

A Greedy Distributed Time Synchronization Algorithm for Wireless Sensor Networks

King-Yip Cheng, King-Shan Lui, Yik-Chung Wu and Vincent Tam

Department of Electrical and Electronic Engineering

The University of Hong Kong

Pokfulam Road, Hong Kong, China

Abstract—Time synchronization is a critical service for many wireless sensor network applications. However, traditional time synchronization protocols for wired networks are not suitable to resource-constrained sensor networks. In this paper, a distributed network-wise synchronization protocol is presented. The protocol employs Pairwise Broadcast Synchronization (PBS) in which sensors can be synchronized by merely overhearing the exchange of synchronization packets. We investigate how to minimize the number of PBS required to synchronize all nodes in a network. We show that the problem of finding the minimum number of PBS required is *NP-complete*. A distributed greedy algorithm is proposed. The protocol is tested by extensive simulations. Although the algorithm behind is heuristic-based, the performance is closed to the centralized algorithm. The message overhead is compared with that of Timing-Sync Protocol for Sensor Networks (TPSN). The message overhead introduced by the distributed greedy algorithm is justifiable by the messages saved from reducing the number of pairwise synchronizations performed.

I. INTRODUCTION

With the advance in various enabling technologies like Micro-Electro-Mechanical System (MEMS), signal processing and wireless communication, wireless sensor networks (WSNs) have drawn much attention from the academia and industry as they offer an unprecedented range of potential applications [1] [2]. These applications include habitat monitoring, target tracking, seismic detection, chemical detection, medical monitoring, etc. Despite the wide scope of applications, resource-constrained WSNs introduce a lot of challenges for researchers to tackle. Due to the limited computing power, energy and storage, services which can be readily provided in traditional wired networks have to be re-designed for wireless sensor networks. Time synchronization is one of these services [3]. In WSN applications involving data fusion, target tracking or event monitoring, precise timing information is critical to the functionality of the networks. Whenever an event is detected, packets will be sent to a data sink to report the event with two contexts, time and location of the event. Without coupling these two contexts, the detection of an event is meaningless to many applications. Time synchronization is needed to provide a consistent notion of time across the network. Precise timing is particularly important to emerging wireless multimedia sensor networks [4] which multimedia data may be streamed in real time.

Time synchronization has been studied for a long time in wired distributed systems, the Network-Time-Protocol

(NTP) [5] is the de-facto time synchronization protocol in the Internet. Computers can be synchronized to a reference time source with an accuracy of milliseconds. Since sensor nodes are battery-powered but are expected to operate for a long time without human intervention, implementation of NTP in battery-powered sensors is too costly in terms of energy consumption. Sensors near the reference node will become traffic hot-spots when relaying time synchronization packets and the network lifetime will be affected substantially.

Much effort have been spent in synchronizing a pair of nodes. Most synchronization protocols for WSNs are based on two fundamental approaches, Sender-Receiver Synchronization (SRS) and Receiver-Receiver synchronization (RRS). The former is a traditional approach adopted in NTP. Node to be synchronized initiates a two-way message exchange with the reference node. Timestamps are exchanged to estimate the clock offset and skew. The latter approach is similar to SRS but involves three or more nodes. Although these two approaches enable pairwise synchronization in WSNs, extending pairwise time synchronization to network-wise time synchronization in an energy efficient manner is still an issue in wireless sensor network. Recently, Noh *et al.* [6] proposed a protocol for synchronization in WSNs called Pairwise Broadcast Synchronization (PBS) protocol which can greatly reduce the number of message exchange in time synchronization with comparable accuracy. PBS makes use of the broadcast channels of WSNs. Through overhearing synchronization packet exchange, neighbouring nodes can synchronize themselves with the reference node. Noh's approach can greatly reduce the number of packet exchange required but it requires two super nodes to broadcast the synchronization packets to the sensors. There is a lack of efficient distributed protocol to perform network-wise synchronization.

In view of this, we propose a distributed multi-hop synchronization protocol with PBS in this paper. The protocol is designed for networks whose topologies remain static in a considerable period of time. The paper is organized as follows. Related works are discussed in Section II. Fundamental approaches and existing protocols for synchronization in WSNs will be discussed. In Section III, we analyze the network-wise synchronization problem with PBS. We demonstrate that finding the optimal synchronization sequence with PBS is an *NP-complete* problem. In Section IV, a distributed PBS-based network-wise synchronization protocol will be presented. We

justify our proposal by extensive simulations. Results are presented in Section V. We conclude the paper in Section VI.

