

# Enhanced Master-Slave Time Synchronization Architecture for Wireless Sensor Networks\*

Francis Otto, Drake P. Mirembe, Elizabeth Olule and Song Ouyang

**Abstract**—A number of time synchronization protocols for Wireless Sensor Networks rely on a single (master) node as source of reference time by which other (slave) nodes can be synchronized. Although the amount of overhead is greatly reduced, this is not a reliable technique since the master’s availability is not guaranteed. In this paper we address the issue of dependability of the master node in a master-slave based time synchronization technique. We explore, as a security measure, the use of a second node as a standby node that can takeover the role of the default master in case of its failure. Our proposed architecture provides for automatic master node failure detection and a mechanism that enables the second master to take over the role before the rest of the nodes on the network are affected by the failure.

**Index Terms**—Architecture, Master Node, Security, Time Synchronization, Wireless Sensor Network

## I. INTRODUCTION

IN recent years, tremendous technological advances have led to the development of low-cost sensors, which are capable of wireless communication and data processing [1, 2, 3, 4, 5]. It is predicted that in the near future sensors will be everywhere, and sensing virtually everything.

WSNs are used to monitor real-world phenomena. For such monitoring applications, physical time often plays a crucial role. Providing synchronized physical time is a complex task due to various challenging characteristics of WSNs. Recently a lot of research has been conducted to address this problem. As a result many clock synchronization protocols have been proposed [7] [8] [9] [10] [11] [12] [13] [15] [16]. In this paper we classify the existing clock synchronization protocols for WSNs into two categories basing on the role of the node in synchronization process (see Table 1).

In *Peer-to-Peer* design any node can communicate directly with every other node in the network in order to synchronize

its clock. This design eliminates the risk of master node failure, which would prevent further synchronization, and offers more flexibility. However they are more difficult to control. Besides there is relatively high network traffic overhead and consequently high energy consumption compared to master-slave design. This makes it unsuitable option considering the peculiar characteristics of WSNs.

In a Master-Slave design one node is assigned as the master and the other nodes as slaves. The slave nodes consider the local clock reading of the master as the reference time and attempt to synchronize with the master. In general, the master node requires CPU resources proportional to the number of

TABLE I  
CLASSIFICATION OF TIME SYNCHRONIZATION PROTOCOLS BASED ON ROLE

Protocol	Category
RBS [7]	Peer-to-peer
Romer’s [10]	Peer-to-peer
PalChaudhuri et al. [11]	Peer-to-peer
Sichitiu and Veerarittiphan [15]	Peer-to-peer
Time-diffusion sync [12]	Peer-to-peer
Asynchronous diffusion [13]	Peer-to-peer
Continuous Clock Sync [8]	Master-Slave
Timing-Sync [16]	Master-Slave
Delay Measurement Time Sync [9]	Master-Slave
RBS [7]	Peer-to-peer
Romer’s [10]	Peer-to-peer
PalChaudhuri et al. [11]	Peer-to-peer
Sichitiu and Veerarittiphan [15]	Peer-to-peer
Time-diffusion sync [12]	Peer-to-peer
Asynchronous diffusion [13]	Peer-to-peer

slaves, and nodes with powerful processors or lighter loads are assigned to be the master node.

The objective of our project is to design a time synchronization protocol based on *enhanced* master-slave architecture to ensure a more secured and hence dependable protocol. We note that although the amount of overhead is greatly reduced in a master slave design, the current protocols based on master-slave design are not reliable since there are no clear mechanisms that provide for reliable and immediate replacement of a master node in case it fails. We propose security enhancement by introducing the concept of resort node as a backup of master node so that in the presence of a failed master node, the network synchronization is not discontinued. Our enhanced master-slave time synchronization architecture makes it difficult for the entire

Francis Otto, Elizabeth Olule and Prof. Song Ouyang are with the School of Information Science and Engineering, Central South University, Changsha, Hunan, P.R. China 410083. Contacts: Emails; [ottofrancis@yahoo.com](mailto:ottofrancis@yahoo.com), [olule\\_e@yahoo.com](mailto:olule_e@yahoo.com). Tel : +86 13975112449, +86 13975837010

Drake P. Mirembe was with Radboud University Nijmegen, Netherlands. He is now with the Department of Networks, Faculty of Computing & IT, Makerere University, P.O.Box 7062 Kampala, Uganda, East Africa.. Contacts: Email [dmirembe@cit.mak.ac.ug](mailto:dmirembe@cit.mak.ac.ug).

