor, etc.) of the motes (nodes). Those limited capabilities require to adopt simple algorithms, especially for annex functions like data gathering. The limited energy requires to search low consumption solutions. One of the best strategies is to reduce the duration of mote's awake periods and the number of communications.

Especially in this constrained context, data gathering needs time synchronization between motes. Otherwise, whether the gathering fails or energy is unnecessarily lost.

Data gathering needs also to route data to the sink point. Therefore, all nodes does not have the same role. Some of them are in the backbone, and then have more jobs to do and de facto consume more energy. The network dies because the backbone nodes deplete their energy if nothing is done.

Many works (see the state of the art in Sections II and III) treat one of the mentioned problems: either efficient routing or synchronization. But, to our knowledge, none of them treats both in a single solution. Consequently, in practice, nodes accumulate communications and jobs to do.

In this article, we propose an algorithm, which combines an existing efficient data gathering [1] and a novel synchronization method, which does not increase the mote awake duration nor the number of messages in comparison to the data gathering performed alone, while minimizing the overhead in messages.

In the following section, we remind the gathering part of the algorithm. In Section III, we explain our novel time synchronization protocol. Section IV presents the specificities of the Micaz® sensors used in the experiments. We present in Section V the algorithm combining gathering and synchronization and its performances in Section VI. This article finishes by giving the conclusions and some perspectives to this work.

II. Efficient Data Gathering

One typical way is to use Connected Dominating Sets (CDS), also called backbones, to route data to the sink. The nodes belonging to a backbone use more energy to forward data because they are much more messages to wait and treat: at least one by child. The goal is then to minimize the number of nodes in a backbone [2]. Because of robustness, scalability and mainly efficiency requirements, most algorithms are operating in a distributed manner with local election of the nodes belonging to the backbone [3]. But, after a certain period of time, the network will be disconnected while the leaf nodes may still have a lot of energy. An improvement (in order to increase the lifetime) is to compute several disjoint backbones [4]. One then tries to use those backbones alternatively. Unfortunately, computing the maximum number of disjoint CDS (called connected domatic number) is a hard task. Furthermore, this number may be very small. For example, Islam et al. [5] show that for a grid graph where one of the dimensions is greater or equal to 3, the connected domatic number is 1. Thus in such a graph, one cannot expect to increase the lifetime of the network using disjoint CDS.

The disjoint constraint may be relaxed by trying to find a set of CDS such that the maximum number of CDS a node belongs to is minimized. Such a distributed algorithm is proposed in [6]. Nevertheless, this model does not take into account the real consumption of energy of the nodes, which depends (among others) on the number of received messages, i.e., on the degree of the node.

In our case, all data have to be gathered to a fixed sink. We thus only have to compute a directed in-tree rooted at the

 ISBN: 978-1-61208-296-7

sink, and the sink may initiate this computation (the algorithm is not localized but only distributed). This specification allows us to need only a small number of messages to compute a backbone.

III. Time Synchronization

In a perfect data gathering, all the nodes receive data from their children simultaneously and send data to their parent a short time later. After that, nodes can enter in a sleeping period until the next gathering. Without any time synchronization, all nodes must wait each other, canceling the sleeping period. Of course, the greater the sleeping period duration is, the longer the lifetime of the network is. Unfortunately, accurate time synchronization protocols [7] imply exchange of many messages between nodes and a lot of calculations, which reduce the sleeping period duration. Furthermore, nodes cannot send messages simultaneously because of wireless media access control protocols. So, in our point of view, a good time synchronization protocol does not need to be very accurate but should not add messages and should not reduce the sleeping period duration.

Typically, each node must evaluate two parameters [8]: the offset, i.e., the difference between the reference clock and the node clock, and the skew due to the drift of a node clock relative to a perfect clock. Offset is the leading parameter in the short-term, for example because nodes are not switched on simultaneously. Skew becomes important in the long-term, for example when data gatherings are infrequent.

