A Distributed Multihop Time Synchronization Protocol for Wireless Sensor Networks using Pairwise Broadcast Synchronization

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

***Abstract*—Recently, a time synchronization algorithm called Pairwise Broadcast Synchronization (PBS) is proposed. With PBS, a sensor can be synchronized by overhearing synchronization packet exchange among its neighbouring sensors without sending out any packet itself. In an one-hop sensor network where every node is a neighbour of each other, a single PBS message exchange between two nodes would facilitate all nodes to synchronize. However, in a multi-hop sensor network, PBS message exchanges in several node pairs are needed in order to achieve network-wide synchronization. To reduce the number of message exchanges, these node pairs should be carefully chosen. In this paper, we investigate how to choose these "appropriate" sensors aiming at reducing the number of PBS message exchanges while allowing every node to synchronize. This selection problem is shown to be NP-complete, for which the greedy heuristic is a good polynomial-time approximation algorithm. Nevertheless, a centralized algorithm is not suitable for wireless sensor networks. Therefore, we develop a distributed heuristic algorithm allowing a sensor to determine how to synchronize itself based on its neighbourhood information only. The protocol is tested through extensive simulations. The simulation results reveal that the proposed protocol gives consistent performance under different conditions with its performance comparable to that of the centralized algorithm.**

***Index Terms*—Time synchronization, sensor networks.**

I. Introduction

With the advance in various enabling technologies including the 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]. 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 size, services which can be readily provided in traditional wired networks have to be re-designed for WSNs. Time synchronization is one of these services [3]. In applications like habitat monitoring [4], 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 such information together, the detection of an event is meaningless to many applications. A consistent notion of time is also important to the emerging wireless multimedia sensor networks [5] where precise physical timing is required for multimedia data streaming in real time. Furthermore, time synchronization is also crucial to the duty cycles scheduling for MAC protocols of WSNs [6] [7] and in-network processing [8].

Time synchronization has been studied for a long time in distributed systems over conventional networks. The Network-Time-Protocol (NTP) [9] is the de-facto time synchronization protocol in the Internet. Since sensor nodes are battery-powered, energy consumption becomes a major consideration if NTP is implemented in sensors. Sensors near the reference node will become hot spots as they relay time synchronization packets. As a result, the network lifetime will be affected substantially.

Much effort has been spent in synchronizing a pair of nodes within a network. Most synchronization protocols for WSNs are based on two fundamental approaches: Sender-to-Receiver Synchronization (SRS) and Receiver-to-Receiver synchronization (RRS) [10].

SRS is based on the traditional approach adopted in NTP. A node initiates a two-way message exchange with the reference node when it wants to be synchronized. The first SRS-based protocol is the Timing-sync Protocol for Sensor Networks (TPSN) [11]. In this protocol, nodes are synchronized in a pairwise manner. The clock offset between the sender and the receiver are estimated based on exchanged timestamps. Another example of SRS-based protocol is the Tiny-Sync and Mini-Sync in [12]. The mechanism is similar to TPSN that sender and receiver exchange timing packets. The timestamps are used to establish bounds on the relative clock skew and offset. Since TPSN, Tiny-Sync and Mini-Sync are all pairwise in nature, every node in a network needs to exchange messages with another node, which requires a considerable amount of energy for it.

RRS does not require the reference node to exchange timing packets with other sensors. Only unsynchronized nodes exchange the timing packets. Elson *et al.* [13] proposed a RRS-based protocol called Reference Broadcast Synchronization (RBS). The idea is similar to a synchronization algorithm in broadcast LANs [14]. A beacon node uses the broadcast channel to send beacons to receivers. Receivers mark the reception time and exchange the timestamps with other receivers upon receiving the beacons. The relative clock skew and offset

Manuscript received January 25, 2008; revised June 26, 2008 and December 7, 2008; accepted December 22, 2008. The associate editor coordinating the review of this paper and approving it for publication was R. M. Buehrer.

The authors are with the Department of Electrical and Electronic Engineering, The University of Hong Kong (e-mail:{kycheng, kslui, ycwu, vtam}@eee.hku.hk).

This work was supported in part by the General Research Fund, Project No. HKU 7181/07E and the HKU Small Project Fund, Project No. 200507176083.
Digital Object Identifier 10.1109/TWC.2009.080112

are estimated by linear regression on the timestamps. Despite the simple procedures on the receiver side, providing periodic reference pulses is costly in terms of energy, especially in the multihop scenario. Special nodes have to be provided to transmit the reference pulses or sensors have to take turns to broadcast the reference pulses periodically. However, both methods do not scale well to large sensor networks.

Although these two approaches enable pairwise synchronization in WSNs, extending pairwise time synchronization to network-wide time synchronization in an energy-efficient manner is still a difficult issue in WSNs. Recently, Noh *et al.* [15] proposed the third approach for synchronization in WSNs, called Pairwise Broadcast Synchronization (PBS). PBS uses the broadcast nature of wireless channels to synchronize nodes within the same broadcast domain. Two super nodes, A and P, exchange timing information like TPSN. Due to the broadcast nature of the communication channels, another node in the neighbourhood (say Node B) can overhear the message exchange between Node A and Node P. When the broadcast messages arrive at Node B, Node B marks the arrival times and obtains the timestamps broadcasted by Nodes 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. Thus every node that overhears the exchange of timing information can synchronize itself without sending a message. This greatly reduces the number of messages required for synchronizing nodes within a broadcast domain. Energy can be saved since receiving a message usually consumes less energy than transmitting one. Furthermore, it is shown [15] that PBS achieves the same synchronization accuracy as RBS.

