

# Cluster Based Hierarchical Wireless Sensor Networks (CHWSN) and Time Synchronization in CHWSN

Poonam Yadav  
IIIT, Allahabad, India  
yadav\_poonam@is.iita.ac.in

Nagesh Yadav  
IIIT, Allahabad, India  
nagesh@is.iita.ac.in

Shirshu Varma  
IIIT, Allahabad, India  
shirshu@iita.ac.in

**Abstract**—A Cluster Based Hierarchical Wireless Sensor Network Architecture is proposed to facilitate more than one application sharing the whole or a part of a wireless sensor network, where each application may have its own network, processing requirements and protocols. Such a network is divided into clusters and the clusters are organized hierarchically. For synchronizing the CHWSN there is the requirement for a cluster based hierarchical time synchronization algorithm. None of the already existing time synchronization algorithms satisfy the needs of time synchronization in our CHWSN architecture. Thus the existing time synchronization algorithms TPSN (Time synchronization Protocol for Sensor Networks) and FTSP (Flooding Time Synchronization Protocol) are modified to fulfill the needs of time synchronization in CHWSN Architecture and developed the Cluster Based Hierarchical, Flooding Time Synchronization algorithm (CBH-FTS). It is a hybrid algorithm, where instead of flooding the synchronization messages to neighbor's node, the root node multicasts the time-sync message to selected cluster-heads using the relevant semantics. Hierarchy of cluster-heads could transmit the synchronization messages down the hierarchy of cluster-heads thus synchronizing only the required part of the network associated with an application. The synchronization could be a result of a decision at root node or could be a result of a request for synchronization from a node to a cluster-head. The CBH-FTS Protocol is semantic driven and covers multiple levels in hierarchy.

## I. INTRODUCTION

Advances in technology enabled researchers in the field of wireless sensor networks to forecast unprecedented growth of ubiquitous applications. However, anticipated extensive use of these sensor networks has not taken place, partly due to high deployment costs coupled with inflexible designs [1 and 16]. Assumptions such as, homogeneity in sensor hardware and software components, and diversity of application requirements, H/W limitations need for homogeneous network environments, and highly specific and inextensible designs, have resulted in the inability to accommodate dynamic applications and addition of new applications over the same network [2 and 17].

Designing sensor networks that are deployed once but would respond to 'yet-to-be identified' applications, dynamically changing network topologies and that scale in multiple dimensions while adhering to stringent resource constraints is a challenging task. The authors proposed a novel architecture for sensor networks called Deploy Once Multiple Application

systems(DOMA) [1]. The DOMA based sensor network architecture characterized by flexible, semantic driven, multiple hierarchical levels of clusters of nodes is described elsewhere [18 and 19].

The multiple dimensions include multiple modalities of components at the sensor, node, network and application levels that are dispersed over a vast spatially and temporally distributed region. Resource constraints include energy, bandwidth and processing limitations of each component.

DOMA supports multiple applications through the formation of virtual networks of hierarchy of components with varying capabilities and functionalities in a dynamic and autonomous manner. The network components could be added or removed logically and physically from the network. DOMA achieves scalability in a resource efficient manner by reducing information load at each level and in each component of the system through the generation of relevant context driven semantics [1].

Context driven semantics is the ability to utilize semantics associated with a context to affect system behavior [1].

The architecture of the hierarchical, cluster based, flexible network is shown in figure 1. Here more than one application can share the whole or part of the network. Each application may have its own network, processing requirements and protocols. We assumed that whole network is divided into clusters and there is hierarchy of clusters.

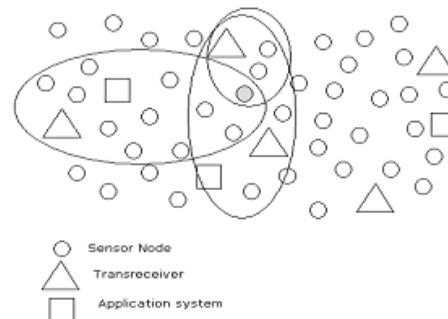


Fig. 1. Shared hierarchical cluster based wireless sensor network [1]

Some of the important assumptions made in the architecture

for flexible, shared, hierarchical, cluster based wireless sensor network are as under:

- 1) Root node is predefined.
- 2) Every node in the network is aware of the immediate cluster-head.
- 3) The cluster-heads at some level know their own parent cluster-head.
- 4) Cluster-heads might be sensing nodes or simple transceivers.
- 5) Transceivers could be fixed, and know their position. They have no energy limitation and could be maintained.
- 6) Non-transceiver cluster-heads are simple nodes that could be assigned the job of cluster-head and are replaceable by another node in case of failure. The cluster-heads are selected dynamically based on energy level. The cluster-heads are position unaware and have no fixed position.
- 7) Hierarchy of cluster-heads is assumed as shown in figure 2.