II. RELATED WORKS

Elson *et al.* [7] proposed a receiver-only protocol called Reference Broadcast Synchronization (RBS). In the simplest mode of RBS, a beacon node use the broadcast channel to send beacons to a pair of receivers. The receiver marks the reception time and exchange the timestamps upon receiving the beacons. It should be noted that the beacons do not bear any timestamps. This can remove the nondeterministic delay in the send time and access time. It is assumed that the propagation delay is negligible. If there is no clock skew, the time offset between the receivers is simply the difference of the timestamps. To improve the accuracy, multiple beacons are emitted and the offset is obtained by taking the average of the samples. If clock skew is taken into consideration, receivers can estimate the relative clock skew and offset by linear regression. Despite the simple procedures on the receiver side, providing periodic reference pulses is much more costly, especially in multihop manner. Special nodes have to be provided to transmit the reference pulses or sensors take turns to broadcast the reference pulses periodically. However, both methods do not scale to large sensor networks.

Maroti *et al.* [8] proposed a similar approach called Flooding Time Synchronization Protocol (FTSP). The sender also broadcast a beacon but with timestamp. MAC-layer timestamps are used to remove the jitter of interrupt handling and encoding/decoding times. Thus, receivers do not need to exchange their timestamps of reception to estimate the clock offset. Clock skew can also be estimated using linear regression like RBS. However, the multihop synchronization protocol does not take energy efficiency into consideration. The timing information is distributed by flooding synchronization packets across the network.

Ganeriwal *et al.* [9] proposed the Timing-sync Protocol for Sensor Network (TPSN) which resembles to NTP. The protocol follows the sender-receiver approach. Nodes are synchronized in a pairwise manner. The sender synchronizes its clock to the clock of the receiver. The synchronization process involves exchanging of two packets. The sender first sends a timestamped packet to the receiver. The receiver marks the instant when it receives the packet and sends back an timestamped acknowledgement to the sender. Hence, the acknowledgement bears two timestamps. After receiving the acknowledgement, the sender obtains four timestamps, the sending and reception times of the two packets. The clock offset between the sender and the receiver can be estimated by these timestamps. To extend the pairwise synchronization to multi-hop synchronization, a level discovery phase is placed before synchronization is performed. The level discovery phase constructs a hierarchical structure in the network. A sensor is selected as the root node and has the highest level. Nodes are synchronized along the hierarchical structure. Sensors are synchronized by counterparts of the level that is immediately above.

Sichitiu *et al.* [10] proposed two sender-receiver synchronization protocols called Tiny-Sync and Mini-Sync. The mechanism is similar to TPSN that sender and receiver exchange timing packets. However, the receiver immediately sends back the acknowledgment after receiving the first packet from the sender. The timestamps are used to establish bounds on the relative clock skew and offset. Tiny-Sync differs from Mini-Sync in terms of the storage and computational requirements. Tiny-Sync requires less storage and computations but also produces less accurate results. Tiny-Sync and Mini-Sync adopt a level-based strategy similar to the TPSN in global synchronization.

Since TPSN, Tiny-Sync and Mini-Sync are all pairwise in nature, sensors of higher levels involve in multiple synchronization processes that are initiated by nodes of lower levels. Sensors have to spend considerable amount of energy for it.

Li *et al.* [11] proposed two global synchronization algorithms, the Rate-based Synchronous Diffusion Algorithm and Asynchronous Diffusion Algorithm. In the synchronous algorithm, node i exchanges its clock times with its neighbours and updates its own clock according to the difference between its clock and those of its neighbours. A parameter called diffusion rate, r_{ij} , is used to control how much the clock of sensor i is adjusted towards that of sensor j . It is shown that if every node carries out the synchronization process accordingly, the clocks of all sensors will be converged. Since, node operations have to be performed in a set order in the synchronous algorithm, Asynchronous Diffusion Algorithm is proposed to remove this constraint. Clocks are updated as the local averages computed by connected neighbours. Unlike previous algorithms, clocks are not eventually synchronized to a reference time but to the average value.

Noh *et al.* [6] proposed a receiver-only synchronization approach called Pairwise Broadcast Synchronization (PBS). Similar to RBS, PBS use the broadcast channel to synchronize nodes within the same broadcast domain. Through broadcasting, two super nodes, A , P , exchange timing information like TPSN. When the broadcast messages arrive at node B , node B marks the arrival times and obtain the timestamps broadcasted by node A and P . With the overheard timestamps and the arrival times, sensor B can estimate the clock skew and the offset relative to node P . In [6], it is shown that PBS achieve the same synchronization accuracy as RBS. Thus every node overhears the timing information exchange can synchronize itself without sending a message. This greatly reduce the number of total message required for synchronizing nodes within a broadcast domain. Energy can be saved since receiving usually consumes less energy than transmitting. Although PBS enables groupwise synchronization, there is no multihop synchronization protocol to extend PBS for network-wide synchronization.