In [16], Noh *et al.* devised two extensions to the PBS for multihop synchronization. Both extensions are greedy algorithms. The extensions aim to achieve energy-efficient network-wide synchronization by minimizing the number of message exchanges performed. The first extension is called the Network-wide Pair Selection Algorithm (NPS) and is a centralized algorithm. A network hierarchy similar to that of TPSN [11] is first built. The node with the reference time acts as the root. PBS are performed in a sequential manner along the hierarchy from the top (*root*) to the bottom. Nodes of higher levels synchronize nodes of lower levels. The pair of nodes with the maximum number of unsynchronized neighbours in their common coverage are chosen to perform message exchanges. The selection process iterates until all nodes are synchronized. Since the selection process requires global information, a large amount of message overhead will be introduced if it is applied in a large WSN. It violates the objective of performing network-wide synchronization in an energy-efficient manner. To circumvent this, another extension called the Group-wise Pair Selection Algorithm (GPS) was proposed. The algorithm resembles NPS except that decisions are made locally by nodes of higher levels using local connectivity information. Thus, GPS greatly reduces the message overhead when synchronization pairs are being determined. However, the performance of GPS during actual synchronization is a lot worse than that of NPS as many unnecessary PBS are performed when comparing with NPS. In later sections, a detailed treatment is given to address the pair selection problem. A distributed protocol is presented which requires only local information exchange and manages to reduce the number of pairwise synchronizations for network-wide synchronization.

In [17], we argued that selecting the minimum number of nodes to participate PBS message exchanges is an NP-complete problem. We also developed a distributed multi-hop synchronization protocol which significantly reduces the number of PBS message exchanges for network-wide synchronization. In this paper, we provide formal NP-completeness proofs to the problem. We also investigate the performance of our protocol through computational complexity analysis and extensive simulations. We compare our protocol to TPSN and GPS in networks of various topologies.

It is worth mentioning that Maroti *et al.* [18] proposed a protocol called Flooding Time Synchronization Protocol (FTSP) which does not incorporate any two-way message exchange. Unfortunately, the timing information is distributed by flooding synchronization packets across the network and this process consumes a large amount of energy. On the other hand, Li *et al.* [19] proposed two global synchronization algorithms. Clocks are updated according to the time differences between sensors. However, unlike the previous algorithms, clocks are not eventually synchronized to a reference time but to the average value, thus limiting its usefulness.

The paper is organized as follows. The network-wide synchronization problem with PBS is analyzed in Section II. It is demonstrated that synchronizing all sensors with the minimum number of PBS operations is an *NP-complete* problem. In Section III, a distributed PBS-based network-wide synchronization protocol is presented. Our proposal is justified by extensive simulations in Section IV and we conclude our paper in Section V.

II. Problem Formulation

The communication range of a sensor node is relatively small when compared to the dimensions of the whole wireless sensor network. Thus, it is almost impossible to synchronize all nodes by performing a single PBS. A multi-hop time synchronization protocol is needed so that all nodes can synchronize to a reference time by performing a number of PBS. Usually, the reference time is simply the clock of a normal sensor node. A trivial protocol is successive synchronizations. Nodes that are immediate neighbours of 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 fully exploit the advantage of PBS. Nodes overhearing a complete PBS message exchange can synchronize themselves. When the pairs of nodes performing PBS message exchanges are carefully chosen, the number of PBS message exchanges can be reduced, thus saving more energy. Notice that in this paper, although we describe our protocol based on the PBS time synchronization mechanism, our protocol works with any synchronization scheme that allows a node to synchronize by overhearing the messages exchanged by a pair of neighbouring nodes.

Figure 1 illustrates an example. Two nodes are neighbours if there is a link between them. Let Node 1 be the reference node.

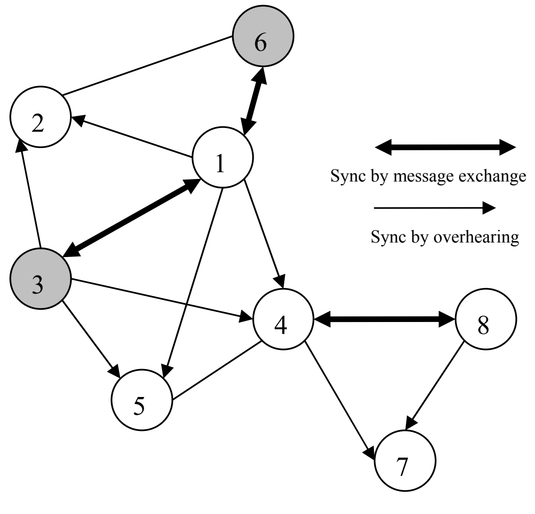


Fig. 1. Network-wide PBS synchronization for a simple network.

The rest of the nodes are unsynchronized. If Node 1 knows the connectivity information, Node 1 can reduce the number of PBS operations by exchanging messages with the node which has the maximum number of common unsynchronized neighbours with Node 1. Therefore, Node 1 will first synchronize Node 3 by message exchange. This also makes Node 2, Node 4 and Node 5 become synchronized by overhearing. Afterwards, Node 6, Node 7 and Node 8 are left unsynchronized because they cannot overhear both Nodes 1 and 3. Since Node 6 is the only neighbour of Node 1 that is not synchronized, it will be synchronized in the next PBS message exchange with Node 1. Notice that Node 6 can be synchronized with Node 2 after Node 2 is synchronized, but synchronizing with Node 1 is preferable. It is because in general synchronization errors will be accumulated. Finally, since Node 7 and Node 8 are not one-hop neighbours of Node 1, they are synchronized by Node 4 after Node 4 is synchronized by Node 1. Node 8 is synchronized by exchanging packets with Node 4 while Node 7 is synchronized by overhearing.

Although suitable synchronization pairs can be found by trial and error, it may not be the best in terms of the number of PBS message exchanges performed. 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 node and unnecessary PBS message exchanges may be performed if the selection on the synchronization pair is made by each synchronized node independently. Reference nodes do not realize that some nodes are already synchronized by other reference nodes as synchronized nodes do not exchange information. PBS message exchanges may still be carried out to synchronize those already synchronized sensors. This explains why the Group-wise Pair Selection Algorithm [16] fails to deliver satisfactory results.