\* This paper is of an ongoing project to build a fault-tolerant time synchronisation protocol for wsn based on master-slave architecture.

network to notice the failure of the master node.

A resort master node is a node or base station with similar characteristics as the currently active master node. It only plays master node roles when it detects a default master node failure or when the resources of the default master are depleted. This increases the availability of master node in spite of the presence of failed master node.

## II. RELATED WORK

Time synchronization is a research area with a very long history. Over time, numerous algorithms have been proposed and have been in large-scale use. The Network Time Protocol (NTP) [6] is perhaps one of the most advanced and time-tested systems.

NTP, a mater-slave protocol, was designed for large-scale networks with a rather static topology (such as the Internet). We point out that NTP is not suitable for WSNs particularly due to its huge energy demand.

Reference Broadcasting Synchronization (RBS), proposed by Elson [7], is one of the protocols introduced in consideration of the characteristics of WSNs. However RBS, a peer-to-peer design, uses too much broadcast- hence more traffic is generated.

Despite many advantages of RBS, it was noticed [9] that there is relatively high network traffic overhead and consequently high energy consumption in RBS. The disadvantage is that the number of message exchanges is high. For a single hop WSN of  $n$  nodes, it needs at least  $n$  messages to be exchanged between the nodes.

Delay Measurement Time Synchronization (DMTS) was proposed in [9]. DMTS is based on the estimation of all delays involved in time synchronization message transfer path. In DMTS a leader is selected as time master and broadcasts its time. All the receiver devices measure the time delay and set their time as received master time plus measured time transfer delay. The challenge here is that anything can go wrong with the master node that provides a reference time. The master node may fail. Knowing the limitations of the sensor nodes it is not practically reliable to select any node to be a time leader to replace the functions of a master node.

Although the current literature shows tremendous effort towards finding a suitable technique for WSNs time synchronization, little emphasis has been placed on the security aspect of it. When it comes to security, master-slave design is most vulnerable. Our approach is different from the previous researchers in that we are looking at securing time synchronization process. Realizing the security vulnerability of DMTS and other master-slave designs, in this paper, we thus propose a modification to the master-slave architecture as an enhancement of security for the master node in the terms of dependability.

## III. ENHANCED MASTER-SLAVE TIME SYNCHRONIZATION

### A. Our Framework

Our design emphasizes reliability and fault tolerance in mater-slave architecture. We provide a resort node as a back-up of the master node. Successful time synchronization design should consider the peculiar characteristics of WSN at its core. Most importantly, in our case, is the fact that base-stations as well as the sensor nodes are exposed to numerous situations which may lead to failure. Therefore, we have focused on the reliability, which means that the protocol stands in spite of the possibility of master node failing.

In developing our design we focus on two scenarios: the first scenario involves a case when the master node performance is degraded for example when its resources are at critical level and the node cannot continue to perform the synchronization process with the required level of efficiency. The second scenario focuses on how to handle unplanned failure of the master node. In both cases our design focuses on providing a more reliable protocol, which stands the graceful degradation in the presence of a failed master node. Therefore we provide the following:

- Resort master; to back up the master by staying alert so that in case of failure of the master, it provides an automatic solution.
- Communication between the two synchronization nodes (master and resort); an effective regular communication between the two will make any failure to be detected quickly.
- Master selection algorithm; for choosing a suitable node to become master or a resort.
- Resource analyzer; used for gathering and analyzing resources available in the nodes on the network. On this basis master selection algorithm can determine which node