A. Offset computation process

Offset is calculated [8] whether by a round-trip between the reference node and an evaluated node, or after the estimation of the delay D involved in the transport of a message between these two nodes. The first type of protocols supposes that D is equal in each way. The accuracy of all methods depends on the quality measurements of D. In practice, D varies from one message to the other and is in part unpredictable. That's why all methods use statistics to improve the offset estimation. But, except for RBS [9], these statistics are based on repetitive exchanges of messages between always the same nodes. The result is assigned to a specific node. The complexity increases when the evaluated node is not directly reachable by the time reference node. Classically [7,9], the synchronization is then achieved in stages, some nodes being elected as intermediate time reference.

Our strategy, inspired by RBS, is different from two points. First, if the network is homogeneous – i.e., all nodes have the same architecture, execute the same program, etc. - statistics are calculated spatially instead of temporally. Second, the evaluated node doesn't need to be directly reached by the time reference node. Suppose a simple network as shown in Figure 1. Node 0 sends by broadcast a message M_1 at time Tr, so M_1 is received approximately at the same time by node 1 and by node 2 (The difference in distance between nodes involves a negligible time difference). Node 2 sends the message M_2 immediately after receiving M_1. Node 1 receives M_1 at $T[0]$ and, a few later, M_2 at $T[1]$. Then, from the point of view of node 1, $D=T[1]-T[0]$. In order to do so, D must be considered as a constant in the network.

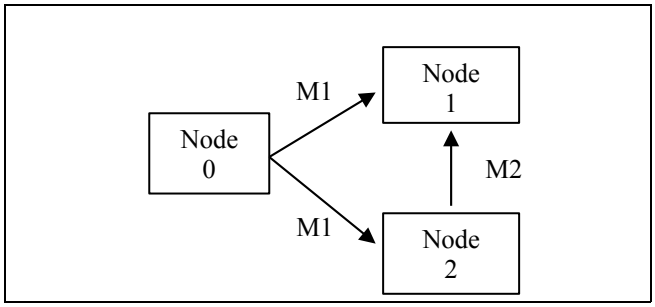

Figure 1. Synchronization by two ways

If fact, D is the difference of the delay between nodes 0 and 1 passing by node 2 (=2D) and the delay between nodes 0 and 1 (=D). If the time reference node is node 0, the offset of node 1 is $T[1]-T[0]-D$. So, the reference time Tr must be incorporated at least in M_1.

Our method requires to calculate a tree, which is necessary for data gatherings anyway, and each node must have at least two neighbors. The synchronization can be done along with the tree calculation. The reference time is that of the root of the tree, i.e., the sink.

Formally, the evaluated node receives a message from its parent (which is at a depth $d[0]$ in the tree) at time $T[0]$ and from other neighbors (depth $d[i]$) at time $T[i]$. All $d[i]$ must be different from $d[0]$. Then, the delay D from the point of view of the evaluated node is:

$$D = \text{mean}((T[i]-T[0]) / (d[i]-d[0])) \quad (1)$$

and its offset can be calculated by:

$$\text{offset} = \text{mean}(T[i]-Tr-d[i]*D) \quad (2)$$

This is done each backbone calculation.

B. Skew computation process

The skew is often supposed to be a constant [7,8,9], at least over a period of a few minutes. Typically, something like the offset variations over time are fitted by a line (linear regression, for example). The skew is the slope of this line. For that, many offsets are calculated in a relatively short time. Then, a good skew estimation requires a very high offset accuracy and especially much more messages consuming the battery.