To have a more complete analysis, we look at the network-wide synchronization problem from a centralized point of view. Nodes are assigned to different levels which represent the hop count from the reference node. If a node is not a neighbour of the reference node, it has to rely on other synchronized node to perform synchronization. However, synchronization error accumulates along the path where successive synchronizations are performed. To minimize the synchronization error, level $(i+1)$ nodes should only be synchronized by nodes of level i. This means that level $(i+1)$ node either synchronizes directly with a node on level i by message exchange or overhears the messages exchanged between a level i node and a neighbour of level $(i+1)$. The network-wide 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 using the minimum number of PBS message exchanges. In level 0, there is only one node which is the reference node. The problem of synchronizing level 1 nodes is a special case of the synchronization of other levels. We first analyse the simpler level 1 problem and then analyse the problem of other levels. We denote the problem with one reference node as Single Reference Synchronization Problem (SRSP) and problem with multiple reference nodes as Multiple Reference Synchronization Problem (MRSP).

A. Single Reference Synchronization Problem (SRSP)

The network is modelled as a graph $G(V,E)$ where V is the set of points representing the nodes in the network. One node in V is the reference node while the others are unsynchronized nodes. 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. The packets broadcasted by the reference node can be overheard by every unsynchronized nodes (but it may not be true to the packets broadcasted by the unsynchronized nodes). Links are symmetric and so $(v_i,\ v_j) \in E$ implies $(v_j,\ v_i) \in E$. The Single Reference Synchronization Problem, described in the language of theory of computation, is

$$\begin{aligned} \text{SRSP} \ = \{ \langle G, v_r, k \rangle :& \\ & G=(V,E) \text{ is an undirected graph,} \\ & v_r \in V \text{ is the reference node,} \\ & (v_r, v_i) \in E \text{ if } v_i \in V \text{ and } v_i \neq v_r, \\ & k \in Z, \text{ and} \\ & v_r \text{ synchronizes all nodes in } V \text{ with} \\ & \quad \text{at most } k \text{ PBS message exchanges} \}. \end{aligned}$$

Theorem 1: **Single Reference Synchronization Problem is NP-complete.**

Proof: We call the set of nodes which are directly synchronized by v_r as *Synchronization Set* and denoted it as V'. $V' \subseteq V \backslash \{v_r\}$. We choose the Synchronization Set as the certificate of an SRSP. The verification algorithm first checks whether $|V'| \leq k$. Afterwards, for each $v_i \in V$, the algorithm verifies whether v_i is synchronized. Specifically, it checks if any one of the following conditions is met.

- $v_i \in V'$, i.e., v_i is directly synchronized by v_r.
- $(v_i, v_j) \in E$, for some $v_j \in V'$, i.e., v_i is synchronized by overhearing the synchronization packets exchanged between v_j and v_r.

This verification can be run in polynomial time. Hence, *SRSP* $\in$ NP. Next, we prove that every dominating set problem can be reduced to an SRSP instance. An instance of the dominating set problem consists of an undirected graph, $G(V,E)$. A dominating set, D, is a subset of V such that for every $v_i \in V$, there exists an edge $(v_i, v_j) \in E$ where $v_j \in D$

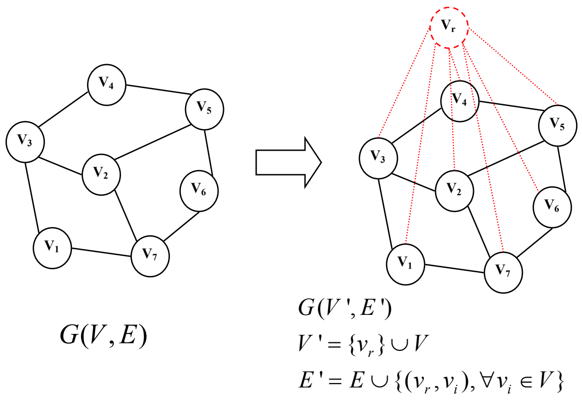


Fig. 2. Reducing a dominating set problem to a SRSP problem.

or $v_i \in D$. Finding a dominating set D with size at most k is *NP-complete*. Given a dominating set problem instance $\{\langle G, k\rangle\}$, we reduce it to an SRSP instance $\{\langle G', v_r, k\rangle\}$ that the SRSP instance has a Synchronization Set of size at most k if and only if the corresponding dominating set problem has a dominating set of size at most k. G' is constructed as follows:

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

An example of the reduction is illustrated in Figure 2. A reference node v_r is added and it is connected to every v_i in V. The reduction is straightforward and can be done in polynomial time. Furthermore, a dominating set D of size k exists for the dominating set problem if and only if a Synchronization Set of size k exists for the SRSP. Given a dominating set D of size k, it is the Synchronization Set of size k for the reduced SRSP as one of the conditions given above must be satisfied. On the other hand, given a Synchronization Set of size k of a reduced SRSP, by a similar argument, it is also the dominating set for the dominating set problem.

■

B. Multiple Reference Synchronization Problem (MRSP)

In the SRSP problem, there is only one reference node with which other nodes can exchange timing information. However, it is very likely that a level $(i+1)$ node can be synchronized by more than one node in level i, for $i > 0$. In this section, we analyze the Multiple Reference Synchronization Problem (MRSP). The topology of level i nodes and level $(i+1)$ nodes is represented as an undirected graph $G(V, E)$. $V = V_M \bigcup V_N$ is the set of vertices where $V_M = \{m_1, ..., m_M\}$ are nodes of level i and $V_N = \{n_1, ..., n_N\}$ are the nodes of level $(i+1)$. 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. It should be noted 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 since only nodes in V_M are synchronized. When message exchange 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. The Multiple Reference Synchronization Problem (MRSP) is

MRSP $= \{\langle G, k\rangle$:
$G = (V, E)$ is an undirected graph,
$V = V_M \bigcup V_N$, V_M and V_N are the sets of level i and $(i+1)$ nodes, respectively,
for all $n_k \in V_N$, there is some $m_j \in V_m$ such that $(m_j, n_k) \in E$,
$k \in Z$, and
nodes in V_N are synchronized by performing at most k PBS operations with nodes in $V_M\}$.