III. ANALYSIS OF NETWORK-WISE SYNCHRONIZATION WITH PBS

In most WSN applications, it is almost impossible to install two super nodes with communication ranges covering the

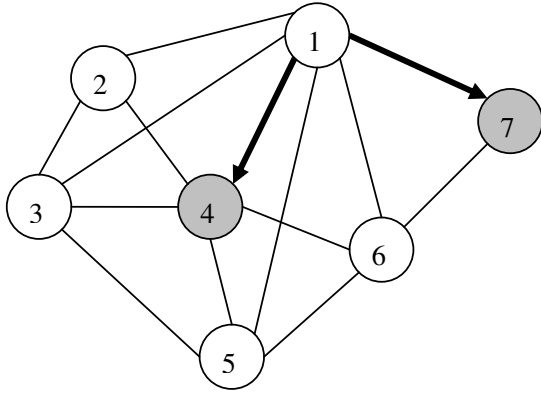


Fig. 1. Synchronization sequence: $(1 \rightarrow 4), (1 \rightarrow 7)$

whole sensor network. To relax the requirement of super nodes, nodes have to perform PBS by themselves. A multi-hop time synchronization protocol is needed so that all nodes can synchronize to a reference time. A particular node is chosen as the reference node and its local time is considered as the reference time. A simple protocol can be adapted from existing works. All nodes can be synchronized by successive synchronizations. Nodes that are one hop away from the reference node are first synchronized. Then, the synchronized nodes can help to synchronize nodes that are two hops away from the reference node. The process continues until all nodes are synchronized. However, this simple protocol does not allow full play to PBS. Nodes overhearing a complete PBS message exchange can synchronize themselves. Thus, if the pair of nodes performing PBS are carefully chosen, the number of PBS can be reduced, hence saving more energy. Figure 1 illustrates an example. Node 1 is the reference node. The rest of the nodes are unsynchronized and are neighbours of node 1. If node 1 knows the connectivity information, node 1 can minimize the number of PBS by synchronizing with the node that the maximum number of unsynchronized nodes can be simultaneously synchronized by overhearing. Therefore, it will synchronize node 4 by PBS. This makes node 2 to node 6 become synchronized at the same time. Afterwards, only node 7 is left unsynchronized. It will be synchronized in the next PBS and all nodes are synchronized thereafter. Node 7 can be synchronized by node 2 after node 2 is synchronized, but synchronizing with node 1 is more preferable. It is because synchronization error will be accumulated. Although a solution can be found in this way, it may not be the best in terms of the number of PBS performed. In the later section, we will show that the problem of finding the minimum number of PBS to synchronize all neighbours of the reference node is *NP-complete*.

The problem becomes more complicated when unsynchronized nodes are more than one-hop away from the reference node. These nodes can be synchronized by more than one synchronized nodes and unnecessary PBS may be performed if decision on the synchronization sequence is made by each

synchronized node independently. Considering the previous example again, the second PBS performed between node 1 and node 7 is unnecessary if node 7 is connected to another synchronized node and just has been synchronized.

To have a more complete analysis, we look at the network-wise synchronization problem from a centralized point of view. Suppose there are N nodes in a sensor network which $N - 1$ nodes are unsynchronized and the one left is the reference node. The network can be considered as a graph $G(V, E)$. V is the set of points representing the nodes in the network. For any pair of nodes $v_i, v_j \in V$, $(v_i, v_j) \in E$ if node i and node j can communicate with each other. Nodes are assigned to levels which represents the hop count from the reference node. To minimize the synchronization error, level $(i + 1)$ nodes should only be synchronized by nodes of level i which means that it either synchronizes directly with a node on level i or overhears the messages exchanged between a level i node and a neighbor of level $i + 1$. Then the network-wise synchronization can be broken down into a number of independent sub-problems. Each sub-problem is to synchronize all nodes of level $(i + 1)$ to nodes of level i by minimum number of PBS. Sub-problems can be further classified into two types depending on the number of nodes in the lower level, level i . In most scenarios, each sub-problem is basically the same except for the sub-problem of synchronizing level 1 nodes since there is only one node in level 0, the reference node.

