Self-repairing Clusters for Time-Efficient and Scalable Actor-Fault-Tolerance in Wireless Sensor and Actor Networks

Loucif Amirouche[1], Djamel Djenouri[2], and Nadjib Badache[2]

[1] El-Djazair Information Technology, Algiers, Algeria
l.amirouche@sonelgaz.dz
[2] CERIST Research Center, Algiers, Algeria
ddjenouri@mail.cerist.dz, badache@mail.cerist.dz

Abstract. A new solution for fault-tolerance in wireless sensor and actor networks (WSAN) is proposed. The solution deals with fault-tolerance of actors, contrary to most of the literature that only considers sensors. It considers real-time communication, and ensures the execution of tasks with low latency despite fault occurrence. A simplified MAMS (multiple-actor multiple-sensor) model is used, where sensed events are duplicated only to a limited number of actors. This is different from the basic MAMS model and semi-passive coordination (SPC), which use data dissemination to all actors for every event. Although it provides high level of fault-tolerance, this large dissemination is costly in terms of power consumption and communication overhead. The proposed solution relies on the construction of self-repairing clusters amongst actors, on which the simplified MAMS is applied. This clustering enables actors to rapidly replace one another whenever some actor breaks down, and eliminates the need of consensus protocol execution upon fault detection, as required by the current approaches to decide which actor should replace the faulty node. The extensive simulation study carried out with TOSSIM in different scenarios shows that the proposed protocol reduces the latency of replacing faulty actors compared to current protocols like SPC. The reduction of the overall delay for executing actions reaches 59%, with very close fault-tolerance (action execution success rate). The difference for this metric does not exceed 8% in the worst case. Scenarios of different network sizes confirm the results and demonstrate the protocol's scalability.

1 Introduction

A wireless sensor and actor network (WSAN) is a heterogenous networ where nodes communicate through wireless links to cooperatively monitor the environment and accordingly react on it. Sensors are small and usually static devices with limited resources, while actors or (actuators) are more powerful devices, equipped with more powerful resources. Actors are able either to move and perform appropriate actions, or launch an action on several actuation devices(action mobility). Sensors are responsible for sensing the physical environment, while actors use data collected by sensors to make appropriate decisions and accordingly

R. Szabó et al. (Eds.): AccessNets 2010, LNICST 63, pp. 113–123, 2011.

react on the environment. There is a variety of WSAN's applications, such as forest monitoring and fire extinguishing, battlefield surveillance, intrusion detection, automatic irrigation of cultivated fields, and last but not least biomedical applications. In many applications, tolerating the breakdown of sensors, and particularly actors is mandatory for real deployment. Many solutions offering fault-tolerance to sensors have been proposed thus far, but they completely ignore actor faults. One of the common techniques used to increase availability, and recently used to enable fault tolerance is the MAMS (multiple-actor multiple-sensor) model. In this model, every single event is distributed to all actors in the network. The few solutions dealing with actor fault-tolerance use this model, which in addition to the high complexity, it requires a consensus arrangement between actors for every single event involving action. SPC (semi-passive coordination) reduces the need of consensus protocol execution by fixing a single primary actor. Still, a consensus is needed to decide which actor should be used to replace the primary actor whenever it breaks down.

The proposed solution used a simplified version of MAMS, where the number of duplications is largely reduced, and consensus step is eliminated. First, a clustering protocol is proposed, which is executed once at the network initialization. It permits to divide sensors into clusters with one actor as clusterhead, then to group each couple of actors able to replace each other into a high level cluster (including the two actors and their members). The large cluster is called self-repairing cluster or SR-cluster, as it is able to automatically replace one of the actors with the other as soon as it breaks down. To ensure this, MAMS is applied, but only within the SR-cluster domain. That is, a sensor reports events to its cluster-head (primary actor), as well as the other actuator of the SR-cluster (secondary actor with respect to this sensor). Comparative simulation study with TOSSIM shows the proposed method considerably reduces the execution latency compared to SPC approach, while keeping fault-tolerance high enough compared to fault-intolerant solutions.

The remaining of the paper is organized as fellows: The related work is presented in the next, followed by the new solution in section 3. Simulation results are presented in Section 4, and finally Section 5 concludes the paper and summarizes the perspectives.

