SENDER TO RECEIVER SYNCHRONIZATION IN WIRELESS SENSOR NETWORKS – A SIMULATION STUDY

1SHASHANK BHOLANE, 2DEVENDRASINGH THAKORE

1Student, M.E.(Computer), 2nd Year, B.V.U. College of Engineering,
2Professor, Computer Engineering, B.V.U. College of Engineering,
Bharati Vidyapeeth Deemed University,
Pune–411043, Maharashtra, India.

1shashankbholane@gmail.com, 2deventhakur@yahoo.com

ABSTRACT

Time synchronization in Wireless Sensor Networks (WSN) is very challenging area for researchers. There are numerous applications of Wireless Sensor Networks including habitat monitoring, forest fire detection, agriculture, medical, instrumentation, mining, smart homes, military applications and many others. Wireless Sensor Networks are generally deployed in harsh geographical and weather conditions. The main goal is collective or collaborative operation of numerous tiny sensors sensing some or the other parameter and sending all the data for fusion. Such collected data is processed and interpreted for observing specific phenomena. In all such applications time synchronization plays an important role to achieve a common notion of time. All sensor nodes have own clock with inherent drifts and they timestamp every data they send to sink node for processing. A fundamental approach for network wide synchronization is sender to receiver synchronization. This paper analyses the behavior of classical sender receiver synchronization protocol. The best way to study WSN is simulation. Protocol is implemented in Network Simulator and simulation guideline is set for the researchers.


1. INTRODUCTION

Wireless Sensor Network (WSN) [1] is basically an ad hoc network consisting of hundreds of tiny sensors communicating with each other and a gateway node for observing specific phenomena. The data fusion is crucial task as lot of data is generated through numerous sensors. The right interpretation of this collected data is important and depends on the time stamps on data packets received and sent by sensor nodes. For example if a moving vehicle is
being observed by few sensors then the position of sensors, time at which vehicle was detected by every sensor and the local clock of every sensor which timestamps this data with its time interprets the movement of that vehicle. Same applies to forest fire detection. Here the clocks play an important role and are expected to have a common notion of time. That is time synchronization is required.

To study time synchronization few test beds are available but with maximum 50-60 nodes and are very costly. To check the scalability of the protocols we need to rely on intuition. With advancement in VLSI technology the low power devices are easily available in cheaper rate. Still there are limitations on experiments with such tiny low power devices in laboratories. Specifically a Wireless Sensor Network consists of numerous small sensors with very limited battery life and other constraints. It becomes difficult for researchers to conduct experiments in labs and hence simulation is opted. There are many simulators available for WSN simulation like Glomosim [2], TOSSIM [2], COOJA [2], OMNET++ [2], Network Simulator 2 (NS2) [3] and many more. For simulation purpose NS-2 is most popular amongst researchers as it is open source. The best advantage is it works with C, C++, tcl/Otcl and at node level.

2. FACTORS AFFECTING TIME SYNCHRONIZATION

2.1 Environmental Effects
Drift rates of clocks may differ with fluctuations in environmental temperature, pressure, and humidity. While typical wired computers are operated in rather stable environments (e.g., A/C-controlled cluster rooms or offices), wireless sensors are frequently placed outdoors and in harsh environments where these fluctuations in ambient properties are common.

2.2 Energy Constraints
Wireless sensor nodes are typically driven by finite power sources, that is, either disposable or rechargeable (e.g., via solar panels) batteries. Battery replacement can add significantly to the cost of a WSN, particularly in large-scale networks and when the nodes are in difficult-to-service locations. Therefore, time synchronization protocols should not contribute significantly to the energy consumption of wireless nodes to ensure long battery life times. Since communication among sensor nodes is typically the basis for time synchronization, an energy-efficient synchronization protocol should aim for the minimum amount of the smallest possible messages necessary to obtain synchronized nodes.