Theorem 2: **The Multiple Reference Synchronization Problem is NP-complete.**

Proof: We denote the pair of nodes performing a PBS operation as a tuple, $t(m_j, n_k)$ where m_j is a sensor of level i and n_k is a sensor of level $(i+1)$. Sensors n_k and m_j exchange timing information and n_k is directly synchronized. We use the set of tuples called the Synchronization Tuples, T_S, as a certificate to the MRSP. The verification algorithm is similar to the SRSP. It firstly confirms that $|T_S| \leq k$. Afterwards, it checks whether all level $(i+1)$ nodes are synchronized. Thus, at least one of the following conditions has to be satisfied for n_l which is a level $(i+1)$ node.

- n_l appears in at least a tuple of T_S, i.e., $\exists t(m_j, n_l) \in T_S$ and $m_j \in V_M$.
- There is at least a tuple, $t(m_j, n_k)$, such that n_l is a neighbour of both m_j and n_k , i.e., $\exists t(m_j, n_k) \in T_S$, $m_j \in V_M$ and $n_k \in V_N$, $(m_j, n_l), (n_l, n_k) \in E$.

The verification algorithm can be run in polynomial time and MRSP $\in$ NP. Next, we further prove that every set-covering problem instance can be reduced to an MRSP instance. A set-covering problem instance (X, F, k) consists of a universe X and a family of subsets F. Each element of X belongs to at least one subset in F, i.e.,

$$X = \bigcup_{S \in F} S.$$

The set-covering problem asks whether there is a subset of F of size at most k that covers all the elements of X. It is an NP-complete problem [20]. Let (X, F, k) be a set-covering problem instance which $X = \{x_1, ..., x_{|X|}\}$ is the universe, and $F = \{S_1, S_2, ..., S_{|F|}\}$ is the family of subsets. The MRSP instance is constructed as follows. Each subset S_j is mapped to a level i node, m_j. Each element x_k is mapped to a node of level $(i+1)$, n_k. Hence, $V_M = \{m_1, ..., m_{|F|}\}$ and $V_N = \{n_1, ..., n_{|X|}\}$. For each element x_k in each S_j, an edge (n_k, m_j) is added to E. Moreover, assume that there is an arbitrary order for elements in subset S_j and let x_{j1} be the first element in S_j. For each element x_k in S_j, an edge (n_k, n_{j1}) is added to E where n_{j1} is the node mapped by x_{j1} and $x_{j1} \neq x_k$. An instance of MRSP is then constructed. There are $(|X| + |F|)$ vertices and $O(|F||X|)$ edges. The reduction can be completed in polynomial time. Figure 3 illustrates an example of the reduction. We claim that C, a subset of F, covers all elements of X if and only if $T_S = \{t(m_j, n_k) \mid n_k$

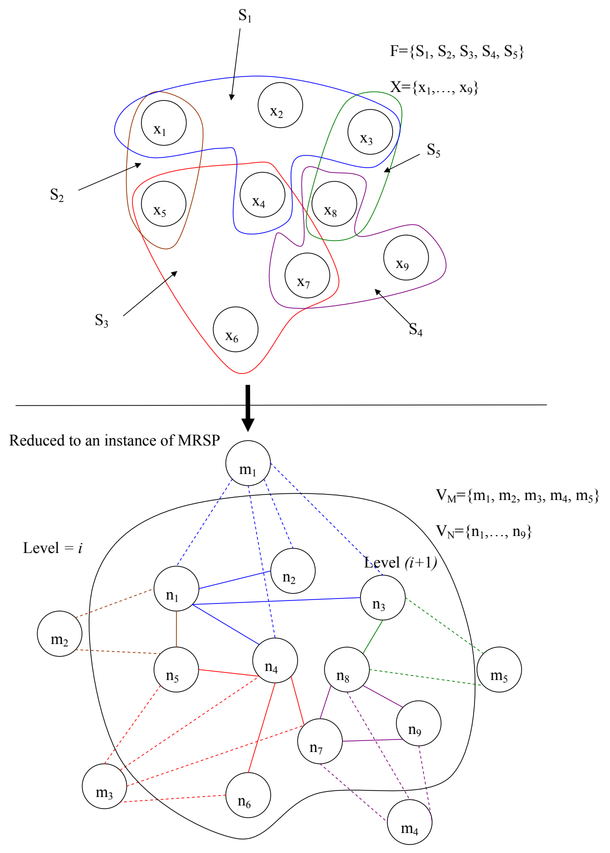


Fig. 3. An MRSP instance reduced from a set-covering problem instance.

maps to x_k and x_k is the first element in S_j where $S_j \in C\}$ is the solution of the reduced MRSP.

$\Longrightarrow$ Let $C \subseteq F$ be a cover of size at most k for a set-covering problem instance. All elements x_k must be resided in at least one subset S_j of C. It implies that all nodes n_k must be connected to at least one of the n_{j1} which 'represents' the first element x_{j1} of a S_j. Thus, all sensors can be synchronized by at most k PBS operations carried out by the sensors $t(m_j, n_{j1})$.

$\Longleftarrow$ Let $T_S = \{t(m_j, n_k)$; for some j and k, $1 \le j \le |F|, 1 \le k \le |X|\}$ be the Synchronization Tuples that all sensors n_k can be synchronized by at most k PBS operations. If a level $(i+1)$ node is to be synchronized, it has to be connected to at least one level i node that carries out the PBS operation. It means that every n_k is connected to at least one m_j appears in T_S and thus every x_k belongs to at least one S_j that corresponds to m_j. Hence, a cover of size at most k can be found. ■

III. A Distributed Multihop Synchronization Protocol for WSNs

In the previous section, it is clearly shown that the problem of finding the minimum number of node pairs performing PBS message exchanges to achieve network-wide synchronization is an NP-complete problem, no matter which levels of nodes are considered. Although no polynomial-time algorithm has been found for the optimal solution, sub-optimal solutions for dominating set problems and set cover problems can be obtained by approximation algorithms [21], such as greedy algorithms.

