Simulation and Analysis of Receiver-Receiver Time Synchronization in Wireless Sensor Networks

Shashank Bholane¹, Devendrasingh Thakore²
¹M.E.(Computer Engineering), BVU college of engineering, Pune, Maharashtra, India
²Professor, Computer Engineering, BVU college of engineering, Pune, Maharashtra, India
¹shashankbholane@gmail.com, ²deventhakur@yahoo.com

Abstract
Wireless sensor networks were initially deployed for military applications. Gradually researchers found them to be very useful in applications like weather monitoring, target tracking, agriculture, industrial applications, and recently smart homes and kindergartens. All the WSN applications need partial or full time synchronization. Applications like acoustic ranging, target tracking or monitoring need a common notion of time. Every data is time stamped sensor nodes local clock. Two main approaches to time synchronization are receiver-receiver synchronization and sender-receiver synchronization. In this paper we analyze the receiver-receiver synchronization and discuss the results of simulation in network simulator. This study, design considerations and simulation methodology will help a lot to the designer for designing a time synchronization scheme or system.

Keywords: Time synchronization, wireless sensor networks, protocol, receiver-receiver synchronization.

1. Introduction
Wireless sensor network is an ad hoc network consisting of numerous sensors connected wirelessly to achieve a common task collaboratively. There is a drastic difference in wired networks and WSNs. Wireless sensors are small in size and have many resource constraints like battery power, low memory, life etc. For a wired network, two methods of time synchronization are most common. Network Time Protocol (NTP) [1] and Global Positioning System (GPS) are both used for synchronization. Neither protocol is useful for wireless synchronization [2]. Both require resources not available in wireless networks.

The Network Time Protocol requires an extremely accurate clock, usually a server with an atomic clock. The client computer wanting to synchronize with the server will send a UDP packet requesting the time information. The server will then return the timing information and as a result the computers would be synchronized. Because of many wireless devices are powered by batteries, a server with an atomic clock is impractical for a wireless network.

GPS requires the wireless device to communicate with satellites in order to synchronize. This requires a GPS receiver in each wireless device. Again because of power constraints, this is impractical for wireless networks. Also sensor networks consist of inexpensive wireless nodes. A GPS receiver on each wireless node would be expensive and therefore unfeasible. The time accuracy of GPS depends on how many satellites the receiver can communicate with at a given time. This will not always be the same, so the time accuracy will vary. Furthermore Global Positioning System devices depend on line of sight communication to the satellite, which may not always be available where wireless networks are deployed.

The constraints of wireless sensor networks do not allow for traditional wired network time synchronization protocols. Wireless sensor networks are limited to size, power, and complexity. Neither the Network Time Protocol nor GPS were designed for such constraints.

2. Basics of time Synchronization

Synchronization is typically based on some sort of message exchange among sensor nodes. If the medium supports broadcast, multiple devices can be synchronized simultaneously with a low number of messages.

Synchronization Messages: Most existing time synchronization protocols are based on pair-wise synchronization, where two nodes synchronize their clocks using at least one synchronization message.
Network-wide synchronization can be achieved by repeating this process among multiple node pairs until every node in a network has been able to adjust its clock. [3]

2.1 One-Way Message Exchange

The simplest approach of pair-wise synchronization occurs when only a single message is used to synchronize two nodes, that is, one node sends a time stamp to another node. For example node i sends a synchronization message to node j at time $t_1$, embedding $t_1$ as time stamp into the message. Upon reception of this message, node j obtains a time stamp $t_2$ from its own local clock. The difference between the two time stamps is an indicator of the clock offset (between the clocks of nodes i and j) $\delta$. More accurately, the difference between the two times is expressed as:

$$t_2 - t_1 = D + \delta$$

where D is the unknown propagation time. Propagation times in the wireless medium are very small (a few microseconds) and are often ignored or assumed to be a certain constant value. Note that using this approach, node j is able to calculate an offset and adjust its clock to match the clock of node i.

