# Implementation of TPSN protocol

**CS647 COURSE PROJECT REPORT**
by

**Abhirup Ghosh (Roll no :09305052)**
**Victor Chakraborty (Roll no :09305903)**

*under the guidance of*
**Prof. Bhaskaran Raman**

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Powai, Mumbai 400076.

## Contents

## 1 Introduction

Time Synchronization is very important issue in wireless networks. It is used in various fields like data collection, duty cycle management. In limited resource platform such as tmote time synchronization becomes a critical challenge.

Computer clocks mostly use hardware oscillator. For an ideal clock the oscillator frequency should be constant throught run but in practice due to various physical factors oscillator frequency varies unpredictably. This results in *clock drift*. To synchronize two clocks we need the offset (time gap) between them. But as drift varies with time this offset dosen't remains constant. So we need to time synchronize at regular interval.

We have implemented time synchronization by TPSN (Timing-sync Protocol for sensor Networks) and evaluated its performance with RBS (Reference Brodcast Synchronization) protocol.

## 2 Protocol Details

### 2.1 TPSN

Ganeriwal et. al. proposed this network wide tree based time synchronization protocol. This protocol comprises of two phases[1]:

- *Level Discovery Phase:* One node is selected as root node. This may be done dynamically using leader election algorithms. The root node is assigned level zero. It broadcasts *level_discovery* packets. Nodes receiving these packets assign themselves as level one nodes and rebroadcasts those packets. The broadcast chain goes on through the network.

- *Synchronization Phase:* The basic feature of the protocol is shown in the Figure 1. Node A sends a time sync message to Node B at T1. On receiving this message B notes down the local time T2 of the reception. It replies at time T3 to node A. A gets this reply at time T4. Propagation delay is assumed to be symmetric accross a link.

$$T2 = T1 + Offset + Prop\_Delay$$

$$T4 = T3 - Offset + Prop\_Delay$$

$$Offset = \frac{(T2-T1)-(T4-T3)}{2}$$

$$Prop\_Delay = \frac{(T2-T1)+(T4-T3)}{2}$$

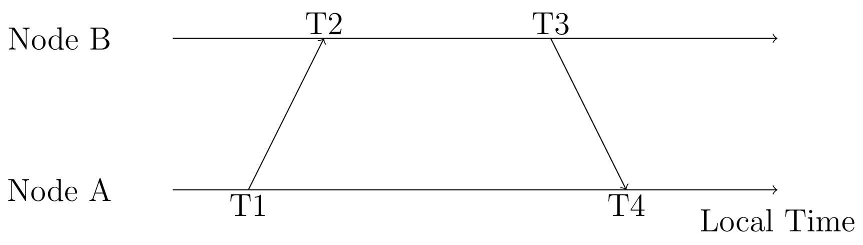

Fig1.Two way handshake of TPSN

## 2.2 RBS

Reference Broadcast Synchronization is based on the beacon messages send by a third party node [1]. This reception of the beacon message denotes a reference point for comparing the clocks.The basic assumption is that propagation delay from the root node to all the nodes in the network is constant.

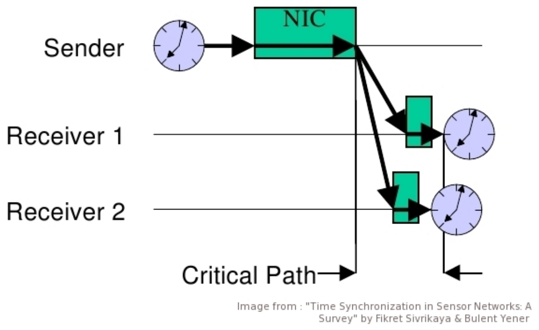
Sender
NIC
Receiver 1
Receiver 2
Critical Path
Image from : "Time Synchronization in Sensor Networks: A Survey" by Fikret Sivrikaya & Bulent Yener

Fig2.RBS Operation

## 3 Implementation Details

- *Hardware used:* Tmote architecture
- *Platform used:* Tinyos2.x installed on linux kernel version 2.6.24-22.
- *Topology:*

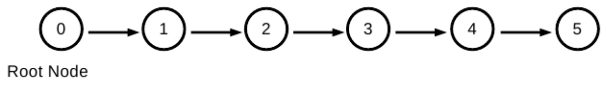

Fig 3A: Chain Topology

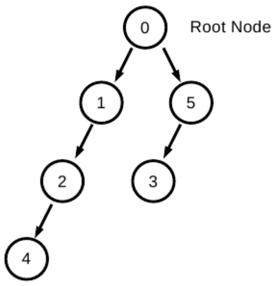

Fig 3B: Tree Topology

- *Configurations:*
  - RBS packet frequency is set to one packet per 5 seconds.
  - TPSN syncing is done once in a second.
- Experiment is done using error rate of 0%, 5%, 10%, 15%, 20%.
- These parameters can be configured by the header files.

TPSN node derive its RBS offset by comparing the reception time of its latest RBS packet and the latest RBS packet received by root. In ideal scenario, these two should have the same RBS ID. In practice due to packet delay and loss while travelling multihop, node often receive stale RBS information form the root. We have addressed this issue by buffering history of RBS pacekts at each node. If the root RBS ID is not equal to the latest RBS ID of a node, it looks into its buffer to pick up the RBS packet reception time for same ID as the root. Default size of this buffer is set to 5.

## 4 Evaluation

We have used the following metrics for measurement:

- TimeToSync : This represents the time taken between the TimeSync request to the parent and getting back the reply.
- SyncError : This is the deviation of the TPSN offset from RBS offset. We have considered RBS as standarad.