Using a greedy algorithm, message exchanges are carried out iteratively by the pair of nodes which synchronize the largest number of unsynchronized sensors until all sensors are synchronized. Nevertheless, this algorithm is centralized in nature. Global connectivity information about sensors has to be known in order to determine which pair of nodes should perform message exchanges. In practice, the centralized greedy algorithm can only be used by the reference node to solve the SRSP since the information required is confined in a local region around the reference node. However, a large amount of communication overhead will be incurred if the centralized greedy algorithm is applied in other levels, i.e., the MRSP. The connectivity information between different reference nodes has to be known as well. Moreover, the information has to be sent to a dedicated node which runs the centralized greedy algorithm to determine the solution. Finally, the decision has to be distributed back to the sensors. All these procedures will inevitably introduce a substantial amount of communication overhead. The overhead overshadows the energy saved by using the PBS.

In view of this, we propose a distributed heuristic-based protocol for network-wide 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 TPSN [11]. That is, the reference node sends out a message to inform its neighbours that they are at level 1. Each level 1 node then declares its level to its neighbours. Some neighbours are also at level 1 and they ignore this message. Those neighbours who have not known their levels yet now realize that they are at level 2. The process continues until each node determines its level. After the level discovery process, nodes of lower level, say level $(i+1)$ send the local connectivity information to their neighbours of the higher level, i.e., level i. With the connectivity information of levels i and $(i+1)$, each node of level i can determine how many nodes of level $(i+1)$ can be synchronized by one message exchange. The numbers are exchanged between level i neighbours. Only the node which synchronizes the largest number of level $(i+1)$ nodes will perform the PBS operation. The ID of nodes that are synchronized will be distributed to other level i nodes. The process repeats until all level $(i+1)$ nodes are synchronized. The details of the protocol are given below:

(1) Nodes use the neighbour information obtained from the level discovery phase to determine the number of neighbours and their IDs. 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_j, is created by each sensor j of level i. The list contains the IDs of all level i neighbours of sensor j. Every sensor j of level i sends the list L_j to its level $(i-1)$ neighbours to notify them which level i sensors are their neighbours. Meanwhile, they also receive corresponding neighbour lists from its neighbours of level $(i+1)$. To reduce the complexity of later steps, the neighbour list is sorted by the IDs of sensors. Therefore, the complexity of this step is $O(m \log m)$, where m is the number of level i neighbours of Node j.

(2) For every Node j of level i, after receiving all neighbour

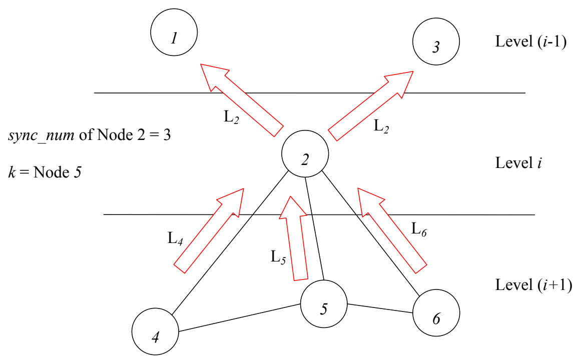
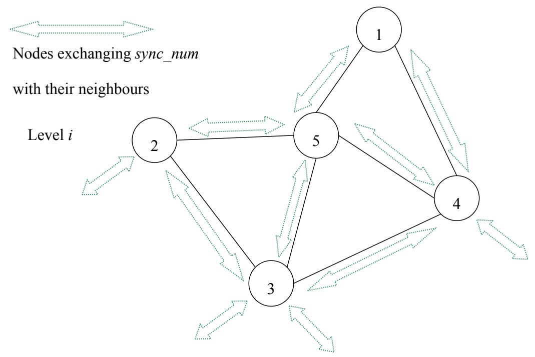


Fig. 4. Nodes exchanging neighbour lists and determining $sync\_num$.



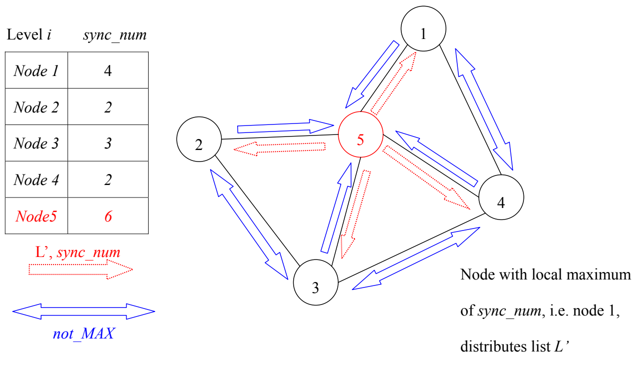


Fig. 5. Local information exchange.

lists from its level $(i+1)$ neighbours, Node j enumerates the lists and checks which node shares the maximum number of common neighbours of level $(i+1)$ with itself. Let $\{L_1, L_2, ..., L_n\}$ be the neighbour lists received by Node j and j's level $(i+1)$ neighbour list is N_j. Node j can determine the common neighbours shared with its level $(i+1)$ neighbour Node k by comparing its neighbour list with that of node k, i.e., $N_j \cap L_k$. These are the nodes that can overhear the PBS message exchange between j and k. After checking all the lists, Node j can determine the maximum number of its neighbours that can be synchronized by **one** PBS message exchange. Let that number be $sync\_num$ of Node j, i.e., $sync\_num = \max_{1 \le k \le n} ||N_j \cap L_k||$ and that particular level $(i+1)$ neighbour be Node k (see Figure 4). The $sync\_num$ will be sent to all level i neighbours of Node j.

In this step, since the neighbour lists are all sorted, the complexity of comparing one neighbour list is $O(n)$ where n is the number of level $(i+1)$ neighbours of Node j. Hence, the complexity of comparing all neighbour lists received by Node j is $O(n^2)$. If the lists are not sorted, the complexity will be $O(n^3)$.

(3) Node j will eventually receive all the $sync\_num$ of its level i neighbours. Node j checks whether its $sync\_num$ is the largest among the received $sync\_num$. There are two cases.

Case 1: The sync_num of Node j itself is the largest among the received sync_num.

