IET Communications

Research Article

IET Journals
The Institution of Engineering and Technology

Optimisation-based time slot assignment and synchronisation for TDMA MAC in industrial wireless sensor network

ISSN 1751-8628
Received on 27th October 2018
Revised 10th June 2019
Accepted on 31st July 2019
E-First on 8th October 2019
doi: 10.1049/iet-com.2018.6065
www.ietdl.org

Ching-Lung Chang[1], Chuan-Yu Chang[1,2], Shuo-Tsung Chen[3] ✉, Shu-Yi Tu[4], Kuan-Yi Ho[1]

[1]*Department of Computer Science and Information Engineering, National Yunlin University of Science and Technology, Yunlin 64002, Taiwan*
[2]*Intelligence Recognition Industry Service Research Center (IR-IS Research Center), National Yunlin University of Science and Technology, Yunlin 64002, Taiwan*
[3]*Department of Information Management, College of Management, Fu Jen Catholic University, New Taipei City, Taiwan*
[4]*Mathematics Department, University of Michigan, Flint, MI 48502, USA*
✉ *E-mail: shough34@yahoo.com.tw*

Abstract: Wireless sensor network in the industrial environment [industrial wireless sensor network (IWSN)] has data delivery time constraint. Due to the dynamic routing and transmission collision, the data delivery time is unpredictable. The authors utilised time division multiple access (TDMA) MAC to avoid data collision and to provide bounded transmission delay. Moreover, a linear programming model is proposed to construct the TDMA schedules, which is focused on spatial reuse and fixed routing in IWSN. The objective function of the model is to minimise the time slot usage to increase the overall network bandwidth. Finally, both simulated annealing algorithm and particle swarm optimisation are applied to approximate the optimal solution of time slot usage.

1 Introduction

Industrial wireless sensor network (IWSN) has two important issues: data delivery time constraint and sensor battery power limit. The collision of data packets or the depletion of the power supply of the sensor during the transmission may cause the sensing data to be transmitted back to the central control centre outside the limited time. If the control centre cannot immediately respond appropriately to the above events, serious consequences may occur. In recent years, some scheduling and analysing methods have been proposed [1–19].

Scheduling algorithms or analysis for IWSN can be roughly classified into two types: single criticality and mixed criticality. In single criticality, authors in [1] proposed a real-time scheduling algorithm with schedulability and with linear topology for IWSN. In [2], authors proposed similar methods for binary-tree networks. Based on the method proposed by Zhang *et al*. and Soldati *et al*. [1, 2], the authors of [3] applied the impact of packet copying to enhance the channel utilisation. The method proposed by Chipara *et al*. [4] supports spatial reuse to improve the schedulability. In addition, the authors of [5–7] proposed a scheduling analysis, a fixed priority scheduling algorithm and two dynamic priority scheduling algorithms to meet the real-time requirement of industrial applications.

In the mixed criticality, the authors of [8, 9] presented ways to use some information to obtain a more precise schedulability analysis and more efficient preemptive fixed priority scheduling. However, uniprocessor systems and controller area networks in [10] do not support data flows. In [11, 12], the authors focus on homogeneous multiprocessor systems. When there is no interference between executing tasks, they do not need to consider how to avoid the interference. However, the interference must be avoided in IWSN. Network-on-chips use wormhole switching. For one data flow, the network-on-chip has to provide all nodes that the data flow uses simultaneously [13, 14], whereas the IWSN only provides two nodes for one hop. Wired networks [15, 16] and IEEE 802.11-based wireless networks [17] are based on the carrier sense multiple access (CSMA) protocol, which is unacceptable by reliable industrial systems. However, IWSN cannot adopt the hybrid protocol due to the unpredictability of the CSMA protocol. The authors of [18] consider a pure time division multiple access (TDMA) protocol. They proposed the priority medium access control (MAC) protocol, which is a distributed method and allows critical data flows to be transmitted as soon as possible. A scheduling algorithm is proposed to guarantee the real-time performance and reliability requirements of data flows with different levels of criticality in [19]. Their algorithm supports centralised optimisation and adaptive adjustment so that they can improve both the scheduling performance and flexibility.

However, the traditional TDMA [18, 19] allocates only one node per time slot. When the number of nodes increases, the number of time slots increases. Under the premise that each node guarantees a fixed bandwidth, the time for the overall network to collect data will also increase. Therefore, if more than two nodes transmit at the same time point and do not interfere with each other, they can be allocated on the same time slot, which can reduce the number of time slots and improve the overall network transmission efficiency. To achieve this goal in this work, we proposed an optimisation-based system architecture which is mainly divided into three stages: network topology mapping, TDMA scheduling, and time synchronisation. First of all, a neighbour list through neighbour exploration is established and then the neighbour list was transmitted to the central control centre to construct the entire network topology. Secondly, we utilised a TDMA protocol to satisfy Quality of Service (QoS) requirements and lower power consumption. Moreover, we apply simulated annealing (SA) algorithm and particle swarm optimisation to obtain optimal TDMA schedules. We also take the spatial reuse into account to enhance the network transmission efficiency. Finally, we use the proposed time synchronisation method to achieve the synchronisation of the whole network. The proposed system architecture is shown in Fig. 1 and the details are described in Sections 2 and 3.