We propose to evaluate the skew using data gatherings messages only. In each data gathering, a node sends its data to its parent node in the tree. To ensure the sending, the parent node returns an acknowledgment to its child node. This <ACK> message contains the time of the parent node. Then, the child node can follow over time its clock drift regarding its parent node without any additional message. Moreover, the skew value is not important in itself. The main

 ISBN: 978-1-61208-296-7

objective of a node is to keep synchronized to its parent over time. For that, if $\Delta T[i]$ is the difference between a child's time and its parent time observed at the i^{th} gathering, then $\Delta T[i+1]$-$\Delta T[i]$ is more important (equal to the skew times the time interval between two gatherings). For the first gatherings, our algorithm provides sufficient margin of errors from experimental results.

IV. The Micaz® motes

The objective is to implement our algorithm in the Micaz®'s platform. Some steps of it are adjustable, depending on the platform.

A. Energy consumption

The transceiver (radio model TI CC2420) is the component that has the highest energy consumption of all relevant components of the Micaz® [10]. In particular, the receive mode consumes 19.7 mA while the processor needs "only" 8 mA in its active mode. When the transceiver is in the transmit mode, the current consumption varies from 8.5 to 17.4 mA depending on the chosen transmit power. Then, the total consumption is always about 28 mA. So, for energy saving, the transceiver must be off as often as possible. That means reducing the number of messages and the time it may be used.

The reference objective of this study is one gathering every minute and a lifetime of one year (a directive of a partner). A node in the economical state consumes approximately 20µA (CC2420 power down mode) + 15µA (CPU save mode). The consumption of other components is neglected here. In one year, an idly node consumes 35µA times 8760h (1 year), that is about 307 mAh. A Micaz® embeds two AA batteries. If their capacity is 2200 mAh each, the lifetime in active mode is about (4400-307)/28 = 146 h. If this lifetime is splitted, each data gathering (with or without backbone calculation) must take at most 1 s.

Note that the CC2420 crystal oscillator start-up time is 1 ms, the condition to switch from the power down mode to the idle mode from which the communications can restart. But the idle mode consumes too much (about 0.43 mA). It is an another proof that energy saving is not completely compatible with high accuracy synchronization protocols.

To validate these calculations, one could use an accurate tool of energy consumption prediction [11]. But, to be really useful here, the tool should also integrate a model of battery. For example, Alkaline batteries, such they use, have a cut-off voltage smaller than the minimum voltage supply for Micaz® and their capacity decreases if the load increases [12]. All of that make the battery lifetime prediction hazardous. So, the mote lifetime is evaluated by some experiments. A Micaz® is programmed to send a message every minute and to wait the rest of the time. The transceiver is switched off a percentage of the duty cycle, after the sending. A LED is blinking 5% of the duty cycle, only for control. It represents an additional amount of consumed energy of about 0.5%.

The lifetime of the mote is reported in Table I. We can see that the lifetime of the mote fully active is not a constant because the behavior of the alkaline batteries. But, its tendency suggests that 146 h is probably easily reachable especially when the active time of the transceiver will be less than 1 %.

TABLE I. Evolution of the Mote Lifetime vs the Percentage of the Active Time of the Transceiver

Active time of the transceiver (%)	100	75
Mote lifetime (h)	113.80±0.66	178.09±1.52
Mote fully active lifetime (h)	113.8	133.6

B. Communication delays

Firstly, our algorithm has a flaw: when a node has only one neighbor (its parent), the delay D can't be calculated as shown above. It must be given as a platform constant for the offset calculation process. But the skew correction process finely corrects the local clock, if necessary, at the first data gathering.

Secondly, in the data gathering phases, backbone's nodes have to wait all their child nodes before sending its data to its parent node. A timer (timeout) is necessary otherwise a deadlock situation may occur if one child node fails. The timer setting is difficult both to ensure the time needed to gather data and to save energy.