2.2 Two-Way Message Exchange

A somewhat more accurate approach is to use two synchronization messages. Here, node j responds with a message issued at time $t_3$, containing time stamps $t_1$, $t_2$, and $t_3$. Upon reception of this second message at time $t_4$, both nodes are able to determine the clock offset, again assuming a fixed value for the propagation delay. However, node i is now able to more accurately determine both the propagation delay and the offset as:

$$\Delta = \frac{(t_2-t_1) - (t_4-t_3)}{2}$$

$$d = \frac{(t_2-t_1) + (t_4-t_3)}{2}$$

Note that this assumes that the propagation delay is identical in both directions and the clock drift does not change between measurements (which is feasible because of the brief time span). While only node i has sufficient information to determine the offset, node i can share the offset value with node j in a third message.

2.3 Receiver–Receiver Synchronization

In receiver–receiver synchronization principle, synchronization is based on the time at which the same message arrives at each receiver. This is in contrast to the more traditional sender–receiver approach of most synchronization schemes. In broadcast environments, these receivers obtain the message at about the same time and then exchange their arrival times to compute an offset (i.e., the difference in reception times indicates the offset of their clocks).

If there are two receivers, three messages are needed to synchronize both receivers. An example of such an approach is the RBS protocol [4] discussed in next section. Note that the broadcast message does not carry a time stamp.

2.4 Non-deterministic delay

Time synchronization schemes have four basic packet delay components: send time, access time, propagation time, and receive time [5]. As shown in Fig.1 the send time is that of the sender constructing the time message to transmit on the network.

![Figure 1. Non-deterministic delay components](image-url)

The access time is that of the MAC layer delay in accessing the network. This could be waiting to transmit in a TDMA protocol. The time for the bits to be physically transmitted on the medium is considered the propagation time. Finally, the receive time is the receiving node processing the message and transferring it to the host.

The major problem of time synchronization is not only that this packet delay exists, but also being able to
predict the time spent on each can be difficult [6]. Eliminating any of these will greatly increase the performance of the synchronization technique.

3. Reference Broadcast Synchronization (RBS) [4]

Many of the time synchronization protocols use a sender to receiver synchronization method where the sender will transmit the timestamp information and the receiver will synchronize to sender. RBS is different because it uses receiver to receiver synchronization. The idea is that a third party will broadcast a beacon to all the receivers. The beacon does not contain any timing information; instead the receivers will compare their clocks to one another to calculate their relative phase offsets.

The timing is based on when the node receives the reference beacon. The simplest form of RBS is one broadcast beacon and two receivers. The timing packet will be broadcasted to the two receivers. The receivers will record when the packet was received according to their local clocks. Then, the two receivers will exchange their timing information and be able to calculate the offset. This is enough information to retain a local timescale.

RBS can be expanded from the simplest form of one broadcast and two receivers to synchronization between \( n \) receivers; where \( n \) is greater than two. This may require more than one broadcast to be sent. Increasing the broadcasts will increase the precision of the synchronization. The reference beacon is broadcasted across all nodes. Once it is received, the receivers note their local time and then exchange timing information with their neighbouring nodes. The nodes will then be able to calculate their offset [4].

This protocol uses a sequence of synchronization messages from a given sender in order to estimate both offset and skew of the local clocks relative to each other. The protocol exploits the concept of time-critical path, that is, the path of a message that contributes to non-deterministic errors in a protocol. Fig. 2 and Fig. 3 compare the time-critical path of traditional protocols, which are based on sender-to-receiver synchronization, with receiver-to-receiver synchronization in RBS.

The delays that occur at the sender side are eliminated by using the physical layer broadcast in sensor networks. The critical path now contains the propagation and the receiver uncertainty. If, however, the transmission range is relatively small, then we can eliminate the propagation time and the critical path only contains the uncertainty of the receiver [4].