We have done experiments with two different topology (Fig.3) using different packet error rate. In each experiment we have taken measurements at every node.

## 5 Results & Analysis

- **TimeToSync** :
  We have measured this value for nodes at different depth in the network keeping PER constant at 10%.

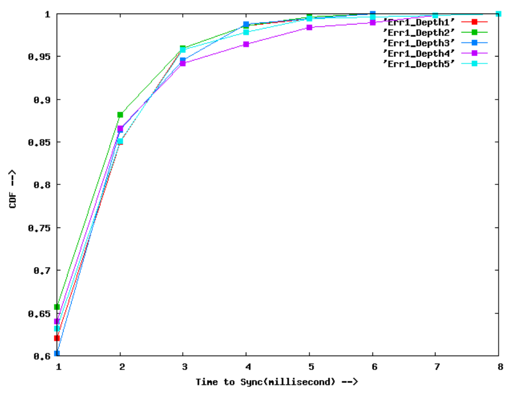

Fig4:CDF of TimeToSync at variable depth in chain topology at PER=10%

This figure shows that the TimeToSync value does not depend on node depth. We have observed same behaviour in tree topology. Hence without losing consistency we have done further experiments of this metric by selecting node1 (depth = 1) only.

We have varied PER and obtained graphs for chain and tree topology.

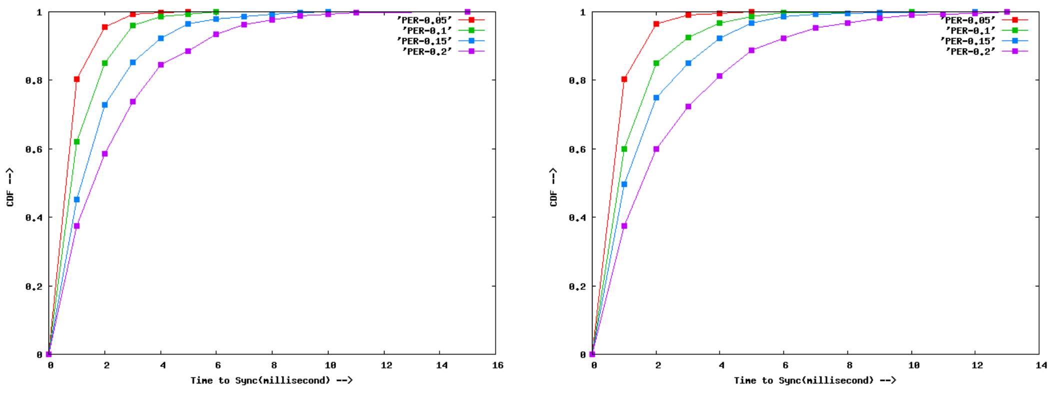

Fig5:CDF of TimeToSync at variable PER for chain(left) and tree(right) topology.

**Analysis :** Higher packet loss rate results into more delay beteween sending sync request and receiving reply. Hence TimeToSync value increases.

- **SyncError** :
  We have measured SyncError as a function for PER and Depth in chain and tree topology.

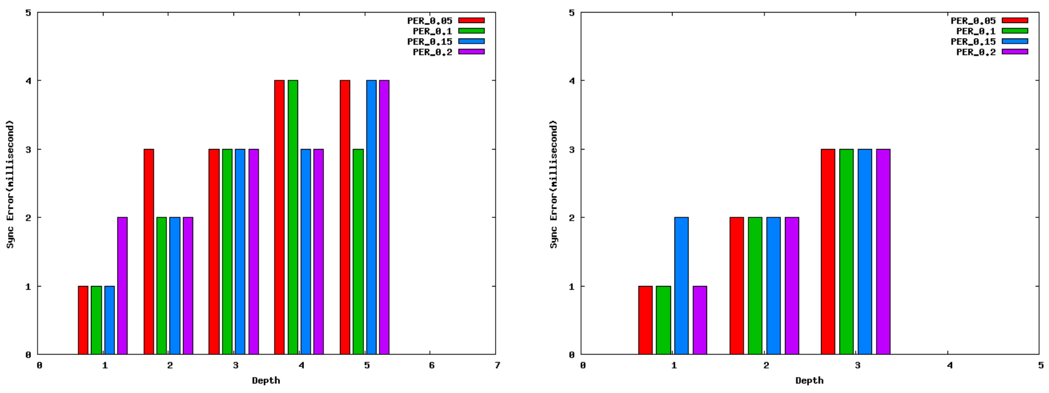

Fig6:Median of SyncError at variable PER and depth for chain(left) and tree(right) topology.

**Analysis :** TPSN protocol makes a node sync with its parent only. As we move down a network, errors at each node accumulates. Hence we get higher error values at node far from root.
Normally SyncError is found to be independent of PER. This is because even in case of high loss rate, a single successful two way handshake between parent and child is enough to sync the child.

## 6 Limitations

1. We have hardcoded the topology during our experiments. In absence of *link abstraction* it may happen that parent child link is not stable. This will make global synchronization from the root unpredictable and errorneous. Instead if we have used dynamic parent selection ,then any node may attach itself to the best link available to it. This would have reduced the problem.

2. We have observed that the random number generator inbuilt in TinyOs has a tendency to generate numbers with median value at about 0.3.This imperfection limits our experiments with error rate upto 20% only.

## 7 Conclusion

Hence we conclude from our experiment that TimeToSync is a function of Packet Error Rate only. It does not depend on depth. SyncError depends on depth of the node in the network but not on the Packet Error Rate.

## References

[1] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE network*, 18(4):45–50, 2004.