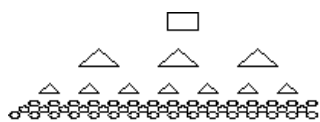


Fig. 2. Hierarchical Architecture

### Time Synchronization

Time synchronization is very critical requirement at all four layers of proposed wireless sensor network architecture[1] namely, Application layer, Network layer, Node layer, Sensors layer. At application Level, time synchronization is required for consistent distributed sensing and control. Distributed data logging depends on the global time of the whole network. Time synchronization at application level could be maintained by already existing Network Timing Protocols (NTP) as used in wired networks[3]. In this paper we will address the problem of time synchronization at network level. At this level synchronization is required to obtain data consistency and coordination. The existing protocols, at this level are TPSN[4], FTSP[5], LTS(Lightweight Time Synchronization)[6], RBS (Reference Broadcast Synchronization)[7], S-Mac and B-Mac layer time synchronization, Adaptive time synchronization[8], Multihop synchronization[9], Post facto synchronization[10] and Optimal time synchronization[11]. All time synchronization protocols are categorized on the basis of local synchronization and global synchronization. On the basis of applications, different models for time synchronization have been studied like virtual clocks, internal synchronization, external synchronization and hybrid synchronization. The two approaches are generally used in time synchronization algorithms: Sender-Receiver synchronization and Receiver-Receiver synchronization.

In RBS, sender broadcasts the reference message. The

receivers receive their local time when receiving the reference message and exchange their local recorded time's with each other. The main advantage of RBS is that it eliminates sender side non-determinism. The disadvantage of this approach is that additional message exchange is necessary to communicate the local time stamps between the nodes[7]. In TPSN algorithm first creates the spanning tree of the network. Each node gets synchronized by exchanging two synchronization messages with its reference node one level higher in the hierarchy[4]. The TPSN achieves two times better performance than RBS by time stamping the radio messages in Medium Access Control(MAC) layer of the radio stack and by relaying two way message exchanges[10]. The shortcoming of TPSN is that it does not estimate the clock drift of nodes, which limits its accuracy, and does not handle dynamic topology changes. LTS(Light weight Time synchronization) was proposed for the applications where the required time accuracy is relatively low. The pairwise synchronization on LTS is similar to TPSN except for the treatment of uncertainty(LTS adopts a statistical model for handling the errors)[6]. The simulation results show that the accuracy of LTS is about 0.5 seconds[6].

In post-facto synchronization, node's clocks are normally unsynchronized. When a stimulus arrives, each node records the time of the stimulus with respect to its own local clock. One node broadcasts a synchronization pulse(beacon) to all nodes in the area using its radio. Nodes that receive this pulse use it as an instantaneous time reference and can normalize their stimulus timestamps with respect to that reference[10].

The FTSP synchronize the local clocks of the network nodes. The FTSP synchronizes the time of the sender to possibly multiple receivers utilizing a single radio message time-stamped at both the sender and the receiver sides. FTSP provides multihop synchronization, the root of the network - a single, dynamically elected node - maintains the global time and all other node - maintains the global time and all other nodes synchronize their clocks to that of the root[5]. The nodes form an ad-hoc structure to transfer the global time from the root to all the nodes, as opposed to a fixed spanning-tree based approach proposed in TPSN. This saves the initial phase of establishing the tree and is more robust against node and link failures and dynamic topology changes. Average per-hop synchronization error in FTSP is less as compared to TPSN and RBS.