A. Synchronizing nodes of level 1

Let $G(V, E)$ be a graph that $V = \{v_r, v_1, \dots, v_n\}$ represents the set of nodes of level 1 and the reference node, v_r . Let D be the synchronization set which is a subset of level one nodes. The reference node performs PBS with every node in D . We aim to find a D such that for all nodes $v_i, i \neq r$, at least one of the following conditions are met:

- $v_i \in D$ such that v_i can be synchronized by exchanging timing information with the reference node.
- $(v_i, v_j) \in E$, for some $v_j \in D$ such that v_i can be synchronized by overhearing the synchronization packets exchanged between v_j and the reference node.

In this Level 1 problem, finding the D with the minimum size is actually equivalent to the dominating set problem. An instance of dominating set problem consists of a graph, $G'(V', E')$ which V' is the set of vertices and E' is the set of edges. A dominating set K , is a subset of V' such that for all $(v'_i, v'_j) \in E'$, either v'_i or v'_j or both are in K . It is *NP-complete* in finding the dominating set K with the minimum size. Given a dominating set instance $G'(V', E')$, we can reduce it to Level 1 problem, $G(V, E)$, by

$$V = V' \cup \{v_r\}$$

$$E = E' \cup \{(v_r, v_i) | v_i \in V'\}$$

The reduction is illustrated in Figure 2. Suppose that G has a synchronization set $D \subseteq (V - \{v_r\})$, then for all non-reference node v_i in $(V - D)$, there exists a v_j in D such that $(v_i, v_j) \in E$. Since $v_i, v_j \neq v_r$, $(v_i, v_j) \in E'$, hence, D is the dominating set of $G'(V', E')$.

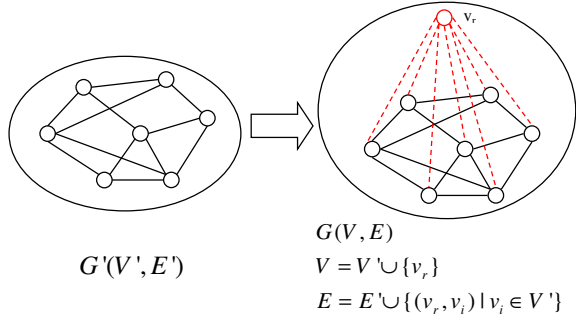


Fig. 2. Reducing a dominating set problem to a Level 1 problem

B. Synchronizing nodes of level $(i + 1)$, $i > 0$

In our Level 1 problem, there is only one reference node which nodes can exchange timing information with. However, it is very likely that any level $(i + 1)$ node can be synchronized by more than one node in level i , for $i > 0$. In this section, we analyze this sub-problem and we call this problem Higher Level Problem (HLP). Let $G(V, E)$ be the graph representing the topology of level i nodes and level $(i + 1)$ nodes. $V = V_M \cup V_N$ is the set of vertices which $V_M = \{m_1, \dots, m_M\}$ are nodes of level i and $V_N = \{n_1, \dots, n_N\}$ are the nodes of level $(i + 1)$. E is the set of edges. Each edge, $(j, k) \in E$, implies that connectivity exists between sensor j and sensor k . There are three types of edges in E : edges connecting nodes in V_M only, edges connecting nodes in V_N only, and edges connecting one node in V_M and one node in V_N . Note that every node in V_N must be a neighbour of a node in V_M . PBS can only be performed between two nodes that are connected by an edge. In addition, one of the nodes is in V_M and another one is in V_N . When PBS is performed between m_j and n_k , where $m_j \in V_M, n_k \in V_N$ and $(m_j, n_k) \in E$, n_k becomes synchronized and those nodes that are connected to both m_j and n_k can overhear the messages and get synchronized as well. Our HLP is to identify the smallest subset, D , of E , such that for every $n_l \in V_N$, at least one of the following conditions is met,

- $\exists m_j \in V_M$ s.t. $(m_j, n_l) \in D$
- $\exists (m_j, n_k) \in D$ s.t. $(m_j, n_l), (n_k, n_l) \in E, m_j \in V_M, n_k \in V_N$.

Furthermore, we denote $S_{j,k}$ as the set of nodes that can be synchronized by performing PBS between node m_j and node n_k . It can be shown that the well-known *NP-complete* set cover problem [12] can be reduced to our HLP.