The rest of this paper is organised as follows. Section 2 introduces the proposed protocol. In Section 3, we introduce the proposed linear programming models and the corresponding optimal solvers, SA algorithm and particle swarm optimisation. Moreover, we also proposed a time synchronisation scheme. Section 4 shows the simulation results. Finally, Section 5 concludes this work.

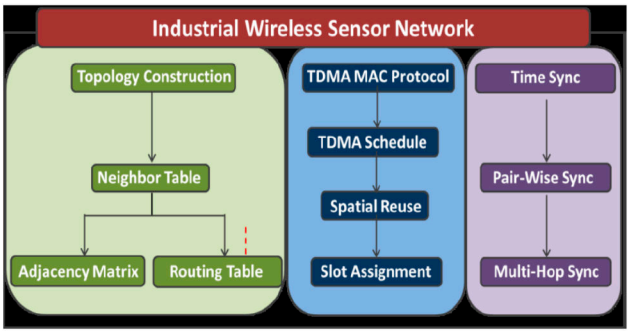

Fig. 1 *Proposed system architecture*

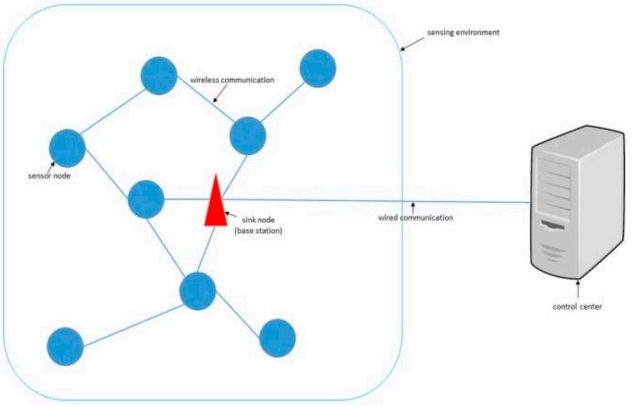

Fig. 2 *Entire system of a wireless sensing networks*

Contents of the frames		
Node ID	MAC Address	Sink Hop Counts
1	10-BF-48-D7-4E-75	1

Fig. 3 *Frame containing the node's ID, node's MAC, and Sink-Hop-Counts*

Routing Table(Stored in node)			
Node ID	Adjacencies Node	Previous Node	Sink Hop Counts
2	1、3	1	2

Fig. 4 *Routing table*

Neighbor Table(Stored in center controller)			
Node ID	Adjacencies ID	Previous Node	Sink Hop Counts
1	0、2、9	0	1
2	1、3	1	2
3	2、13、14	2	3
4	0、5、7	0	1
5	4、6	4	2
6	5、8	5	3
7	4、12	4	2
8	6	6	4

Fig. 5 *Neighbour table*

2 Proposed protocol

The entire system of wireless sensing networks includes control centre, sink nodes, wireless networks, and a large number of sensing nodes with communication capabilities as shown in Fig. 2. In the system, sensing nodes and wireless networks are the two cores. Each sensing node contains four major units: a sensing unit, a processing unit, a wireless transmission unit, and a power supply unit. The sensing unit is composed of a wide variety of micro-sensors which can measure temperature, humidity, brightness, acceleration, pressure, sound etc. The collected analogue signal is transmitted to the signal conversion element and converted into a digital signal. The processing unit is similar to a central processor in a personal computer and is responsible for executing code, coordinating, sending back data, and controlling different units. This unit also contains a small storage unit to store the collected environmental information. The wireless transmission unit is responsible for the communication between the sensing node and other nodes, and transmits the sensor data to the data collector (sink node). The power supply unit is mainly used to provide the energy required for the sensor node hardware to operate. Generally, a standard lithium battery or a solar battery that can draw energy from the environment can be selected.

In addition, these sensing nodes have the ability to self-organise the network, each representing a node in the network. The information collected by the sensor can be stored in the aggregation node (sink node) through the network and then transmitted back to the host or control centre.