## II. CLUSTER BASED HIERARCHICAL, FLOODING TIME SYNCHRONIZATION PROTOCOL (CBH-FTSP)

The proposed cluster based hierarchical, flooding time synchronization algorithm is hybrid algorithm, combining the features of TPSN and FTSP. Instead of flooding the synchronization messages to neighbor's node, the root node multicast the time-sync message to selected cluster-heads. A hierarchy of multilevel, cluster-heads are considered here. Cluster-heads are recognized by their level like level1 cluster-head, level2 cluster-heads, etc. So root node is at level0, sends time-sync message to level1 only, now cluster-heads at level1 exchange

time-sync message to set local clock skew. Once the cluster-heads at level1 are being synchronized, these will broadcast the time-sync message to node cluster-head nodes in their cluster and also multicast the time-sync message to the cluster-heads of level2 and so on. So after a finite time period whole network is synchronized with the root node. The issue to be addressed here is that, it can possible the root node can mislead all of the nodes lying in its subtree like in TPSN, the possible solution provided is all the cluster-heads must exchange time-sync messages between them. The pseudo code for CBH-FTSP is described in Appendix A.

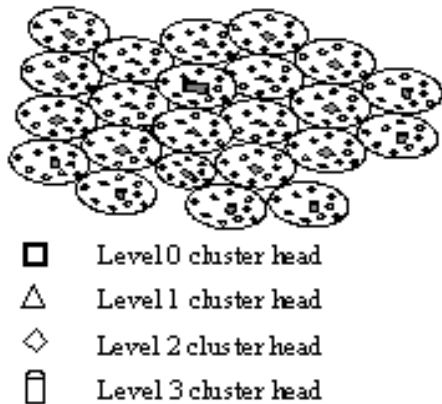


Fig. 3. Proposed Wireless Sensor Network Architecture

### III. IMPLEMENTATION OF TIME SYNCHRONIZATION ALGORITHM ON TELOS B

#### A. Definitions:

The discussion uses the following standard definitions:

**Accuracy:** Accuracy of a clock is how well its time compares with standard /true time.

**Precision:** How precisely time can be resolved in a particular time keeping system.

**Offset:** Offset of two clocks is the time difference between them.

**Skew:** Skew of two clocks is the frequency difference between them.

**Reliability:** The reliability of a time keeping system is the fraction of time it can be kept operating and connected in the network (without respect to stability and accuracy).

**Scope and availability:** The geographic span of nodes that are synchronized, and completeness of coverage within that region.

**Efficiency:** The time and energy expenditures needed to achieve synchronization.

#### B. Reasons for different local clock times of Nodes

There are three reasons for the nodes to be representing different times in their respective clocks.

1. The nodes might have been started at different times.

2. The quartz crystal at each of these nodes might running at slightly different frequencies, causing the clock values to gradually diverge from each other (termed as skew error).

3. The frequency of the clock can change variably over time because of aging and ambient conditions such as temperature (termed as drift error or instability).

#### C. Radio Message Delays [4]

There are following delays when sending a radio message: Send Time, Access Time, Transmission Time, Propagation Time, Reception Time, Receive Time.

#### D. Message types:

##### Top to bottom flow

- 1) Syn\_msg1: Time-sync message from cluster-head at level n to cluster-head at level n-1.
- 2) Syn\_msg2: Time-sync message from cluster-head at level n to nodes in same cluster ( non-cluster-head nodes).
- 3) Syn\_msg3: Time-sync message from cluster-head at level n to other cluster-head at same level.

##### Bottom to top flow

- 4) Syn\_msg4: Request for Time-sync message from non-cluster nodes to cluster-head of same level.
- 5) Syn\_msg5: Request for Time-sync message from cluster-head at level n-1 to cluster-head at level n.

#### E. Message Formats:

At this point we have considered only top to bottom flow of messages; Format of these messages are defined below:

```
typedef struct syn_msg1 {
uint8_t cLevel; // cluster level
uint16_t nodeID; // the parent node id
uint8_t seqNum; // sequence number for the root
uint32_t sendingTime; //global time
uint32_t arrivalTime; // not transmitted
} TimeSyncMsg1;
enum {
AM_TIMESYNCMMSG1 = 0xAA,
TIMESYNCMMSG1_LEN = sizeof(TimeSyncMsg1)-
sizeof(uint32_t),
TS_TIMER_MODE = 0, // TimeSyncMode interface
TS_USER_MODE = 1, // TimeSyncMode interface
};
```

```
typedef struct syn_msg2 {
uint8_t cLevel; // cluster level
uint16_t nodeID; // the peer node
uint8_t seqNum; // sequence number for the root
uint32_t sendingTime; //not transmitted
uint32_t arrivalTime; // arrival time of time sync message
} TimeSyncMsg2;
enum {
AM_TIMESYNCMMSG2 = 0xBB,
TIMESYNCMMSG2_LEN= sizeof(TimeSyncMsg2) -
```

```

sizeof(uint32_t),
TS_TIMER_MODE = 0, // see TimeSyncMode interface
TS_USER_MODE = 1, // see TimeSyncMode interface
};

typedef struct syn_msg3
{
uint8_t cLevel;
uint16_t nodeID; // the node id of the sender
uint8_t seqNum; // sequence number for the root
uint32_t sendingTime;
uint32_t arrivalTime; // not transmitted
} TimeSyncMsg3;
enum {
AM_TIMESYNCMMSG3 = 0xCC,
TIMESYNCMMSG3.LEN = sizeof(TimeSyncMsg3)-
sizeof(uint32_t),
TS_TIMER_MODE = 0, // TimeSyncMode interface
TS_USER_MODE = 1, //TimeSyncMode interface
};