![Figure 2. Traditional time synchronization](image2)

![Figure 3. RBS synchronization](image3)

3.1 Multi-hop RBS

In many cases, the nodes that need synchronized time may not be in the coverage area of some common node. Then, some other nodes should act as gateways for time translation between neighbourhoods to route the time information from one node to another.

Fig. 4. depicts a case where multi-hop synchronization is required. For example, node 1 and node 7 are not in the same neighbourhood, i.e., they do not share a common sender from which they can both receive a synchronization pulse. In this case, node 4 acts as a gateway node between the two neighbourhoods. When senders A and B broadcast synchronization pulses to their neighbourhood as usual, node 4 gets both of these pulses and can thus relate the local clocks of A and B, i.e., the two neighbourhoods. When a beacon sender
broadcasts a synchronization pulse, it essentially creates a set of nodes (a neighbourhood) in which nodes can relate their local clocks among each other. Now consider a graph whose vertices correspond to sensor nodes in the network.

![Figure 4. Multi-hop RBS](image)

An edge between two vertices in this graph exists if the corresponding nodes in the network are within the same neighbourhood formed by RBS, i.e., if the two nodes can receive synchronization pulses from the same beacon sender. Then multi-hop synchronization can be performed along the edges of this graph. To this end, the concept of “time routing in multi-hop networks” is introduced. Finding the shortest path between two nodes would yield a minimal error multi-hop synchronization path for this pair of nodes [4]. Moreover, the authors proposed assigning weights to edges to represent the quality of pair-wise synchronizations (e.g., using the residual error of the linear fit). In the analysis of the multi-hop RBS algorithm, the authors argue that there is just a slow decay in precision by multi-hop synchronization; the average synchronization error is proportional to \( n \) for an \( n \)-hop network. [4]

### 3.2 Qualitative and Quantitative Evaluation of RBS

The precision of RBS was found to be 29.1µs. The piggybacking is absent. There is no GUI support. It was tested for very few nodes (2-20). Its accuracy, energy efficiency and overall complexity is high. It has good scalability with no fault tolerance in the basic protocol.

### 4. Simulation Methodology

The minimal number of sensor nodes required to monitor a specific area (A), is provided by the equation below, where \( r \) represents the sensing range of each sensor node.

\[
NS = \frac{2.A.PI}{r^2} * \frac{1}{\sqrt{27}}
\]  

(3)

However, this approach considers that all nodes have the same monitoring capabilities. Moreover, it does not take into account the existence of obstacles, such as trees or walls. Such scenario can only be applied when ideal radio environment characteristics are present. It is necessary to consider different deployment strategies, which could be used in inhospitable areas or could be more suitable to different sensor network applications. Since the problem of placement of time synchronizing nodes is NP-complete [7] we adopted three strategies: Greedy, heuristic and random.

### 4.1 Dominating Set Problem

We need to find out how many reference nodes or time synchronizing nodes are required to cover all the vertices of given graph (nodes in network represented as graph) i.e. dominating set.

#### 4.1.1 Heuristic approach:

In this section we present a heuristic method [8] for solving the minimum dominating set problem. This problem is formally defined as follows.

**Definition:** Let \( G = (V, E) \) be a graph. A subset \( D \) of \( V \), such that every vertex \( v \) in \( V - D \) is adjacent to at least one vertex in \( D \), is called dominating set of \( G \).

**Method:** Dominating Set (G)

**Input:** graph \( G = (V, E) \)

**Output:** dominating set \( D \)

We used an array, where, every element is cover count of that index \( v \), is the number of vertices currently covered by (i.e. adjacent to) \( v \). We assume that \( v \) is adjacent to itself. The second array, called Score, contains scores of vertices. Initially, the score of a
vertex equals to its cover count to reflect the rule of thumb that vertices with low degrees should be checked first. After applying series of operations every non-isolated vertex will be covered, too. Consequently, all the vertices of the input graph will be covered and we find a dominating set.