The following experiments are used to evaluate the real delays D depending on the number of neighbors. A Micaz® node, randomly chosen, called the evaluated node *eN*, is programmed to send a broadcast message that contains its system time S_s. Neighbor(s) send(s) back this message without any modification (echo) to *eN* as soon as possible. The nominative sendings allow not saturating unnecessarily the neighbors. At the reception of the echo by the neighbor *Vi* (one per neighbor), *eN* gets as soon as possible its associated system time $S_e[Vi]$ and sends it back to *Vi* in a <ACK> message. Neighbors have a timer associated with the sending of the echo. If the timer fires without receiving this <ACK>, they send echo again. Each neighbor receiving this <ACK> message stops until the next test. This ensures that *eN* has processed all neighbors. It is necessary because neither CSMA/CA nor the beacon mode are incorporated by default in Micaz®. We only have the clear channel assessment. At the end of a trial, *eN* sends S_s and all $S_e[Vi]$ to a wireless network "sniffer". This "sniffer" captures all exchanges too. $S_e[Vi]$-S_s is the *eN*'s delay D plus the *Vi*'s one.

This procedure is repeated 30 times, every 5s, for statistical analysis. The repetition rate is chosen to minimize the influence of the clock drift while leaving time for nodes to prepare for the next round.

The system time accuracy is estimated at 0,25 ms.

Table II shows that the nodes, even placed in the same situation, do not necessarily behave in the same way (see 1 neighbor column). When the number of neighbors increases, the mean delay, the standard deviation and the max value increase but not evenly for all nodes. Nodes little bit more reactive maintain relatively low values, but others not. With 3 or 4 neighbors, it is better to forecast 100 ms to gather data.

 ISBN: 978-1-61208-296-7

Otherwise, one of the nodes does not have the time to send data to its parent. This table gives also an idea of the needed synchronization accuracy. It seems that accuracy around 5 ms is enough.

We must specify that all these results depend on the chosen timer. Indeed, it is one of our future works to refine the timer formulation.

TABLE II. EVOLUTION OF DELAYS VS THE NUMBER OF NEIGHBORS

Number of neighbors	Evaluated node's delay + neighbor's delay			
	Neighbor	*Mean (ms)*	*Standard deviation (ms)*	*Max value (ms)*
1	"1"	3.1	2.1	7.3
	or "2"	5.0	3.2	13.0
2	"1"	6.6	9.3	39.1
	"2"	5.4	8.2	41.6
3	"1"	11.5	20.2	104.9
	"2"	8.0	9.0	30.3
	"3"	11.3	17.7	89.2
4	"1"	6.3	7.2	30.1
	"2"	9.4	9.6	38.7
	"3"	7.6	7.6	29.6
	"4"	12.0	20.6	97.8

Table III shows the most important consequence when the number of neighbors increases. Many messages are lost by *eN* because it is busy by treating other message just received.

TABLE III. EVOLUTION OF PERCENTAGE OF LOST MESSAGES AND RETRANSMISSIONS VS THE NUMBER OF NEIGHBORS

Number of neighbors	Percentage of lost messages	Percentage of retransmissions
1	0	0
2	10	16 to 20
3	20	26 to 38
4	32	40 to 70

C. Clock skew

Two motes are programmed to send a message every minute to the "sniffer". Then, the mote's clock can be compared to the computer's clock, which is supposed to be accurate on the experiment's duration (about 80 h). The first mote skews of +6.6 s (about +2.0 s per day or +23 ppm), while the second one skews of +4.6 s (about +1.4 s per day or +16 ppm). These results corroborate some previous works [13].

V. THE PROPOSED ALGORITHM

Our algorithm includes two phases:

- a backbone calculation + synchronization phase,
- a data gathering + skew correction phase.

The first phase is repeated using several conditions. The basic condition is the number of gatherings: each *X* gatherings, a new backbone calculation is initiated by the sink node. Other conditions to initiate a new backbone calculation are relative to data loss (just one data or beyond a percentage threshold). The choice of these conditions depends on the application. The choice of *X* (not necessary a constant) depends on the percentage of energy consumed during a gathering phase.

The second phase must incorporate a signaling procedure if a condition on data loss is used. Indeed, leaf nodes in particular do not know if a gathering is complete or not, and consequently if a new backbone calculation will be initiated or not. Between 2 first phases, gatherings (second phases) are repeated with a rhythm depending on the application.