```

#### IV. TEST SCENARIO

The algorithm is tested with 5 Telosb Motes[14].  
Node id 1: Connected to Computer USB port running the standard TOSBASE application. On PC side we ran a serial port listening java application that parses the TOS\_Msg sniffed by the TOSBASE. The reported data sniffed by TOSBASE application is shown in appendix A.  
Node id 2: Beacon sender broadcasts beacon after specified time (5 seconds or 30 seconds)

Every node receiving this beacon, responds back to TOSBASE. Each node sends its global and local time (global time is calculated with respect to local time here, that's why each node has different global time reported at each beacon's response to TOSBASE)

Node id 3: Client node in cluster 1.  
Node id 4: Client node in cluster 1.  
Node id 5: Clusterhead of cluster1.  
Nodes 3, 4 and 5 run test code that includes time synchronization algorithm and debugging component that handles sending data to base station on reception of beacon. The Testtimesync original nesC application written for mica2 is downloaded from a website[12]and then code is configured for telosb.

##### A. Test Results

Before running the Time synchronization algorithm on telosb motes, some environmental variable has to be set in cygwin. The tunable parameter setting of test code for telosb is shown in Appendix A.

##### Test 1: Only single cluster-head in the network (Node id=5)

The variation in global time multicasted by root node after

every 30 seconds is shown in Figure 4.

The maximum delay in global time in 30 seconds =  $100/960000 = 104.166\mu s$   
So maximum delay in global time in 1 second =  $100/960000 * 30 = 3.472\mu s$

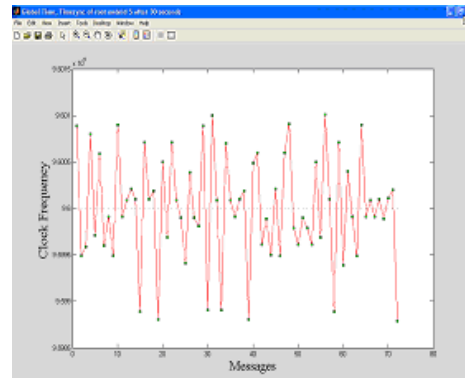


Fig. 4. Global time broadcasted by cluster-head after every 30 seconds

##### Test 2: Global and local time of single cluster-head (node id 5) in network recorded by base station (node id 1) after every beacon generated by beacon generator (node id 2) (see Figure 5 and figure 6)

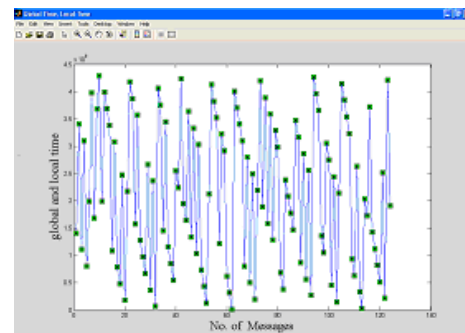


Fig. 5. Global and local time of cluster-head after every beacon

##### Test 3: Global time multicasted by cluster node and both clients. Client (node id 4) disturbed from synchronized condition. Cluster-head and client (node id 3) synchronized to nearly same global time by delay of few micro seconds(nearly 3.14us) (Figure 7).