A set cover problem instance (X, F) consists of a universe X and a family of subset F . Each element of X belongs to at least one subset m , i.e.

$$X = \bigcup_{m \in F} m$$

The set cover problem is to find a subset of F of minimum size that covers all the elements of X . The reduction is given below and an example is illustrated in Figure 3 and 4.

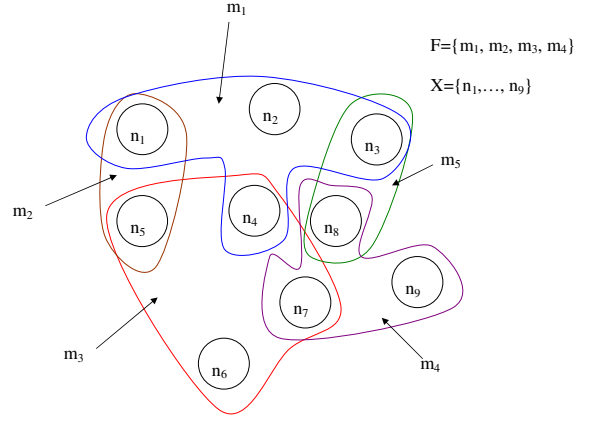


Fig. 3. A set cover problem instance, (X, F)

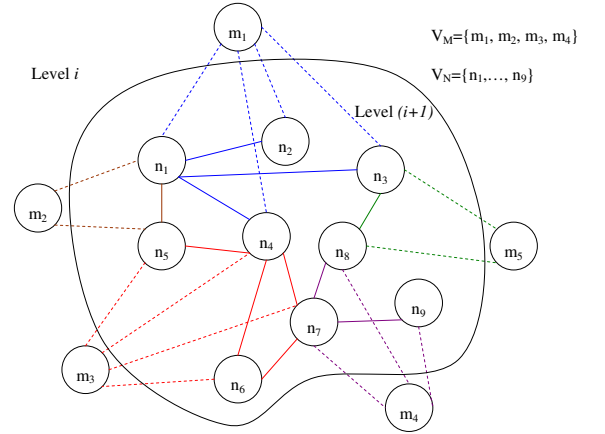


Fig. 4. Sub-problem corresponding to set cover problem of Figure 3

For a set cover problem instance (X, F) which $X = \{n_1, \dots, n_{|X|}\}$ is the universe, and $F = \{m_1, m_2, \dots, m_{|F|}\}$ is a family of subsets, we map each n_k to a node of level $(i + 1)$. We also map each subset m_j to a level i node. Hence, $V_M = \{m_1, \dots, m_{|F|}\}$ and $V_N = \{n_1, \dots, n_{|X|}\}$. For each $m \in F$, we put $(n, m) \in E$ for every n in subset m . Let n' be the first element of a subset m , we put $(n', n) \in E$ for every n in the subset m . Then an instance of the sub-problem is obtained. Figure 3 is the original set cover problem and Figure 4 is the corresponding HLP.

We argue that $A \subseteq F$ is the family of subsets that covers all elements in the set cover problem iff $D = \{(a, n) | n \text{ is the first element in } a, a \in A\}$ is a solution of HLP.

If A is a set cover of V_N , performing PBS between (a, n) where n is the first element in a will synchronize all nodes in a . If PBS are performed for all a in A , then all nodes in V_N will be synchronized as A covers all nodes in V_N . Thus, D is a solution of HLP.

If $D = \{(a, n) | n \text{ is the first element in } a, a \in A\}$ is a solution of HLP, a subset of nodes in V_N is associated to each a which is the subset of nodes that are synchronized

by performing PBS between a and n . The family of subset of nodes that are associated to A must cover V_N since every node in V_N is synchronized by D .

IV. A DISTRIBUTED GREEDY GLOBAL SYNCHRONIZATION PROTOCOL FOR WSNS

In the previous section, we have shown that the problem of finding the optimal synchronization set is of NP-complete, no matter which levels of nodes are considered. Although no polynomial-time algorithm has been found yet, sub-optimal solutions for dominating set problems and set cover problems can be obtained by approximation algorithms [13], e.g. greedy algorithm. A synchronization set D for HLP can be constructed as follows:

Algorithm 1 Centralized Greedy Algorithm

```

 $D = \emptyset, L = \emptyset, X = \emptyset$ 
while  $X \neq V_N$  do
  select  $(m_j, n_k)$  which maximize  $|S_{j,k} \cap (V_N - X)|$ 
  add  $(m_j, n_k)$  into  $D$  and add  $S_{j,k}$  into  $L$ 
   $X = \bigcup_{S \in L} S$ 
end while

```

The Level 1 problem can also be solved in a similar manner. However, this algorithm is centralized in nature. It can only be performed by the reference node in practice since information required is confined in a local region around the reference node. The number of packets exchanged are relatively small. But, huge communication overheads will be incurred if the centralized greedy algorithm is applied in other levels. The connectivity information between nodes of different levels has to be known, too. Moreover, the information has to be sent to a dedicated node which runs the centralized greedy algorithm to determine the synchronization set. Finally, the decision has to be distributed back to the sensors. All these procedures will inevitably introduce substantial communication overheads. The overheads may overshadow the energy saved by using PBS.