2 Related Work

Fault-tolerance in wireless sensor networks (WSN) have been largely considered by the research community, and several solutions have been proposed. Different approaches have been used, such as information sharing [1], information filtering [2], clustering [3], data checkpointing and recovery methods [4]. Nonetheless, these solutions do not apply directly to WSAN, notably to actors' failure, due to their heterogeneity and the special features of actors in terms of energy, computation and storage capacity, etc. More importantly, actors tend to be deployed in limited number, and tolerating their fault is critical to design reliable applications. The first survey dealing with WSAN and research challenges is [5]. In

[6] the authors propose the use of multi-actor multi-sensor (MAMS) model to ensure fault-tolerance in WSAN. In this model every sensor sends data to several actors, and every actor receives data from several sensors in the event area. This model is obviously more fault-tolerant than the single-actor multi-sensor (SAMS) model. However, for each event a consensus among actors is needed to elect a primary actor that will react upon the event. This requires a costly negotiation step (consensus) to be executed for each actuation event. Semi-passive-coordination (SPC) [7] is an improvement of the basic MAMS model, where only one actor is used as primary, and the others are considered as backups. Sensor-actor communication is done in three phases; Broadcast, decision, and update. A sensor s_i capturing an event e_i submits the collected data towards all the actors. Backup actors forward the data to the primary actor, which is the main responsible for execution of actions related to the event e_i. Once a decision is made by the primary actor, an update message is sent to all backup actors using some group communication protocol [8] [9]. Accordingly, all the backup actors acknowledge the update message. When the primary actor breaks down, a backup actor is elected as a new primary actor using an election algorithm [10]. The new primary actor sends an update message to all backups and waits for receiving all acknowledgements. This technique rises two major problems. The first one is lack of scalability, as a unique actor cannot responde to all events in a large network. The second is the action execution latency when the primary actors breaks down. The proposed solution tackle these issues and proposes a scalable approach that ensures fast substitution of faulty actors.

3 New Solution

3.1 Network Model

We suppose nodes are densely deployed in the event area, enabling availability of multiple routes between any two communicating nodes. Each actor is able to cover a limited area of the sensed region. Number of Actors is supposed high enough to cover the whole sensed region, with enough redundancy on coverage such as every actor can be replaced by at least another one in case of fault. All sensor are assumed to be aware of their direct (one hop) neighbors. To ensure this a simple neighbor discovery protocol can be run at network setup. All nodes are supposed to be synchronized. A synchronization algorithm, like [11] [12], can be used for this end. The proposed solution applies to both senor/actor (SA) model, and sensor/actor/actuation-device (SAA) model. In the first case, actor mobility is needed to replace faulty nodes. The second model is much efficient as it separates the actuation device from the action decision, and eliminates the need of a mechanical movement as long as actuation devices are correctly operating.

3.2 Solution Description

The proposed protocol divides the network into several equal-size self-repairing clusters, where every sensor is associated to a single primary actor then every two

actors are gathered in a higher level cluster called self-repairing cluster or (SR-cluster). The SR-cluster may be considered as the fusion of the two clusters and each actor is considered as secondary clusterhead by members of the other actor's small cluster. A simple MAMS model is used within the SR-cluster, where data are sent to both cluster-heads. As soon as one of the two clusters breaks down, the other one replaces it and executes the action. The use of only two clusters instead of using more eliminates the need of any consensus protocol execution to replace the faulty actor, which accelerates the execution of the waiting actions. The proposed protocol runs in the following four steps.

3.3 Phase 1: Hello Propagation

This phase enables the creation of primary clusters, along with construction of routes towards the primary cluster-head. An actor, CH-source, in an event region W-source, initially broadcasts a HELLO message with a fixed TTL, i.e. the packet will be propagated up to TTL's value hops. The TTL value may depend upon the residual energy of the actor, the number of its neighboring nodes, etc. The HELLO packet carries information about the original actor along with routing information, which is updated on each hop. Two classes of routes are defined; real-time paths (RTP) and low-energy paths (LEP). Routing information carried in the HELLO packet that reflects the energy level and the latency of nodes on the route traversed thus far by the packet are used respectively to update LEP and RTP tables. The metric of the route constructed by a HELLO packet is simply the cumulative cost (energy and delay for LEP and RTP routes respectively). Each free sensor (FS) receiving the HELLO packet becomes a member of the CH-source, or M-source (member of the source cluster). When the HELLO packet reaches a node belonging to another cluster, say CH-destination, it becomes a sensor border, SB, and launches the second step of the protocol to attempt gather CH-source and CH-destination in an SR-cluster. This phase is launched asynchronously by every actor, once at the initialization of the network.