#### V. CONCLUSION

Due to lack of wireless network simulation software for telosb motes, it is hard to simulate the whole code on computer; that's why we tested our algorithm on real telosb motes. For testing purpose we used the modified Testtimesync[15] component. Testtimesync code originally used for testing

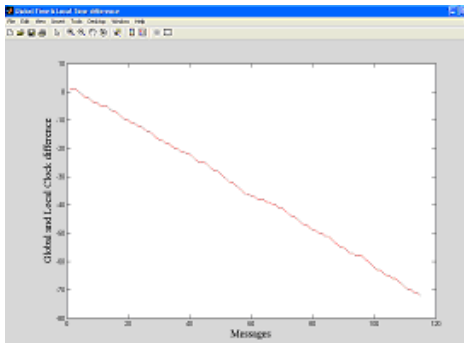


Fig. 6. Global and local time difference of client node

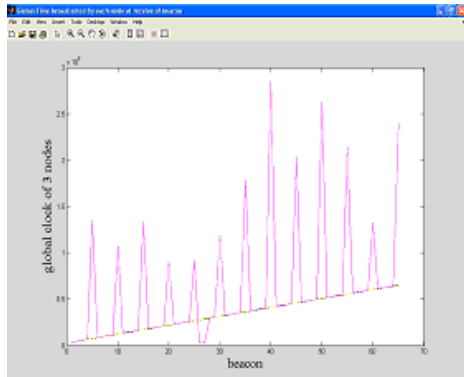


Fig. 7. Global time of cluster-head and client nodes.

FTSP on mica2 motes. Even though we modified the code to make it work well with telosb motes, but still there is some compatibility problem. We tested the algorithm only for one level of hierarchy (one Clusterhead and two client nodes). At one level the global time variation is coming in microseconds. Further analysis of algorithm for higher level of hierarchy is required.

#### ACKNOWLEDGEMENTS

We would like to acknowledge and thank Prof. M.RadhaKrishna of IIT, Allahabad for the help and contribution in enhancing the ideas. We also thank IIT, Allahabad for providing excellent lab facilities and motivating research environment.

#### REFERENCES

- [1] Anuragmayi Thuremella, Poonam Yadav, et al., "DOMA: Deploy Once Multi Application Sensor Network Framework," Proceeding of the 2nd International Conference on Wireless Communication and Sensor Networks, Dec, 2006.
- [2] Poonam Yadav, Anuragmayi Thuremella, Nagesh Yadav, et al., "Design of Flexible Mote: Fleximote," Proceeding of the 2nd International Conference on Wireless Communication and Sensor Networks, Dec, 2006.
- [3] Mills, D.L. "Internet Time synchronization: The Network Time Protocol," IEEE transactions on Communications COM 39 no. 10 p. 1482-1493, October 1991.

- [4] Ganeriwal, S., Kumar, R., and Srivastava, M.B., "Timing synchronization protocol for sensor networks". The First ACM conference on Embedded Networked Sensor System (SenSys), pp.138-149, November 2003.
- [5] Maroti M., Kusy B., Simon G., Ledeczi A., "The Flooding Time synchronization Protocol," ACM Second International Conference on Embedded Networked Sensor Systems (SenSys 04), pp. 39-49, Baltimore, MD, November 3, 2004.
- [6] "International Workshop on Wireless Sensor Networks and Applications," Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications San Diego, CA, USA Pages: 11 - 19. Year of Publication: 2003. ISBN: 1-58113-764-8
- [7] Elson, J., Girod, L., and Estrin, D., "Fine-grained time synchronization using reference broadcasts," Fifth symposium on operating Systems Design and Implementation, December 2002.
- [8] Ganeriwal, S., et al., "Rate-adaptive time synchronization for long-lived sensor networks," In Proceedings of ACM SIGMETRICS international conference on measurement and modeling in computer (Short Paper), pp. 374-375 , 2 pages , Alberta, Canada , June 2005.
- [9] Jungmin So, Nitin Vaidya, "MTSF: A Timing synchronization Protocol to Support Synchronous Operations in Multihop Wireless Networks," Technical Report, Dept. of Computer Science, and Coordinated Science Laboratory University of Illinois at Urbana-Champaign, October 2004.
- [10] Elson J.E, "Time synchronization in Wireless sensor Networks," Ph. D Thesis, University of California, Los Angeles 2003.
- [11] Karp, R., et al., "Optimal and Global Time in Sensor networks," Technical Report 0009, Center for embedded Networked Sensing, University of California, Los Angeles, April 2003.
- [12] <https://www.isis.vanderbilt.edu/projects/nest>
- [13] <http://webs.cs.berkeley.edu/tos>
- [14] <http://www.xbow.com/Products/productdetails.aspx?sid=252>
- [15] //tinyos-1.x/contrib/VU/Testtimesync
- [16] Schurr, N., Marecki, J., Tambe, M., Scerri, P., and Lewis, J., "The Future of Disaster-Response: Humans Working with Multi-Agent Teams (Without Being Overwhelmed)," American Association of Artificial Intelligence (AAAI) Spring Symposium, AAAI Press, Menlo Park, California-94025, 2005.
- [17] Lionel M. Ni, Yanmin Zhu, Jian Ma, Minglu Li, Qiong Luo, Yunhao Liu, S. C. Cheung, Qiang Yang, "Semantic Sensor Net: An Extensible Framework", ICCNMC 2005, pp.1144-1153
- [18] Poonam Yadav, "Design of Flexible Wireless Sensor Networks and Motes," M.Tech Thesis, Indian Institute of Information Technology, Allahabad. July-2007.