In view of this, we propose a distributed heuristic-based protocol for network-wise synchronization with PBS. It is assumed that every node in the network has a unique ID and levels are assigned to sensors as in the level discovery phase presented in [9]. Again, we represent the connectivity of level i and $(i + 1)$ nodes by a graph $G(V, E)$ in the same way as previous section. The protocol for level i and level $(i + 1)$ nodes can be broken down into the following steps:

- (1) Nodes first discover how many neighbours they have. Neighbour information can be obtained when the level discovery phased is performed. Neighbours of a level i node are classified into three categories based on their levels, level $(i - 1)$, level i and level $(i + 1)$. A list, L , is created by each sensor of level i that contains the IDs of all of its level i neighbours. For each sensor of level i , they send the list L to their level $(i - 1)$ neighbours to notify them which level i sensors are their neighbours. They also receive neighbour lists from nodes of level

$(i + 1)$.

- (2) For every node of m_j of level i , after receiving all neighbour lists from its level $(i + 1)$ neighbours, m_j can determine the maximum number of its neighbours that can be synchronized by **one** PBS. Let that number be $sync_num$ of m_j and that particular level $(i + 1)$ node be n_k (see Figure 5). The $sync_num$ of m_j will be sent to all level i neighbours of m_j .

- (3) Each level i node will receive all the $sync_num$ of its level i neighbours eventually. If the $sync_num$ of m_j itself is the largest among the received $sync_num$, m_j will send each level i neighbour a list, L' , which contains the IDs of the sensors that can be synchronized by performing PBS with node n_k . Node IDs comparison can be used as a simple tie-break for equal $sync_num$. Node n_k is included in another list, the synchronization list of m_j . m_j will synchronize with the nodes in the synchronization list later. List L of m_j is updated as $L = L - L'$, i.e. the synchronized nodes will be removed from the neighbour list of m_j . Hence, $sync_num$ is also updated and it must be smaller than the old value. The updated $sync_num$ of m_j is also distributed together with L' . If the updated $sync_num$ equals 0, m_k can jump to step (5), afterwards. Thus, neighbours receiving the $sync_num$ of value 0 are indirectly notified that they do not need to wait for any update from m_j anymore. If m_j finds that its $sync_num$ is not the maximum, it will send a packet *not_MAX* to its level i neighbours (see Figure 6). Then, m_j waits for replies from its neighbours and there are two possible scenarios for m_j after receiving all replies from its level i neighbours.

- m_j receives one or more lists L' and corresponding $sync_num$ from its level i neighbour(s) and packets *not_MAX* from the rest of its level i neighbours. List L of m_j will be updated as $L = L - (\bigcup L')$. A new $sync_num$ can be determined. The new $sync_num$ will be distributed to all level i neighbours of m_j . If the updated $sync_num$ equals 0, m_j will jump to step (5) after sending the $sync_num$. As mentioned before, neighbours of m_j will not wait for any update from m_j .
- m_j receives packets *not_MAX* from all of its level i neighbours. The $sync_num$ of m_j remains unchanged. Then, m_j will send its $sync_num$ to all of its level i neighbours again. This scenario is possible. The node which m_k thought to have the maximum $sync_num$ may not find itself have the maximum $sync_num$. That node and m_k have two different neighbour sets.

- (4) Step (3) will be reiterated until all lists L of level i nodes become empty lists.