3.4 Phase 2: *SR_REQ*

After receiving a HELLO packet from all its neighboring nodes, or after a timeout from receiving the first HELLO packet, the SB sends an *SR_REQ* packet (Self-repairing cluster construction request) towards its primary actor, CH-destination, through the RTP path. It includes information about the actor originator of the HELLO packet, CH-source. This information is used by the CH-destination to check if the CH-destination can cover CH-source area, which is a vital conditions for constructing an SR-cluster. Then after collecting *SR_REQ* packets from different SB the CH-destination responds accordingly by a positive or negative *SR_REP* towards two SB that it chooses as sensor gateways (SG) if the response is positive. The choice of these gateways depends on the current residual energy of available candidates [13], to provide long-time reliable communication between the two clusters. The response message takes the reverse RTP path towards the two selected SG, which are in charge of launching the third step.

Algorithm 1. Script Describing the Protocol

Initialization
if (Node is Actor) **then**
Broadcast HELLO
end if
When receive HELLO
if (Node is FS) **then**
Calculate RTP and LEP to CH_Source and Update routing Table
if (HELLO.Hop < HELLO.TTL) **then**
HELLO.Hop++
Update and Broadcast HELLO
end if
else
Set node state to SB
Wait For Receiving Hello from all neighbors or timeout
Initialize and Send SR_REQ to CH_dest
end if
When receive SR_REQ
if (Node is Actor) **then**
Select best two SG
if ($SR\_cluster$ construction condition is TRUE) **then**
Initialize and Send positive SR_REP to SG
else
Initialize and Send negative SR_REP to SG
end if
else
Update and Forward SR_REQ to CH_dest
end if
Whene receive CA_REP
if (Node is Sensor) **then**
Update and Forward $HELLO\_REP$ to SG
if (Node is SG) **then**
if (CA_REP is positive) **then**
Node sate = SG
Update and Broadcast positive $HELLO\_REP$
else
Update and Forward negative $HELLO\_REP$ to CH_source
end if
end if
end if
receive $HELLO\_REP$
if (CA_REP is positive) **then**
Calculate RTP and LEP to SG and Update routing Table
if ($HELLO\_REP$.Hop < $HELLO\_REP$.TTL) **then**
$HELLO\_REP$.Hop++
Update and Broadcast $HELLO\_REP$
end if
else
Update and Forward negative $HELLO\_REP$ to CH_Source
end if
When receive data to forward
Switch(data.QoS)
case(MA-RTP): Use RTP to send data to $CHsource$ and $CHdest$
case(SA-RTP): Use RTP to send data to $CHsource$ only
case(MA-LEP): Use LEP to send data to $CHsource$ and $CHdest$
case(SA-LEP): Use LEP to send data to $CHsource$ only

3.5 Phase 3: Route Update

During this phase, if the CA_REP is negative the two SG just transmit in unicast a negative $HELLO\_REP$ to CH-source. This means that CH-destination cannot cover CH-source's event zone, which prevents the construction of the SR-cluster. The actor CH-source may decide to increase the TTL and rebroadcast the $HELLO$ packet to search for another possible backup. This can also be done if the actor does not receive any $HELLO\_REP$, i.e. no SB has been reached. On the other hand, if the CA_REP is positive the two SG broadcast a positive $HELLO\_REP$ with doubled TTL such that to reach sensors of the two clusters and to update entries towards the SG in the sensor's RTP and LEP routing tables.

3.6 Phase 4: Data Transmission

As soon as routing tables of all M-source sensors in W-source are updated, each one would be able to reach CH-destination through the two SB[1]. Four modes are used for data transmission in an SR-cluster, following the required QoS of the data packet. The first mode is multi-actor real-time path (MA-RTP), where sensors send data to both actors using RTP routing. This mode is the most reliable and delay-efficient, and it may be used for critical data where reaction time is required to be minimal. In this case, the backup actor may react to the event if no ACK of action execution is received from the responsible actor. Substitution is then performed rapidly. The second mode is multi-actor low-energy path (MA-LEP), where data are sent to both actors but using LEP routing. This mode also ensures a fault-tolerance but with possible small extra delay for the sake of saving energy. It can be used to send data related to events where reaction is critical but not necessary in realtime. The remaining modes are single-actor real-time path (SA-RTP) and single-actor low-power path (SA-LEP). They use only one actor and may be used for real-time traffic and regular traffic (respectively) that may tolerate non-execution of action. The protocol is illustrated in Algorithm 1.