## APPENDIX A

Pseudocode of Cluster based hierarchical, flooding time synchronization

### A. Pseudocode

```

Pseudocode calculating the offset
Static i = 0;
Delay = 0;
Receive (TOSMsg)
If (TOSMsg.ClusterLevel == Clusterlevel - 1)
{ If (TOSMsg.type == AA )
{ If (Clusterhead)
{ Seq = TOSMsg.Sequence;
Globaltime = TOSMsg.globaltime; LocalTime = localtime;
//Offset = Globaltime + delay - LocalTime;
Call send(Broadcast address, TOSMsg);// TOSMsg.type =
BB;}}
if (TOSMsg.type == BB )
{If (Clusterhead){
if ( TOSMsg.globaltime ==Globaltime && TOSMsg.Sequence
==Seq)
{i++;
If (i%4 ){
//Offset [i] = Globaltime + 2*delay - localtime;
Delay += localtime - LocalTime;
}}//end if(i%4)} // end if tomsg
} //end if clusterhead
} //end if tomsgtype if (i= 3){
NewGlobaltime = call newglobaltime();
}

send (NewGlobaltime);
} //end if tomsg clusterhead

If (TOSMsg.type == DD && TOSMsg.ClusterLevel ==
Clusterlevel +1) { If (Clusterhead) { send ( Local_Node_id,
TOSMsg) // TOSMsg.type = AA; }
If (TOSMsg.type == CC && TOSMsg.ClusterLevel ==
Clusterlevel)
{If (! Clusterhead) {
Seq = TOSMsg.Sequence;
Globaltime = TOSMsg.globaltime
LocalTime = localtime;
}}
newglobaltime ()
{
AverageDelay = Delay / i;
Offset = Globaltime + averageDelay - LocalTime;
return NewGlobalTime;
}
send (NewGlobaltime)
{ generate new TOSMsg.
TOSMsg.type = AA;
TOSMsg.Globaltime = NewGlobalTime;
TOSMsg.Sequence = Seq;

```

```

TOSMsg.ClusterLevel = Clusterlevel;
Send (Broadcast address, TOSMsg); //message to next level
of clusterhead
Generate new TOSMsg
TOSMsg.type = CC;
TOSMsg.Globaltime = NewGlobalTime + averageDelay;
Send (Broadcast address, TOSMsg);
}

```

### B. Reported Data:

The following four messages are sniffed by the base station:

```

Syn_msg1: Msg.type = AA
Sync_msg3: Msg.type = CC
Testtimesyncmsg (TOS.Msg): Msg. type = B0
TimesyncPoll: Msg.type = BA

```

As Testtimesyncmsg and TimesyncPoll are not the part of CBH-FTSP, these are used only for testing purpose of CBH-FTSP. The message formats of Testtimesyncmsg and TimesyncPoll are taken from a paper[5]. As we have not considered skew delay in our algorithm, the few columns in the Testtimesyncmsg like skew and regression table entries are undefined.

### C. Tunable Parameters of test code for telosb

TIMESYNC\_RATE (seconds): defines the periodicity of transmission of time-sync msg by each node.  
TIMESYNC\_SYSTIME - if defined, the faster CPU (7MHz) clock is used, otherwise 32k external crystal is used . For telosb it is required that this should not be defined.  
TIMESYNC\_POLLER\_RATE - defines the periodicity of transmission of beacon message by reference broadcast.