A. The backbone calculation + synchronization phase

This phase also incorporates the neighbor discovery. This is done by observing the ID of each received message.

There are 3 kinds of messages:

- <INV>: the "invitation" message, broadcasted. It contains the backbone ID *B*, the sender ID, the sink time *Ts*, the total waiting in the path *W*, the sender level in the tree *L*, etc.
- <F>: the "parent" message, sent repeatedly to the parent node until <ACK> is received, contains *B*, the sender ID, etc.
- <ACK>: the acknowledge message, not mentioned in Figure 2.

The principle of the backbone calculation is that each node will choose as a parent in the backbone the first node from which it received an "invitation" message. Since all the nodes but the sink send their invitation after receiving one, this ensures that we built a directed in-tree rooted at the sink. Fine-tuning of the algorithm is done by the computation of the delay *w* each sensor has to wait before sending an invitation. The main idea is that less remaining energy the node has, longer the delay is. Of course, if the delay increases, the probability for a node that its <INV> will be the first received by its neighbors decreases. Thereby, the probability that the node belongs to the backbone decreases too. This tends to calculate different backbones and equalize the energy consumption of nodes. The main consequence is an increasing of the network lifetime. More details can be found in [1].

Note that if the delay is constant for all nodes, then this phase is formally identical to a Breadth First Search (BFS) with incorporated synchronization.

Finally, it is noteworthy that no additional message is required to ensure the synchronization. The three ones are necessary for the backbone calculation. And the additional computations are very light.

B. The gathering + skew correction phase

The backbone nodes wake up at the same time as its child nodes because they must be ready to receive <DATA>. This time is depending on the level in the backbone,

following the staggered sleep scheduling scheme like in D-MAC [14] but at the application level. Working at the application level instead of the MAC level allows to reduce the transmission slot duration as soon as possible.

For the sink node (also the reference time node) *S*, send <INV> ; *B*++

For all other nodes *N* do:

- **Receive the first <INV>**:
  (parent ID = sender's ID) and (*N*'s *L* = *L*+1)
  Note *Ts*, *L*[0]=*L*, *T*[0]=*T*-*W*, where *T* is the local system time
  Send <F>
  Chose a delay *w*; *W* = *W* + *w*; wait *w*
  Send <INV>

- **Receive other <INV>**:
  If (*L* ≠ *L*[0]), *L*[*i*]=*L*, *T*[*i*]=*T*-*W*, *nb_inv*++, *i*++

- **Receive a <F>**:
  N ∈ backbone

- **Just before go sleeping**:
  If (*N* is a parent of none node) then *N* ∉ backbone
  If (*nb_inv* > 0) apply Equation (1) else *D* = 5 ms
  Apply Equation (2) (with *Tr* = *Ts*)

Figure 2. The backbone calculation + synchronization phase

For the leaf nodes do:
Send <DATA> to its parent repeatedly until <ACK> is received
T = *Tf*
If (ALERT=Yes) for the first time, return to the backbone calculation + synchronization phase in *C* gatherings
else go sleeping until the next gathering

For the backbone's nodes do:
Receive <DATA> from a child *N*
Send <ACK> with its local system time *Tf* and ALERT (ALERT=No by default) to *N*
Wait for the end of the gathering period (timer)
Behave as a leaf node

Figure 3. The gathering + skew correction phase

The simplest and energy saving manner for signaling a new backbone calculation is to use an ALERT flag (see Figure 2). ALERT is incorporated in <ACK> messages, associated to the maximum depth in the tree (*DT*) and the sender's time, i.e., the parent's time *Tf*.

It allows to change the node phase together with other nodes through the counter *C*=*DT*-*L*. Note that *DT* gatherings are necessary before effectively initializing a new backbone calculation. It is the main inconvenient of this method. Another solution, when data are vital, is to keep all nodes pending the decision of the sink. That means that all nodes wait the end of the gathering for the ALERT signal, consuming uselessly their energy.