4 Simulation Study

The proposed protocol has been compared by simulation using TOSSIM [14] with the SPC approach (SPC-like protocol) and a basic protocol with single actuator for each region (SA), which does not provide any fault-tolerance. Two metrics have been considered in scenarios with faulty actors: i) efficiency in executing actions (success rate), which is the ratio between the number of executed actions vs. the total tasks (that rise actions) launched, ii) the execution delay (of successful actions), as the time separating the detection of the event (that rises an action) and the execution of the action. The protocols has been evaluated in configurations with different error rates (the rate of faulty actors), and different

[1] This is identical for sensors of the other cluster.

levels of network size (scalability). Each point of the plots presented hereafter is the average of 10 measurements, with 95% of confidence interval. Figures 1 and 2 show the performance metrics vs. error rates. In each execution, every actor's state is randomly set to faulty with probability equals to the appropriate error rate. A grid topology of 150 uniformly distributed nodes ($10 * 15$) has been used, among them 10 equally distant nodes have been configured as actors. The TTL value has been set to 5.

Success rate of the proposed protocol (SR-cluster) presented in Figure 1 is not much affected by the increase of error rate, and kept above 88%. The difference between SR-cluster and SPC-like is minor compared to the difference with SA; it does not exceed 8%, whereas the difference between SR-cluster and SA varies between 10% and 38%. SPC-like uses all actors as potential substitute of faulty actors, while in SR-cluster each actor may be replaced only with one actor (secondary cluster-head of the SR cluster). Trivially, the probability that all actors are faulty is less than the probability that two clusters are so, which justifies the superiority of SPC-like and the small difference vs. SR-cluster. However, the cost of the highest fault-tolerance provided by SPC-like is a very high latency, Figure 2. SA ensures a stable and the lowest delay. The delay of SR-cluster is inevitably higher than SA, and smoothly increases with the error rate. The difference between the two protocols is due to the delay of executing actions requiring actor substitution (in case of failure of primary actors), which do not occur for SA that does not ensure any tolerance. i.e. In case of failure, SA just ignores the action and thus no delay is accounted. Substitution delay of SR-cluster is limited to a timeout for ACK reception at the secondary actor, upon which the replacement procedure is immediately launched. Nonetheless, for SPC-like this delay involves a delay of consensus protocol execution between all actors to elect a substitute. The latter is considerably affected by the error rate (causing increase of number of substitutions). This justifies the highest delay of SPC-like and the dramatic increase. The difference between SR-cluster and SA is around 1 sec, while the difference between SR-cluster and and SPC-like reaches almost 2.7 sec, i.e 60% reduction for SR-cluster over SPC-like.

Figures 3, 4 show the performance metrics in scenarios of different sizes, where the number of nodes has been varied from 25 (grid of $5*5$) to 300 (grid of $15*20$). The number of actors has been varied between 2 to 15 (2, 4, 6, 10, 12, 15 for each grid respectively), and the actors have been uniformly distributed within the grid. The error rate has been set to 40%. We remark that plots of Figure 3 has the same shape as in Figure 1, except a stable but still low success rate for SA. The same can be realized for the delay metric (Figure 4), with the exception of linear increase for SA, which still has the lowest delay. Increase and decrease of the delay and success rate respectively for all protocols, are due to the increase of the network size. This increase inevitably rises the number of hops in routes, which rises the delay. It also rises collisions and thus reduces the success rate. The two figures illustrate that the proposed protocol scale well in balancing the success rate and the latency.