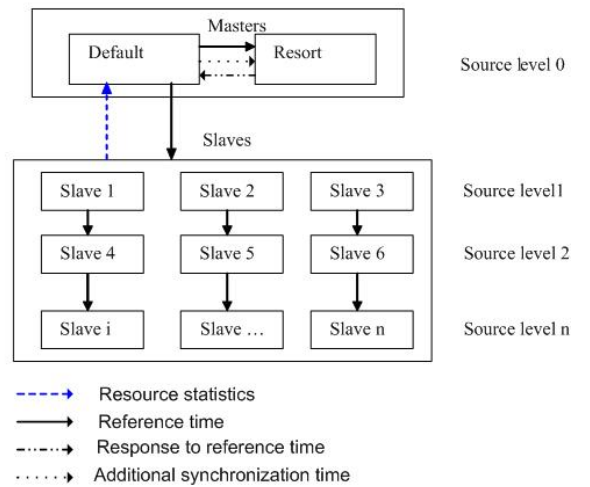


Fig. 1. Enhanced master-slave architecture

is suitable to become a master or a resort.

## B. Enhanced Master-Slave Time Synchronization Architecture

*Remark 1.* Time source level refers to how far the connection between the slave node and the master node. In other words how many nodes through which a slave gets the reference time. Source level 0 means the slave connects to the master directly. If a node connects to the master node through  $n$  nodes then its time source level is Source level  $n$ .

## C. Synchronization Nodes

Our protocol uses dual master nodes technique of securing a master-slave time synchronization so that in case a reference time transmitting (master) node fails, there is a suitable (resort) node that comes into play. The two synchronization/master nodes in our architecture are Default master and Resort.

### 1) Default Master Node

When the network is up, the first task of the default master is to broadcast its local clock reading as a reference time in a time message and it is sent to all directly connected nodes. After broadcasting the reference time, the default master receives the resource statistics from all directly connected nodes and runs the master node selection algorithm in order to find the first resort master node. Neighboring nodes update their resource statistics with the default master whenever there are significant changes in their resource statistics. The level of significance can be set in order to minimize communication between the nodes and master. The default master node does not copy these resource statistics to the resort master and in turn the resort node does not send its statistics because it is not considered when running the algorithm.

Successive master and resort nodes are obtained by running the master selection algorithm upon failure of either the master or the resort nodes. The Default master node is the only node active in broadcasting the reference time to the network.

### 2) Resort master node

A resort master node is a node with similar characteristics as the currently active master node. The resort master node only switches to the master node role when it detects a default master node failure. This switch increases the availability of the reference time in the presence of a failed master node. The resort master node does not broadcast reference time but it continues to receive additional communication from the default master node in the form of extra time synchronization messages. The communication is regular and frequent so that the failure of default master can be noticed faster.

## D. Communication between Master Nodes

The master nodes have the capability of communicating with each other. An effective and regular communication between the two master nodes will make any failure to be detected quickly and responded to accordingly.

The resort master node which is usually one hop away receives additional communication from the default master

node in the form of time synchronization messages. The additional time synchronization messages enable the resort node to determine that the master node is still alive and these messages can be made as small as possible to minimize energy consumption. The resort node does not respond to the additional time synchronization messages.

In order for the master node to determine the state of the resort node, when the resort node receives the general synchronization messages from the master it responds to the master node and it informs the master node that it is alive. This messaging system of additional time synchronization messages and general synchronization messages ensures that minimum energy is spent in communication between the master and resort. Since the failure of the master node would cause more interference than the failure of the resort, more energy is allocated to checking on the status of the master.

If the master broadcasts the reference time at time  $T$ , then the additional synchronization messages are broadcast to the resort node at time  $T/x$  where  $x > 1$  and  $T \bmod x = 0$ . This will ensure that the resort node is informed of the default node failure before the time for the next broadcast elapses.

We proceed to discuss the four scenarios under which the master node selection algorithm is run.

### 1) Master node resources critical

The first case is when the master node resources fall below a given threshold e.g. the power level is declining or the node performance is below optimum. Once this occurs, the master node informs the resort node about its resource status and waits to send the last synchronization message at time  $T/x$ . On receiving the low resource message from the master node, the resort node waits to receive the last synchronization message from the master node at  $T/x$  then switches to the master node status. After sending the last synchronization message, the old master node removes itself as the master. At time  $t=T$ , the new master node sends out its first synchronization message together with its ID. On receiving the message with a new ID, the nodes directly connect to the new master send their resource statistics to the new master and this node runs the node selection algorithm to determine a new resort node.