4.1.2 Greedy Method
Method:

\[ S = \emptyset; \]
while there exists white nodes do
\[ v = \{ v \mid w(v) = \max u w(u) \}; \]
\[ S = S \cup v; \]
end do

In order to understand the problem, we start with a very simple sequential method. We start with \( S = \emptyset \) and greedily add ‘good’ nodes to \( S \) until \( S \) is a dominating set. We call nodes in \( S \) black nodes which are covered (neighbours of nodes in \( S \)) grey, and all uncovered nodes white. By \( w(v) \), we denote the number of white nodes among the direct neighbours of \( v \), including \( v \) itself. We call \( w(v) \) the span of \( v \).

4.2 The Domination Number of a Graph \[9\]

If \( G \) is a simple graph of size \( n \) without isolated vertices and \( G \) is its complement, then, the following equation should be true:

\[ \gamma(G) + \overline{\gamma(G)} \leq n - \delta + 2 \quad \text{...... if } \gamma(G) > 3 \quad (4) \]

where \( \delta \) is the minimum degree of vertices in \( G \), which is assumed to be 1. The domination number \( \gamma(G) \) of \( G \) is the size of the smallest minimal dominating set. The domination numbers obtained for \( G \) and \( G^{\overline{\gamma}} \) for increasing number of nodes are as shown in the table below:

<table>
<thead>
<tr>
<th>No. of nodes(n)</th>
<th>( \gamma(G) )</th>
<th>( \overline{\gamma(G)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>200</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>300</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>400</td>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>500</td>
<td>50</td>
<td>1</td>
</tr>
</tbody>
</table>

We can further use the k-dominating set [11] of the graph. To ensure fault tolerance in synchronizing node placement, we must be able to ensure that even in case of failure of a synchronizing nodes, nodes dependent on that particular node still get synchronized. This is possible if we ensure that nodes are connected to multiple synchronizing nodes. The above problem can be reduced to the problem of finding the k-dominating set of a given graph.

4. Simulation Environment

Ns-2 (Network Simulator version 2) [12], [13] is 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 RBS using Ns-2 and analyzed the results for different parameters. We used concepts in NRL extensions, modified as per RBS protocol and created the required topologies using topology generator ‘Gensen’. [14]

Figure 5. RBS in operation – sending beacon packet

When the simulation starts, we have a network with some nodes designated as time synchronizing nodes. As shown in fig. 5 these nodes will start sending packets (of type BEACON) to their neighboring nodes. The packet will contain a sequence number and a sender ID. All its neighbors after receiving this packet will record the arrival of this BEACON packet and its
sequence number. As shown in fig.4 they will now send a packet (of type SYNC) containing the sequence number of the received SYNC packet, their ID as the sender ID and the time at which they received the packet. A node on receiving the SYNC packet from its neighbor can now calculate the offset between them by subtracting the time it received the packet by the time its neighbor received the packet. This way both of them get synchronized with each other.

Communication Overhead: Fig. 8 shows a relative comparison between the number of packets transferred (communication overhead) through the three approaches of dominating set algorithm and density of the nodes. As per the expectations, the heuristic algorithm works better than greedy algorithm. Also, the line for random approach can be ignored as the packets transferred are less because of the drastic increase in the number of isolated nodes.

5. Evaluation Results

We calculated the error using least-squares linear regression. As shown in Fig. 7, the error decreases exponentially with increase in the number of nodes and the error obtained is within the limits defined by RBS protocol. Also, the graph shows that heuristic method for dominating set problem gives the least error and thus, the best results.

6. Conclusion

In this paper we analyze receiver-receiver synchronization on qualitative and quantitative parameters. The classical RBS protocol is simulated and a simulation guideline is set for designers. The evaluation results show a critical part of analysis and will prove very beneficial for researchers. The methodology can be extended further for other time synchronization protocols.

REFERENCES

[6] B. Sundararaman et al., “Clock synchronization for


[10] On the minimum node degree and connectivity of a wireless multihop network – Christian Bettstetter, Technische Universität München