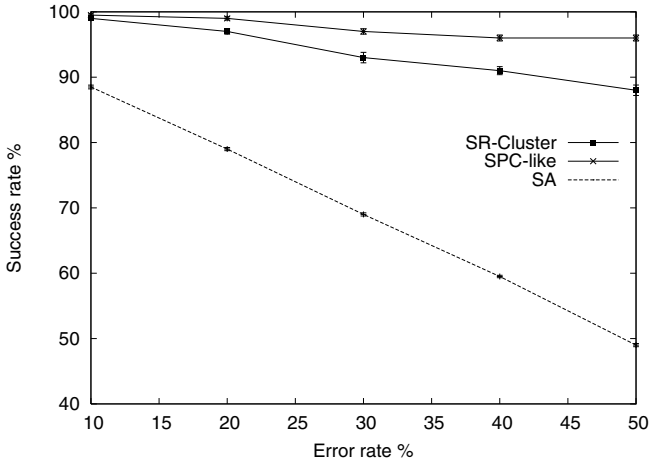

Fig. 1. Success rate vs Error rate

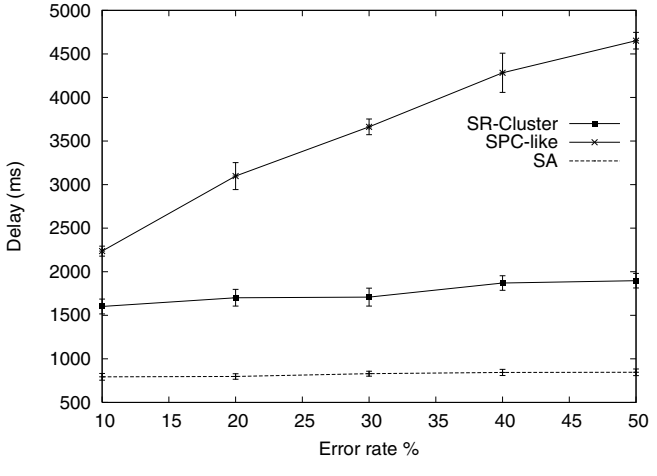

Fig. 2. Delay vs Error rate

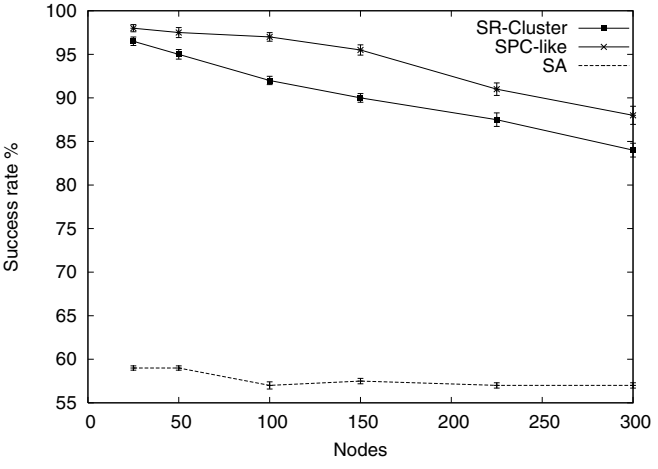

Fig. 3. Success Rate vs Number of nodes

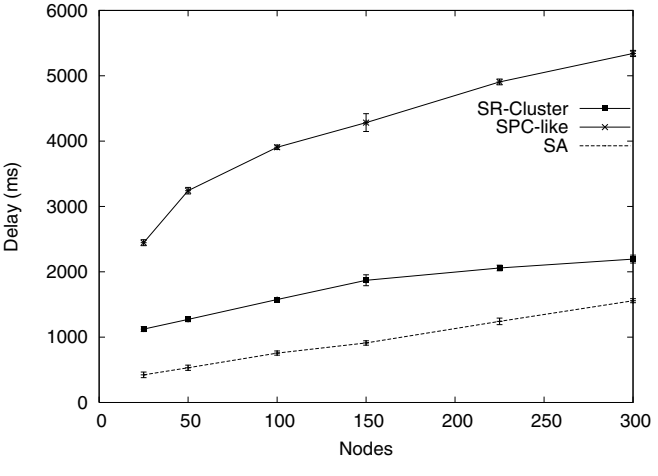

Fig. 4. Delay vs Number of nodes

5 Conslusion

A new delay-efficient fault-tolerant solution has been proposed, which considers actor faults. The solution relies on a two-level hierarchical clustering, and the use of a simplified MAMS (multiple-actor multiple-sensor) communication model. It includes a simple clustering protocol that runs once at the network initialization. It first divides nodes into equal-size clusters with one actor as cluster-head. After that, each two-clusters are gathered in a higher level of hierarchy cluster, called SR-cluster (self-repairing cluster). This cluster ensures self-repairing as it allows each actor to automatically replace the other as soon as the later breaks down. To provide this, events of the two clusters are duplicated but only towards the two actors (simplified MAMS). Limiting the number of actors in the SR-cluster to two eliminates the need of any consensus protocol running step required by the current actor-fault-tolerant solutions, namely SPC (semi-passive coordination), and the basic MAMS. Simulation results carried out using TOSSIM show the proposed protocol (SR-cluster) ensures a fault-tolerance very close to that of SPC, while considerably decreasing delays in executing actions (up to 59%). The cost of this delay reduction is inevitably a minor decrease in fault-tolerance, but the difference does not exceed 8%. Compared to a fault-intolerant protocol with single actor (SA), both protocols (SR-cluster and SPC) provide much higher fault-tolerance. SR-cluster provides from 10% to %38 more performance than SA, which is by far higher than the difference between SR-cluster and SPC. The proposed protocol is thus very appropriate for realtime applications. Furthermore, eliminating the large duplication towards every actor as well as the consensus protocol execution upon each actor failure would be power-efficient (compared to SPC and the basic MAMS). Investigating this issue by measuring some energy metrics represents a perspective to this work. Mathematical analysis of the solution is also in the perspectives.

References

1. Clouqueur, T., Saluja, K.K., Ramanathan, P.: Fault tolerance in collaborative sensor networks for target detection. IEEE Trans. Comput. 53(3), 320–333 (2004)