2.3 Wireless Medium and Mobility
The wireless communication medium is known to be unpredictable and subject to fluctuations in performance due to changes in environmental properties caused by rain, fog, wind, and temperature. Numerous wireless sensors are mobile (e.g., mounted onto vehicles or carried by people), thereby causing significant and rapid changes in topology and connection quality. Finally, sensor nodes may fail or deplete their batteries, necessitating time synchronization that continues to remain functional even when network topology or density changes. In general, the consequence of these challenges is that time synchronization protocols must be designed for robustness and reconfigurability.

3. BASIC CATEGORIZATION [4]

Time synchronization algorithms can be divided in two main types: sender-receiver and receiver-receiver algorithms. Starting from a reference node, to which we want to synchronize the rest of the network, this can be achieved by:
• **Sender-receiver:** the reference node (sender) communicates to each other sensor (receiver), and after some message exchange the receiver is synchronized.

• **Receiver-receiver:** the reference node (sender) sends a message, and the synchronization takes place between two receivers, who compare their timestamps of the same message.

### 4. HOW TPSN WORKS

The Timing-sync Protocol for Sensor Networks (TPSN) [5] is a traditional sender-receiver synchronization approach that uses a tree to organize a network. TPSN uses two phases for synchronization: the level discovery phase (executed during network deployment) and the synchronization phase.

#### 4.1 Level Discovery Phase:

The goal of this phase is to create a hierarchical topology of the network, where each node is assigned a level, with the root node (e.g., a GPS-equipped gateway to the external world) residing on level 0. The root node initiates this phase by broadcasting a level_discovery message that contains the level and the unique identity of the sender. Every immediate neighbor of the root node uses this message to identify its own level (i.e., level 1) and rebroadcasts the level_discovery message with its own identity and level. This process is repeated until every node in the network has identified its level. When a node receives multiple broadcasts from its neighbors, it simply discards them once it has established its level in the hierarchical structure. Situations may occur where nodes do not have an assigned level, for example, when MAC-layer collisions prevent a node from receiving a level_discovery message or when a node joins a network that has already concluded its level discovery phase. In this case, a node can issue a level_request message to its neighbors who reply with their assigned levels. Then, the node assigns itself a level that is one greater than the smallest level received from its neighbors. Node failures can be handled in the same way, that is, when a node at level i realizes that it does not have any neighbors at level i − 1, it also issues a level_request message to reinsert itself into the structure. Finally, if the root node dies, instead of issuing level_request messages, nodes in level 1 execute a leader election algorithm, which then restarts TPSN by beginning a new level discovery phase.

#### 4.2 Synchronization Phase

During the synchronization phase, TPSN employs pairwise synchronization along the edges of the hierarchical structure established in the previous phase, that is, each i level node synchronizes its clock with nodes on level i − 1. A node j issues a synchronization pulse at time \( t_1 \), containing the node’s level and a timestamp. This message is received by node k at time \( t_2 \) and node k responds with an acknowledgment at time \( t_3 \) (containing time stamps \( t_1, t_2, t_3 \), and node k’s level). Finally, this packet is received by node j at time \( t_4 \). TPSN assumes that the propagation delay \( D \) and the clock offset do not change during the brief span of time. Since \( t_1 \) and \( t_4 \) are measured using node j’s clock and \( t_2 \) and \( t_3 \) are measured using node k’s clock, these times have the following relationships:

\[
t_2 = t_1 + D + \text{offset}\quad \text{and}\quad t_4 = t_3 + D - \text{offset}.
\]