Node j sends each level i neighbour a list, L', which contains the IDs of the sensors that can be synchronized by performing one PBS message exchange. List N_j is updated as $N_j = N_j - L'$, i.e., the synchronized nodes will be removed from the neighbour list of Node j. Hence, $sync\_num$ is also updated and it must be smaller than the old value. The updated $sync\_num$ of Node j is also distributed together with the L'. If the updated $sync\_num$ equals 0, Node j will jump to Step (5). Thus, neighbours receiving the $sync\_num$ of value 0 are indirectly notified that they do not need to wait for any update from Node j anymore. The ID of Node k is put into the synchronization list of Node j which stores the IDs of level $(i+1)$ nodes that exchange timing packets with Node j later.

Case 2: The sync_num of Node j itself is NOT the largest among the received sync_num.

Node j sends a packet not_MAX to its level i neighbours (see Figure 5). Then, Node j waits for replies from its neighbours and there are two possible scenarios.

- *Case 2a: Node 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 N_j will be updated as $N_j = N_j - (\bigcup L')$, i.e., nodes synchronized by the sender of L' are removed from the list N_j. A new $sync\_num$ can be determined. The new $sync\_num$ will be distributed to all level i neighbours of Node j. If the updated $sync\_num$ equals 0, Node j jumps to Step (5) after sending the $sync\_num$. As mentioned before, neighbours of Node j will not wait for any update from Node j afterwards.
- *Case 2b: Node j only receives packets not_MAX from all of its level i neighbours.*
  The $sync\_num$ of Node j remains unchanged as Node j and all its neighbours find themselves not having the largest $sync\_num$. Node j will send its $sync\_num$ to all of its level i neighbours again. This scenario is possible since the node which Node j thinks to have the maximum $sync\_num$ may not find itself having the maximum $sync\_num$. That particular node and Node j have two different neighbour sets.

In this step, the checking of $sync\_num$ has a complexity of $O(m)$ where m is the number of level i neighbours of Node j. In Case 1, the removal of synchronized nodes does not introduce additional complexity as the removal can be

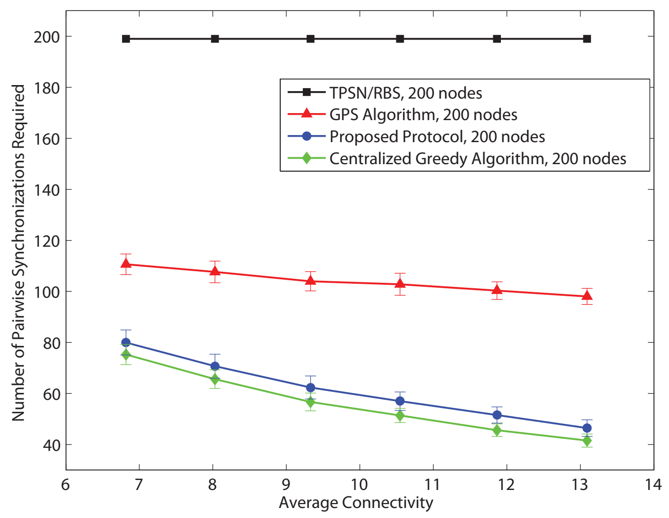


Fig. 6. Average number of PBS required for network-wide synchronization with 200-node unit-disk networks.

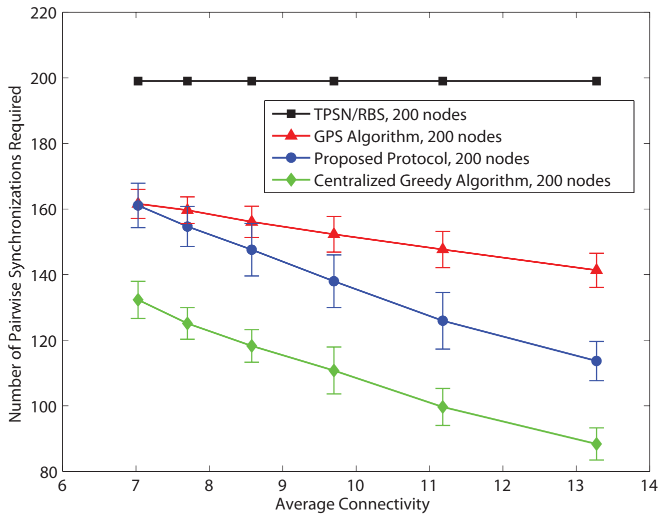


Fig. 7. Average number of PBS required for network-wide synchronization with 200-node non-unit-disk networks.

done when the list N_j is compared with other neighbour lists in Step (2). In Case 2a, the removal process has a complexity of $O(|\bigcup L'|) \leq O(m \times n)$. In Case 2b, Node j only sends the $sync\_num$ again which has a constant complexity of $O(1)$.

(4) Step (3) will be reiterated by Node j until N_j becomes an empty list.

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

All level $(i + 1)$ sensors must be synchronized eventually as there exists at least one sensor which finds itself having the maximum $sync\_num$ after Step (3). The total complexity of Step (1) to Step (3) is $\{O(m \log m) + O(n^2) + O(m) + O(m \times n)\}$ or $\{O(m \log m) + O(n^2) + O(m) + O(1)\}$ depending on the magnitude of the $sync\_num$. Since n and m should be comparable, the total complexity is $O(n^2)$, which is comparable to many other sensor network protocols.

Although a certain number of messages are transmitted to determine the synchronization set, the overhead is one-off. The synchronization set remains the same if the network topology does not change. Nodes simply perform Step (5) for resynchronization. Furthermore, the overhead can be reduced by piggybacking the timesync data when nodes are determining the synchronization set but detailed discussion is beyond the focus of this paper.

IV. Simulation Results and Discussions

To justify the proposed protocol, substantial simulations are performed. As proved in [15], PBS gives synchronization results as accurate as RBS, we do not focus on the timing accuracy. Instead, we examine how many rounds of pairwise message exchanges are required to obtain network-wide synchronization in a WSN. We also investigate the overhead incurred by the protocol and compare it with TPSN and RBS.