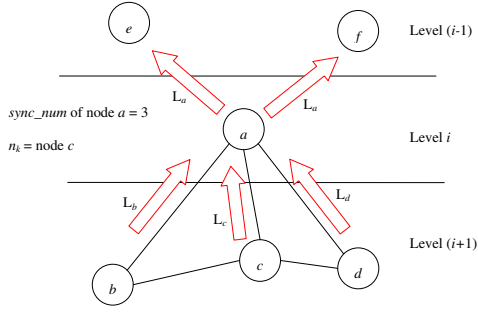
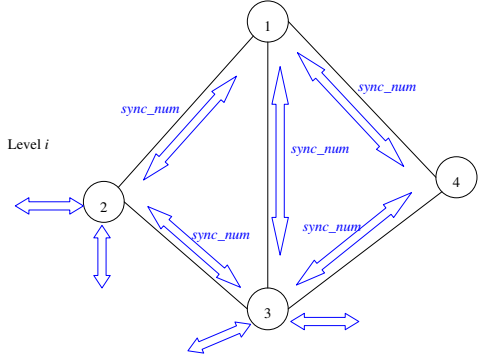
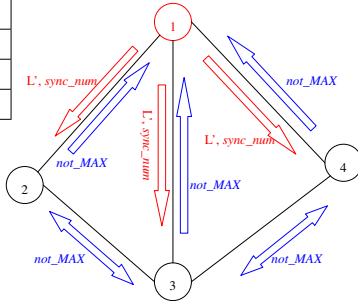


Fig. 5. Nodes exchanging neighbour lists and determining $sync_num$



(a) Nodes exchanging $sync_num$ with their neighbours

Level i	$sync_num$
node 1	5
node 2	2
node 3	3
node 4	2



(b) Node with local maximum of $sync_num$, i.e. node 1, distributes list L'

Fig. 6. Local information exchange

(5) m_j starts performing PBS with the nodes in its synchronization list.

All level $(i + 1)$ sensors must be synchronized eventually as there exist at least one sensor which finds itself having the maximum $sync_num$ after step (3) is iterated once by each level i sensor. Although a certain amount of messages are transmitted to determine the synchronization set, the overheads are incurred just once. The synchronization set remains the same if the network topology does not change. Nodes simply perform step (5) for resynchronization.

V. SIMULATION

Simulations are conducted to justify our protocol. We do not focus on the timing accuracy since PBS is shown to be as accurate as RBS in [6]. Instead, we study how many rounds

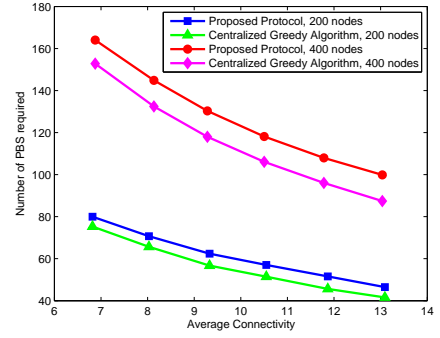


Fig. 7. Average number of PBS required for network-wide synchronization

of PBS are required to obtain network-wide synchronization in a WSN. We also investigate the number of messages required. In the simulations, sensors are uniformly distributed across a square grid. Assuming that sensor x can communicate with sensor y if they are within the communication range of each other, i.e. $\|x - y\| \leq R$. Furthermore, communication ranges are equal among all sensors and R is adjusted to give different degrees of connectivity, i.e. the average number of neighbours per node. 60 network topologies are generated randomly for each setting.

A. Number of Pairwise Broadcast Synchronization for Network-wide Synchronization

Figure 7 gives the average number of PBS required to achieve network-wide synchronization with different degrees of connectivity using our proposed protocol. Results of networks with 200 nodes and 400 nodes are presented. We compare the result with that obtained by the centralized greedy algorithm. It can be observed that our proposed protocol can achieve performance close to that of the centralized counterpart. Networks with 200 nodes and connectivity around 7 can be synchronized after around 80 PBS are performed. If TPSN is used to synchronize the same 200-node network, each node except the root node must synchronize with another node, giving 199 pairwise synchronizations. Therefore, the proposed protocol is more efficient than TPSN in terms of the number of pairwise synchronizations. Also notice that, the number of PBS required drops when connectivity is increased as more neighbours can be synchronized by one PBS. The performance of the proposed protocol is stable that it has a similar trend as that of the centralized one.

B. Number of messages for Network-wide Synchronization

Figure 8 presents the message overheads introduced by the proposed heuristic algorithm. We investigate how many messages are required to achieve network-wide synchronization in networks of different degrees of connectivity and different sizes. The protocol is scalable with network size as information is only exchanged between immediate neighbours. More importantly, it is also scalable with network densities that most wireless sensor networks possess. The number of