Based on these parameters, node j can calculate both the drift and propagation delay as:

\[
D = \frac{(t_2 - t_1) + (t_4 - t_3)}{2}
\]

and

\[
\text{offset} = \frac{(t_2 - t_1) - (t_4 - t_3)}{2}
\]
The synchronization phase is initiated by the root node issuing a time_sync packet. After waiting for some random time (to reduce contention during medium access), nodes in level 1 initiate the two-way message exchange with the root node. Once a node in level 1 receives an acknowledgment from the root, it computes its offset and adjusts its clock. Nodes on level 2 will overhear the synchronization pulses issued by their level 1 neighbours and after a certain backoff time they initiate their pairwise synchronization with nodes in level 1. The backoff time is necessary to give level 1 nodes time to receive and process the acknowledgment of their own synchronization pulses. This process is continued throughout the hierarchical structure until all nodes have synchronized to the root node. The synchronization error of TPSN depends on the depth of the hierarchical structure and the end-to-end latencies experienced by messages during the pairwise synchronization. To minimize these latencies and to reduce the error, TPSN relies on time-stamping of packets at the MAC layer.

5. SIMULATION METHODOLOGY

We use a special topology generator GenSen[6] to generate topologies for Wireless Sensor Nodes. It generates two files which are compatible with the ns-2 simulator. We changed the GenSen as per our requirement. Once the node positions are fixed we read those positions through a c-program and convert that scenario into a graph data structure. The connectivity matrix is generated. Using dominating set algorithm [7] we found the minimum number of time synchronizing nodes. The simulation was run for many times and results were averaged. For Graph $G(V, E)$, and an integer $t$. We checked is there a set $T$ of size $t$ or less of synchronizing nodes such that every node in $\{V - T\}$ is at a $h$-hop distance from the nodes in $T$.

6. SIMULATION ENVIRONMENT

Ns-2 (Network Simulator version 2) [3], an object-oriented, discrete-event-driven network simulator targeted at networking research, which has been extensively used by the networking research community. The latest version for ns-2 is version 2.34.

Ns-2 is a powerful network simulator. It provides substantial support for simulation of TCP, routing, multicast protocols over wired and wireless (local and satellite) networks, etc. Users can define arbitrary network topologies composed of nodes, routers, links and shared medium. A rich set of protocol objects can then be attached to nodes, usually as agents. The simulator suite also includes a graphical visualizer called network animator (Nam) to assist the users get more insights about their simulation by visualizing packet trace data.

We simulated TPSN using Ns-2 and analyzed the results for different parameters. We used concepts in NRL extensions, modified as per TPSN protocol and created the required topologies using topology generator ‘Gensen’. [6]

1000X1000 m area was selected. Topology used was random topology.

7. SIMULATION RESULTS

We maintained an array of connected components. So we could easily find out which nodes are not at all covered. We call them isolated nodes. This number is very significant because these nodes cannot communicate with any other in the network as they are not ‘covered’.
As shown in the fig. 1 we incremented number of nodes from 20 to 140 in a step of 20 at a time. Default range of all nodes was set to 50m. As number of nodes was increased isolated nodes were reduced. It shows better coverage. It is desirable in WSN that there are least non-synchronized nodes.

Then keeping other parameters constant, we increased the range of the nodes gradually up to 100m. As per normal theorem in wireless networks we found that increasing the range decreased isolated nodes. But the range has many practical constraints in WSN since the conditions are different every time and constraints as discussed as above.

Figure 2 shows clearly as the range approaches 100 there is no node which is not part of the connected network. This ensures that the precision is very good as compared to other protocols.

Figure 3 shows a snapshot of working of protocol in ns-2.
CONCLUSION AND FUTURE WORK

In this paper, we proposed an approach to simulate time synchronization protocol in wireless sensor networks. We have shown how any network can be viewed as a graph and with graph theory we can analyze the characteristics easily. Sender to receiver synchronization is analyzed. For network wide synchronization fault tolerance is important issue. The fault tolerance can be analyzed by forcing few nodes to off state in simulation. We have set a simulation guideline which will help the researchers to study other time synchronization protocols by simulation. This will help to select a right time synchronization protocol depending on the application. Also few enhancements are suggested for sender-to receiver synchronization.

REFERENCES