We have performed simulations in 200-node and 400-node networks where the network topologies are square, square with large obstacles, and C-shaped. As the performance trends are similar, in this paper, when we compare the performance of our protocol with other protocols, we present only the results of 200-node square networks. We generated both unit-disk graphs and non-unit-disk graphs in the square networks. In unit-disk graphs, it is assumed that sensor x can communicate with sensor y if they are within the communication range of each other (i.e., $||x - y|| \leq R$). Only bi-directional links are considered. The 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. For non-unit disk graphs, two nodes can communicate if their distance d is less than 1. If $d > 1$, the probability that they can communicate is $1/kd^2$ where k is a constant. We adjust k to get different degrees of connectivity.

A. Number of Pairwise Synchronizations for Network-wide Synchronization

Figures 6 and 7 give the average numbers of pairwise message exchanges required to achieve network-wide synchronization in 200-node unit-disk and non-unit-disk networks, respectively. Each data point gives the average result of 60 different network instances. The vertical bar plotted on each data point represents the corresponding standard deviation. We compare the results with that obtained by the Groupwise Pair Selection (GPS) Algorithm [16] (a distributed algorithm) and the centralized greedy algorithm. Note that the centralized greedy algorithm gives a benchmark on the required message exchanges for a particular network topology. From the figures, our proposed protocol always outperforms the GPS. In unit-disk networks, the performance of our protocol is close to that of the centralized counterpart. The improvement against GPS is between *27%* to *52%* for different connectivities. In non-unit-disk topologies, our protocol outperforms GPS more when the connectivity is higher. When each node has 14 neighbours on average, our protocol saves more than *20%* of PBS exchanges.

If TPSN is used to synchronize a n-node network, each node except the root node must synchronize with another

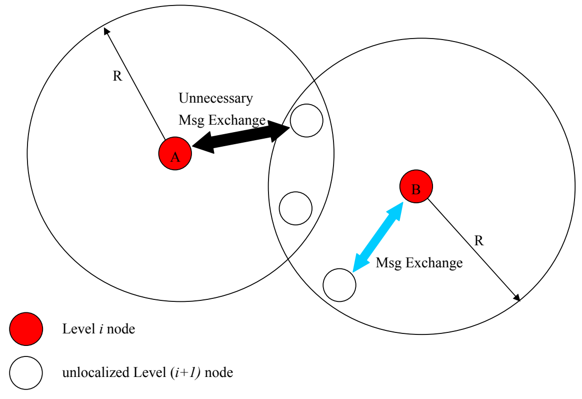


Fig. 8. Unnecessary PBS happens in non-unit-disk networks.

node, giving $n-1$ pairwise synchronizations irrespective of node density. Similar argument holds in the case of RBS [13] as it is also a pairwise-based synchronization algorithm in which receivers exchange timestamps after receiving the beacon. Therefore, TPSN/RBS requires *199* pairwise message exchanges in a 200-node network. The proposed protocol is much more efficient in terms of the number of pairwise message exchanges. It is also worth noting that, the number of PBS required drops when connectivity is increased as more neighbours can be synchronized by one PBS, while for TPSN/RBS, the number of message exchange remains constant regardless of the degree of connectivity.

Notice that there is a gap between the centralized algorithm and the proposed distributed algorithm, particularly in the case of non-unit-disk. It is because "unnecessary" synchronizations are more likely to occur in non-unit-disk networks. A simplified scenario of "unnecessary" synchronization is illustrated in Figure 8. Both Node A and Node B will perform PBS to synchronize the level $(i+1)$ nodes. Synchronization performed by Node A is unnecessary since Node B can synchronize all nodes with one message exchange. However, Node A does not realize that Node B will synchronize all nodes with one message exchange as they cannot communicate with each other. Similar scenarios appear more frequently in non-unit-disk networks as connectivity does not solely depend on the communication range. "Unnecessary" synchronization is likely to occur when two nodes of the same level are close to each other but cannot communicate. If one sensor finds that it can synchronize more nodes by one message exchange than its neighbours, the node nearby (but cannot communicate with) will also do so as their neighbouring nodes are almost identical. This scenario does not appear in unit-disk networks.

B. Message Overhead for Network-wide Synchronization

Although the proposed algorithm outperforms the GPS algorithm in terms of the number of message exchanges, it brings additional communication overhead when information is exchanged between sensors to determine which pairs of sensors to perform message exchanges. To justify the proposed algorithm, we investigate how many overhead messages are required to select nodes to exchange PBS messages in the whole network.

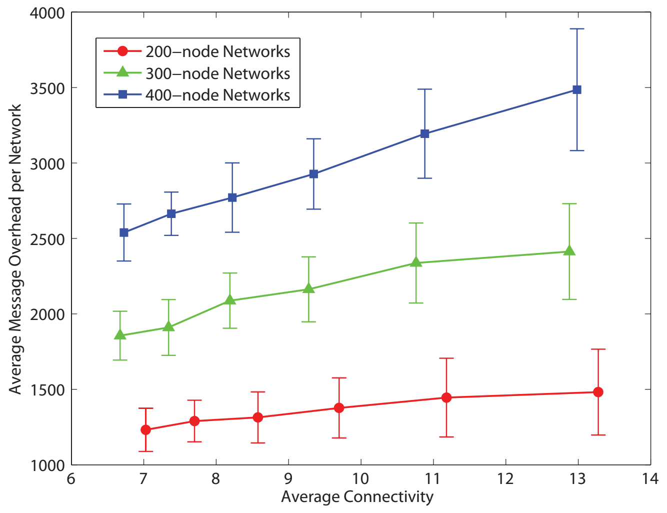


Fig. 9. Average number of overhead messages for the proposed protocol in non-unit-disk networks.