### 2) Resort Node Resources Critical

In the second case, when the resort node resources fall below a given threshold, it sends a message to the default master and then removes itself as the resort node. The default master node runs the master node selection algorithm on receiving the message from the resort node and selects a new resort node.

### 3) Master Node Failure

The master node sends additional synchronization messages to the resort node in order to allow the resort node to determine that the master node is alive. When the resort node does not receive a synchronization message at the expected time, it can do one of two things:

- i. If the end of the next time period will coincide with a general synchronization message then the resort will immediately set itself to master, send a message to

the old master informing it that the resort is now the active master and runs the master selection algorithm after it has sent the first synchronization message and received the resource statistics from its neighbors.

- ii. If the end of the next time period will not coincide with a general synchronization message then the resort waits to see if the next expected synchronization message also does not appear. If it doesn't arrive then the resort then sets it self to master and follows the procedure mentioned above

#### 4) Resort Node Failure

The resort node responds to the general synchronization messages that are sent by the master node. If the master node does not receive a response from the resort node after a cycle of two consecutive general synchronization messages then the master assumes the resort has failed. The master then runs the master selection algorithm and uses the stored resource statistics to determine the next resort node.

In the case where a node failure is temporary, if the time between the node failure and recovery is longer than the periods mentioned in previous two subsections for the master and resort nodes respectively i.e. the node failures were detected, then the node should follow the behavior of a new node in the network i.e. wait for the next synchronization message then send its resource statistics to the master node. If the node failure was not noticed i.e. less than the periods mentioned in subsection 3 and 4 above then the node should continue operating as the master or resort.

#### E. Master Selection Algorithms (MSA)

This algorithm is the core of our argument. In order to maintain two master nodes in the network, Master Node Selection Algorithm is run as a response to a failed master node. The objective of the algorithm is to identify the most suitable node in the network to become a master node.

MSA is initiated by one of the master nodes in event that the other is off. Several factors are considered by the algorithm in selecting a master node such as energy supply, memory, processing power and processing load. These are considered important resource statistics that guide the algorithm in choosing the candidate master node. Before master selection, the 1-hop neighbors of the master node provide their resource statistics and by comparison, each node is therefore probed in order to determine which node has the most optimal amount of resources. In case no node on level 1 has the minimum required resources, nodes of the level 2 will be probed. If there is a node on level 2 that meets this requirement then it will be selected and moved to level 1.

The pseudo code

```
MSA (Default, resort, Slaves)
  If (master fails)
    Set resort node = Default node;
  For all level 1 nodes,
    Compute resource statistics
```

```
End for
Best statistics(from level1)
If (Best statistics < threshold)
  Re-distribute  $S_i$  roles to
```

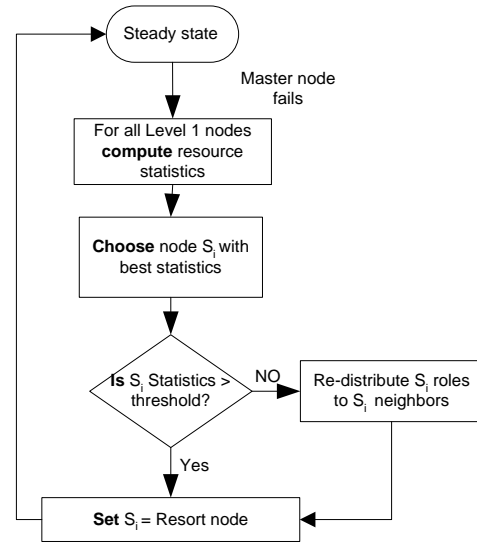


Fig. 2. Master Selection Algorithm (MSA)

```
S_i neighbours;
Set S_i = resort node
End if
End if
End
```

#### F. Slave Node

Slave is a node which has the following roles: (a) receive reference time broadcasts, (b) perform the required synchronization calculations such as delay measurement along the time message path as proposed in [9], (c) adjust its local logical clock reading and (d) provide its resource statistics to the master node when the master node selection algorithm is run. Each slave receives reference time from the master node either directly or through parent and corrects its local clocks to be synchronized with the master clock. Whenever a new node joins the network, if the node is on level 1 then on receiving the first synchronization message, it should forward its resource statistics to the master.