The affectation *T*=*Tf* may seem a bad formula to correct the skew. Normally, the delay *D* must be taken into account. But *D* may overestimate the real transmission time because the context is usually more favorable. Then, at the next gathering, the child may wake up before its parent, failing the firsts sendings of data. It is thus preferable that leaf nodes wake up after their parent.

Once again, no additional message and few additional computations are necessary for the synchronization. The main part of the work is needed to ensure the dependability, the reliability of the data gathering.

The gathering period should last the time required for all children to send their data, that is for example about 100 ms for a network with 4 neighbors per node (see table II). Then, a node in the backbone awakes about 115 ms (mean time) while a leaf node awakes only about 15 ms (mean time) (see table II). That is the reason for changing regularly the backbone.

A time of 115 ms fulfills the recommendations made in Section IV-A. If more than one gathering per minute is expected, the gathering period should be decreased, i.e., the medium access control should be improved (see Section IV-B).

VI. Validation of the Synchronization Protocol on Micaz®

The backbone calculation + synchronization (BCS) phase (Figure 2) is programmed on 5 motes. Another node is programmed as a sink node. The nodes are switched on randomly. The BCS phase starts automatically about 30 s later the sink node has been switched on. For the experiments purposes, each node has an additional functionality: responding to a probe node. When the BCS phase is ended, the probe node sends (broadcast) a special message to all nodes including the sink node. Immediately, nodes have to record their local time. Then, they have to send it to the "sniffer". Clocks, corrected by the calculated offset, are compared to the sink's one, which is the time reference.

The experimental network is presented Figure 4. It is constructed such that several situations are treated:

- A node has only one neighbor (node 5), so *nb_inv* = 0 and *D* is fixed to 5 ms for this node (see Figure 2).
- Some nodes have 2 neighbors, at its same level or a smaller level (nodes 2 and 4).
- Some nodes have many - 3 or 4 - neighbors (nodes 1 and 3).
- The backbone has several levels (3), which is the maximum achievable with this experiment and only 2 "sniffers".
- The backbone is always the same to make statistics without influence of the resulting backbone.

 ISBN: 978-1-61208-296-7

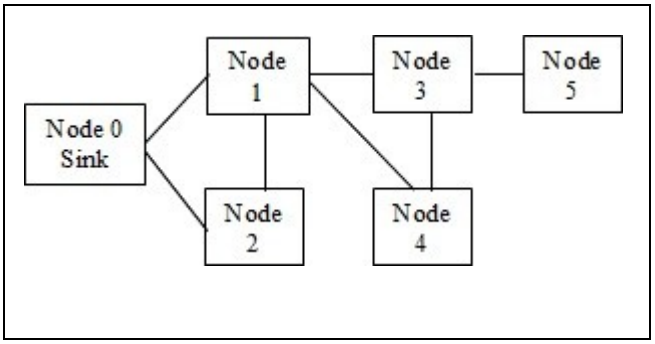

Figure 4. Experimental network

The results are presented in Table IV. They are good, even if they are probably worse than those that have been obtained with other synchronization protocols. They are of the same order of the measured delay shown in Table II, and this is what we hoped, even if the system time accuracy is worse (slower clock rate).

TABLE IV. CLOCK DIFFERENCES BETWEEN THE SINK NODE AND THE OTHER NODES

Node	Mean of clock differences (ms)	Standard deviation (ms)
1	6.6	5.27
2	1.0	8.19
3	3.2	7.33
4	2.4	11.76
5	7.6	3.58

Some remarks should be noted.

First, as expected, and unlike other protocols, the mean of the clock differences does not significantly increase with the depth of the node in the tree. However, our results are almost always positive implying that the delay is probably systematically underestimated. This is probably due to computation time that is not properly taken into account. An optimization of the program should solve the problem.