Figure 9 shows the message overhead introduced by the proposed algorithm in networks with different degrees of connectivity and network sizes in non-unit-disk networks. Since information is only exchanged between neighbours, the protocol is scalable with network size. More importantly, the protocol is also scalable with network densities. As Figure 9 shows, the overhead grows only linearly with the average connectivity. Although 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 while periodic re-synchronization is needed in the long run, the message overhead is one-off as long as the topology of the network does not change, and significant amount of energy can be saved from each round of resynchronization. For example, suppose that in each round of time synchronization, the nodes have to perform 4 two-way message exchanges (8 messages). From Figure 9, a 200-node non-unit-disk network with average connectivity of *9.7* requires about 1400 overhead messages. According to Figure 7, our protocol can save about 60 pairs of PBS exchanges, which is $60 \times 8 = 480$ messages, when compared with TPSN in each round of synchronization. Therefore, the overhead of our protocol will be compensated by the saving in timestamp exchange after 3 rounds of time synchronization.

V. Conclusions

PBS enables sensors to synchronize themselves by overhearing a complete exchange of synchronization packets from their neighbours. This can greatly reduce the energy consumed by time synchronization. Nonetheless, for multi-hop networks, there is a lack of distributed protocol to determine which sensors should perform message exchange or overhearing. It is formally shown in this paper that finding the minimum number of pairwise message exchange in PBS for network-wide synchronization is an NP-complete problem. In view of this, a distributed protocol for multi-hop time synchronization using PBS is presented in this paper. The distributed protocol is inspired by the greedy algorithm. The performance of the protocol is tested by extensive simulations. Results revealed that the proposed protocol outperforms the Groupwise Pair Selection Algorithm [16] in terms of the number of pairwise message exchange. Compared with the TPSN and RBS, the proposed protocol is shown to be more energy-efficient in static networks. Furthermore, the protocol is scalable with network size and connectivity.

REFERENCES

[1] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *Proc. 5th 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] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, USA, 2002, pp. 88–97.

[5] 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.

[6] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. IEEE INFOCOM*, New York, NY, USA, 2002, pp. 1567–1576.

[7] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, "Delay efficient sleep scheduling in wireless sensor networks," in *Proc. IEEE INFO-COM*, Miami, FL, USA, 2005, pp. 2470–2481.

[8] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Networking*, vol. 11, no. 1, pp. 2–16, 2003.

[9] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Trans. Commun.*, vol. 39, pp. 1482–1493, Oct. 1991.

[10] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005.

[11] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 1st International Conference on Embedded Networked Sensor Systems (SenSys 03)*, Los Angeles, CA, USA, 2003, pp. 138–149.

[12] 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, LA, USA, Mar. 2003, pp. 1226–1273.

[13] 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.

[14] P. Verissimo and L. Rodrigues, "A posteriori agreement for fault-tolerant clock synchronization on broadcast networks," in *Proc. Twenty-Second International Symposium on Fault-Tolerant Computing (FTCS'92)*, Boston, MA, USA, July 1992, pp. 527–536.

[15] K.-L. Noh, E. Serpedin, and K. A. Qaraqe, "A new approach for time synchronization in wireless sensor networks: pairwise broadcast synchronization," *IEEE Trans. Wireless Commun.*, vol. 7, no. 9, pp. 3318–3322, Sept. 2008.

[16] K.-L. Noh, Y.-C. Wu, K. Qaraqe, and B. Suter, "Extension of pairwise broadcasting clock synchronization for multi-cluster sensor networks," *EURASIP J. Advances in Signal Processing*, special issue on Distributed Signal Processing Techniques for Wireless Sensor Networks, vol. 2008.

[17] K.-Y. Cheng, K.-S. Lui, Y.-C. Wu, and V. Tam, "A greedy distributed time synchronization algorithm for wireless sensor networks," in *Proc. IEEE International Conference on Communications (ICC)*, May 2008, pp. 2327–2331.

[18] M. Maróti, B. Kusy, G. Simon, and Ákos Lédeczi, "The flooding time synchronization protocol," in *Proc. 2nd International Conference on Embedded Networked Sensor Systems (SenSys 04)*, New York, NY, USA, 2004, pp. 39–49.

[19] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Trans. Comput.*, vol. 55, no. 2, pp. 214–226, 2006.

[20] R. M. Karp, "Reducibility among combinatorial problems," in *Proc. Symposium on the Complexity of Computer Computations*, New York, USA, Mar. 1972, pp. 85–103.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: The MIT Press, 2001.

King-Yip Cheng obtained his Bachelor of Engineering and Master of Philosophy from the Department of Electrical and Electronic Engineering, the University of Hong Kong in 2004 and 2007, respectively. He was a research associate in the same department in 2008. His current research interest mainly focuses on time synchronization and localization in wireless sensor networks.

King-Shan Lui obtained her BEng. (first class honors) and MPhil. degrees in computer science from the Hong Kong University of Science and Technology. She then received her PhD degree from the University of Illinois at Urbana-Champaign, USA, in 2002. She joined the Department of Electrical and Electronic Engineering, the University of Hong Kong, as an assistant professor in August 2002. Her research interests include QoS issues, protocol and algorithm design in the Internet, ad hoc networks, and sensor networks.

Yik-Chung Wu obtained the B.Eng. (EEE) degree in 1998, M.Phil. degree in 2001 from The University of Hong Kong (HKU), and Ph.D. degree in 2005 from Texas A&M University, USA. From Aug. 2005 to Aug. 2006, he was with the Thomson Corporate Research, Princeton, NJ, as a Member of Technical Staff. Since Sep. 2006, he has been with the University of Hong Kong as an Assistant Professor. Yik-Chung's research interests are in general area of signal processing and communication systems, and in particular receiver algorithm design, synchronization techniques, channel estimation and equalization. He is currently serving as an associate editor for the IEEE COMMUNICATIONS LETTERS.

Vincent Tam completed his Ph.D. in Computer Science from the University of Melbourne. Since 1998, he joined the School of Computing, the National University of Singapore, as an academic fellow, and won the Innovative Teaching Award 2000. Vincent is currently a Senior Teaching Consultant in the Department of Electrical and Electronic Engineering, the University of Hong Kong. His research interests include heuristic search techniques and sensor networks.