#### IV. CONCLUSION AND FUTURE WORK

Most of the existing time synchronization protocols for Wireless Sensor Networks have been designed to address a number of concerns arising from peculiar nature of WSN which makes the traditional time synchronization not suitable. Design objectives of most of the protocols have been focused on; Energy efficiency, infrastructure, end-to-end latency, message loss / delivery, and network dynamics. However these designs do not emphasize security concerns particularly in the master-slave design, which relies on a single node to provide a reference time. The problem is that when the master node is fails the whole network would be left in a state of

unsynchronized clocks.

Our main objective is to address the availability issue of master node by providing a resort node that will provide reference time for the rest of the network slave nodes in event that the default master node fails.

As part of our on-going project, we shall continue, by way of experiments, to establish the most suitable way to enhance the master-slave based time synchronization for WSNs. We will also focus on analyses to compare our architecture with other approaches. We thus intend to continue exploring in details the areas of security and efficiency of this design. We shall also look at energy efficiency as well as efficiency in terms of precision.

#### REFERENCES

- [1] J. Hill, M. Horton, R. Kling, L. Krishnamurthy. The Platforms Enabling Wireless Sensor Networks. *Communications of the ACM*, 47(6):41–46, June 2004. W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.
- [2] Improving Life and Industry with Wireless Sensors. Intel Corporation available at ([http://www.intel.com/research/exploratory/deep\\_networking.htm](http://www.intel.com/research/exploratory/deep_networking.htm)), page accessed May 2006.
- [3] SmartDust. Autonomous Sensing and Communication in a Cubic Millimeter. Available at <http://robotics.eecs.berkeley.edu/~pister/SmartDust/>
- [4] Dust Networks. Available at <http://www.dustnetworks.com/flash-index.shtml>, page accessed May 2006.
- [5] TinyOS. An Operating System for Networked Sensors. Available at <http://www.tinyos.net>, page accessed May 2006.
- [6] David L. Mills. Internet time synchronization: The network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, October 1991.
- [7] J. Elson, L. Girod, and D. Estrin. Fine-Grained Network Time Synchronization using Reference Broadcasts. *Proc. Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Vol 36, pp. 147–163, 2002.
- [8] M. Mock, R. Frings, E. Nett, and S. Trikaliotis. Continuous Clock Synchronization in Wireless Real-time Applications. *Proc. 19th IEEE Symposium on Reliable Distributed Systems (SRDS-00)*, pp. 125–133, Oct. 2000.
- [9] S. Ping. Delay Measurement Time Synchronization for Wireless Sensor Networks. *Intel Research, IRB-TR-03-013*, June 2003.
- [10] K. Romer. Time Synchronization in Ad Hoc Networks. *Proc. ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '01)*, pp. 173–182, Oct. 2001.
- [11] S. PalChaudhuri, A. Saha, and D. B. Johnson. Probabilistic Clock Synchronization Service in Sensor Networks. Technical Report TR 03-418, Department of Computer Science, Rice University, 2003.
- [12] W. Su, I. Akyildiz, Time-Diffusion Synchronization Protocols for Sensor Networks, *IEEE/ACM Transactions on Networking*, 2005, in press.
- [13] Q. Li and D. Rus. Global Clock Synchronization in Sensor Networks, *Proc. IEEE Conf. Computer Communications (INFOCOM 2004)*, Vol. 1, pp. 564–574, Hong Kong, China, Mar. 2004.
- [14] K. Arvind. Probabilistic Clock Synchronization in Distributed Systems. *IEEE Transactions on Parallel and Distributed Systems*, 5(5):474–487, May 1994.
- [15] M. L. Sichitiu and C. Veerarittiphan. Simple, Accurate Time Synchronization for Wireless Sensor Networks *Proc. IEEE Wireless Communications and Networking Conference (WCNC 2003)*, pp. 1266–1273, 2003.
- [16] S. Ganeriwal, R. Kumar, and M. Srivastava. Timing-Sync Protocol for Sensor Networks. *Proc. First Int. Conf. on Embedded Networked Sensor Systems*, Los Angeles, California, Nov. 2003.
- [17] J. Hill and D. Culler. Wireless embedded sensor architecture for system-level optimization. Technical report, U.C. Berkeley, 2001.