Second, the number of neighbors influences differently the mean and the standard deviation. When a node has many neighbors, the mean tends to increase and the standard deviation tends to decrease. This shows that our method tends to converge, even if it is towards a value slightly underestimated (cf. the first remark, with cumulative effects).

Third, node 5 is a special case. Its particularly low standard deviation shows that the real delay is fairly stable from one experiment to another. For cons, its mean shows that the default value given to D is also slightly underestimated.

VII. CONCLUSION AND FUTURE WORKS

In this paper, we presented a new protocol that realizes both gathering and a new synchronization method in a wireless sensor network. The load for the gathering is distributed over all the nodes and there is no special message needed for the synchronization.

The perspectives of this work are to refine the timer's formulations and mainly the sleep scheduling for a more deterministic one. Thereafter, the protocol will be tested with a more complex experimental network.

REFERENCES

[1] V. Boudet, S. Durand, L. Gönczy, J. Mathieu, and J. Palaysi, “Efficient gatherings in wireless sensor networks using distributed computation of connected dominating sets”, Sensors & Transducers Journal, vol. 14-2, 2012, pp. 297-307.

[2] S. Guha and S. Khuller, “Approximation algorithms for connected dominating sets”, Algorithmica, vol. 20, April 1998, pp. 374-387.

[3] C. Adjih, P. Jacquet, and L. Viennot, “Computing connected dominated sets with multipoint relays”, Adhoc & Sensor Wireless Networks, vol. 1, no. 1, 2004, pp. 27-39.

[4] T. Moscibroda and R. Wattenhofer, “Maximizing the lifetime of dominating sets”, Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - 5th IEEE International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN 2005), Denver, USA, vol. 13, April 2005, p. 242b. doi: 10.1109/IPDPS.2005.276.

[5] K. Islam, S. G. Akl, and H. Meijer, “A constant factor localized algorithm for computing connected dominating sets in wireless sensor networks”, Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS'08), Melboune, Australia, 2008, pp. 559-566.

[6] K. Islam, S. G. Akl, and H. Meijer, “Distributed generation of a family of connected dominating sets in wireless sensor networks”, in Distributed Computing in Sensor Systems, vol. 5516, B. Krishnamachari, S. Suri, W. Heinzelman, and U. Mitra, Ed., Springer Berlin / Heidelberg, 2009, pp. 343-355.

[7] S. el Khediri, N. Nasri, M. Samet, A. Wei, and A. Kachouri, “Analysis study of time synchronization protocols in wireless sensor networks”, Int. J. of Distributed & Parallel Systems, vol. 3, no. 3, May 2012, pp. 155-165.

[8] F. Sivrikaya and B. Yener, “Time synchronization in sensor networks: a survey”, IEEE Network, vol. 18, issue 4, July 2004, pp. 45-50.

[9] J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts”, Proceedings of the 5th symposium on Operating systems design and implementation (OSDI'02), Boston, USA, December 2002, pp. 147-163.

[10] http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf

[11] O. Landsiedel, K. Wehrle, and S. Götz, “Accurate prediction of power consumption in sensor networks”, Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNetSII), Sydney, Australia, 2005, pp. 37-44.

[12] http://en.wikipedia.org/wiki/Alkaline_battery

[13] M.B. Uddin and C. Castelluccia, “Toward clock skew based wireless sensor node services”, Proceedings of the 5th Wireless Internet Conference (WICON) - International Workshop on Ubiquitous Body Sensor Networks (UBSN 2010), Singapore, March 2010, pp. 1-9.

[14] G. Lu, B. Krishnamachari, and C. Raghavendra, “An adaptive energy-efficient and low-latency MAC for data-gathering in sensor networks”, Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS'04) - 4th IEEE International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN), Santa Fe, USA, April 2004. doi: 10.1109/IPDPS.2004.1303264.

 ISBN: 978-1-61208-296-7