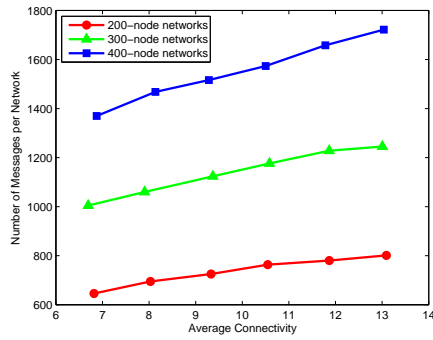


Fig. 8. Average number of messages transmitted to achieve network-wide synchronization

messages required grows modestly when the average number of neighbours is jumped from around 7 to 13. Even though increasing connectivity makes sensors have more neighbours to communicate with, it also reduces the number of PBS required to synchronize the whole network. It should be noted that the message overheads are one-off as long as the topology of the network does not change, but energy can be saved from each round of resynchronization. Using the results obtained from Figure 7 and Figure 8, we can estimate how many rounds of resynchronizations will offset the one-off overheads. For 400-node networks with average connectivity of 8.13, the average number of PBS required to achieve network-wide synchronization is 144.83 and the overheads from our heuristic-based algorithm are 1467.8 messages. If same networks are synchronized by TPSN, 399 pairwise synchronizations are required. Since each PBS and TPSN synchronization require two message exchanges, after above 3 rounds of synchronization, the overheads from our heuristic-based algorithm are cancelled out. In other words, after the third round of synchronizations, less energy is spent in resynchronization compared to TPSN. Table V-B gives the number of synchronization rounds required to cancel out the one-off overheads in networks with different sizes and degrees of connectivity. It can be seen that in all situations, the one-off overhead in the proposed algorithm is offset within 3 rounds of resynchronizations, when compared to TPSN.

VI. CONCLUSION

In this paper, we propose a distributed protocol for multi-hop time synchronization in wireless sensor networks using Pairwise Broadcast Synchronization (PBS). We justify our heuristic-based proposal by showing that finding the optimal synchronization set is NP-complete. The performance of the protocol is examined by extensive simulations. Results show that the protocol is scalable with network size and connectivity. In the future, the protocol will be tested in real sensor networks.

REFERENCES

- [1] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *Proc. of the 5th*

	Average Connectivity	Rounds of Synchronization
200-node Networks	6.8182	2.7135
	8.0285	2.7074
	9.3297	2.6537
	10.5475	2.6876
	11.8658	2.6462
	13.0920	2.6257
300-node Networks	6.6926	2.8841
	7.9042	2.8041
	9.3692	2.7759
	10.5837	2.7731
	11.8650	2.8040
	13.0059	2.7560
400-node Networks	6.8756	2.9155
	8.1360	2.8875
	9.2820	2.8215
	10.5025	2.8016
	11.7818	2.8484
	13.0370	2.8780

TABLE I

NO. OF SYNCHRONIZATION ROUNDS REQUIRED TO OFFSET THE ONE-OFF OVERHEADS WHEN COMPARING WITH TPSN

- annual ACM/IEEE International Conference on Mobile computing and networking (MobiCom 99)*, Seattle, Washington, USA, 1999, pp. 263–270.
- [2] I. F. Akyildiz, W. Su, and Y. Sankarasubramaniam, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, Mar. 2002.
- [3] J. Elson and K. Römer, "Wireless sensor networks: a new regime for time synchronization," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 149–154, 2003.
- [4] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Comput. Networks*, vol. 51, no. 4, pp. 921–960, 2007.
- [5] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Trans. Communications*, vol. 39, pp. 1482–1493, Oct. 1991.
- [6] K.-L. Noh and E. Serpedin, "Pairwise broadcast clock synchronization for wireless sensor networks," in *IEEE International Workshop: From Theory to Practice in Wireless Sensor Networks (T2PWSN 07)*, Helsinki, Finland, June 2007.
- [7] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 147–163, 2002.
- [8] M. Maróti, B. Kusy, G. Simon, and Ákos Lédeczi, "The flooding time synchronization protocol," in *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys 04)*. New York, NY, USA: ACM Press, 2004, pp. 39–49.
- [9] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. of the 1st International Conference on Embedded Networked Sensor Systems (SenSys 03)*, Los Angeles, California, USA, 2003, pp. 138–149.
- [10] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *Proc. IEEE Wireless Communications and Networking (WCNC 03)*, New Orleans, Louisiana, USA, Mar. 2003, pp. 1226–1273.
- [11] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Trans. Comput.*, vol. 55, no. 2, pp. 214–226, 2006.
- [12] R. M. Karp, "Reducibility among combinatorial problems," in *Proc. of a Symposium on the Complexity of Computer Computations*, New York, USA, Mar. 1972, pp. 85–103.
- [13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: The MIT Press, 2001.