2. Ding, M., Liu, F., Thaeler, A., Chen, D., Cheng, X.: Fault-tolerant target localization in sensor networks. EURASIP J. Wirel. Commun. Netw. 2007(1), 19–19 (2007)
3. Gupta, G., Younis, M.: Fault-tolerant clustering of wireless sensor networks. In: IEEE Wireless Communications and Networking, 2003, WCNC 2003, pp. 1579–1584 (2003)
4. Salehy, I., Eltoweissy, M., Agbariax, A., El-Sayedz, H.: A fault tolerance management framework for wireless sensor networks. Journal of Communications 2(4) (2007)
5. Akyildiz, I.F., Kasimoglu, I.H.: Wireless sensor and actor networks: research challenges. Ad Hoc Networks 2(4), 351–367 (2004)
6. Ozaki, K., Kenichi, W., Satoshi, I., Naohiro, H., Tomoya, E.: A fault-tolerant model for wireless sensor-actor system. In: 20th IEEE International Conference on Advanced Information Networking and Applications (AINA 2006), IEEE Digital Library (2006)

7. Ozaki, K., Watanabe, K., Enokido, T., Takizawa, M.: A fault-tolerant model of wireless sensor-actuator network. Int. Journal of Distributed Sensor Networks 4(2), 110–128 (2008)
8. Schiper, A., Birman, K., Stephenson, P.: Lightweight causal and atomic group multicast. ACM Trans. Comput. Syst. 9(3), 272–314 (1991)
9. Nakamura, A., Takizawa, M.: Causally ordering broadcast protocol. In: The 14th IEEE International Conference on Distributed Computing Systems (ICDSC), pp. 48–55 (1994)
10. Nikano, K., Olariu, S.: Uniform leader election protocols for radio networks. IEEE Trans. Parallel Distrib. Syst. 13(5), 516–526 (2002)
11. Boukerche, A., Martirosyan, A.: An efficient algorithm for preserving events' temporal relationships in wireless sensor actor networks. In: Proceedings of the 32nd IEEE Conference on Local Computer Networks, LCN 2007, pp. 771–780. IEEE Computer Society, Washington (2007)
12. Ganeriwal, S., Tsigkogiannis, I., Shim, H., Tsiatsis, V., Srivastava, M.B., Ganesan, D.: Estimating clock uncertainty for efficient duty-cycling in sensor networks. IEEE/ACM Trans. Netw. 17(3), 843–856 (2009)
13. Djenouri, D., Badache, N.: An energy efficient routing protocol for mobile ad hoc network. In: The second IFIP Mediterranean Workshop on Ad-Hoc Networks, MedHoc-Nets 2003, Mahdia, Tunisia, pp. 113–122 (June 2003)
14. Levis, P., Lee, N., Welsh, M., Culler, D.: Tossim: accurate and scalable simulation of entire tinyos applications. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys 2003, pp. 126–137. ACM, New York (2003)