In order to obtain information about the neighbours of each node (or transmitter) in sensor network, the SINK first broadcasts a frame namely *neighbour_discover* containing the node's ID, node's MAC, and SINK-Hop-Count, as shown in Fig. 3. Respectively, node's ID and node's MAC stand for the transmitter's node ID and MAC. SINK-Hop-Count means the distance between the transmitter and SINK.

When a node received the frame *neighbour_discover*, it returns a message ACK to the transmitter and records node's ID, node's MAC, and SINK-Hop-Count into its neighbour table containing the information for each of its neighbours. At the same time, if the value of SINK-Hop-Count in *neighbour_discover* is less than the minimum recorded, the node replaces self node's ID and MAC to the *neighbour_discover* by adding one and broadcasts the frame *neighbour_discover*. If a node does not receive any *neighbour_discover* after a while, it sends the neighbour table to the SINK. Finally, SINK collects all of neighbour table sent by each node and thus constructs topology, namely IWSN topology. Based on this topology, we denote adjacency nodes as interfered nodes. Moreover, the Dijkstra algorithm can be applied to derive the routing path from each node to SINK.

When each node does not receive any ACK message, it transmits routing table, as shown in Fig. 4, to the sink and then the sink forwards the information in routing table to control centre. As shown in Fig. 5, the control centre constructs the neighbour table according to the information. Moreover, we can build an adjacency matrix to show the relationship between any two nodes, as shown in Fig. 6.

After the neighbour table and the adjacency matrix are established, the control centre will then use the newly established neighbour table and Dijkstra's algorithm to calculate the path between each node and SINK shown in Figs. 7 and 8.

3 Linear programming model and simulated annealing

In this section, we introduce the proposed linear programming model and then solve the optimal problem derived from this model by SA.

3.1 TDMA scheduling and linear programming model

Traditionally, TDMA allocates only one node per time slot. When the number of nodes increases, the number of time slots will also increase so that the overall network time for collecting data will increase if each node has a fixed bandwidth. In other words, the frame length will increase with the number of slots and delay the transmission in the network.

Therefore, if two or more nodes that do not interfere with each other at the same time point can be allocated to the same time slot, the number of time slots can be reduced to improve the overall network transmission efficiency. These make the time slot assignment to become an optimal problem that the TDMA schedule has the smallest number of time slots.

	Sink	N1	N2	N3	N4	N5	N6
Sink	-	1	0	0	1	0	0
N1	1	-	1	0	0	0	0
N2	0	1	-	1	0	0	0
N3	0	0	1	-	0	0	0
N4	1	0	0	0	-	1	0
N5	0	0	0	0	1	-	1
N6	0	0	0	0	0	1	-

Fig. 6 *Adjacency matrix*

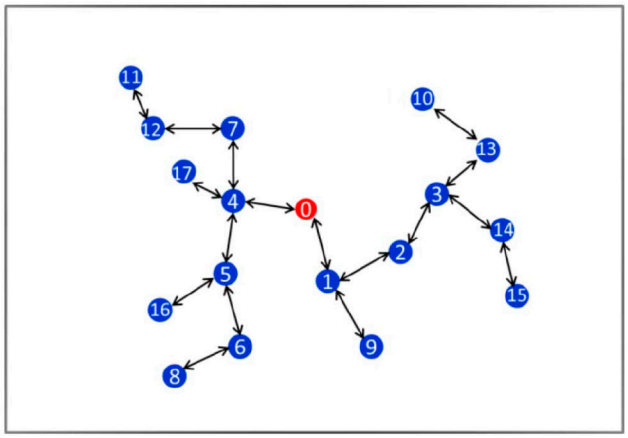

Fig. 7 *Path between each node and sink*

Routing Table(Stored in center controller)	
Node ID	ROUTING PATH
1	0 - 1 - 2
2	0 - 1 - 2
3	0 - 1 - 2 - 3
4	0 - 4
5	0 - 4 - 5
6	0 - 4 - 5 -6
7	0 - 4 - 7
8	0 - 4 -5 - 6 - 8

Fig. 8 *Path list between each node and sink*

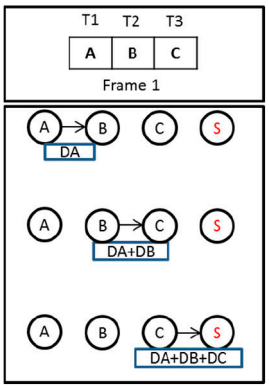

Fig. 9 *Time slot assignment that cooperate routing*

In order to solve this problem, we utilise two mechanisms which are 'Linear Programing Model' and 'SA' to obtain the optimal solution that we wanted. We first describe the proposed linear programming model as follows.

Given parameters:

N: a set of nodes including sensor nodes and sink node in the entire network;
$\bar{N}_n$: a set of interference node that we get from adjacency matrix; any member in the set cannot transmit data at the same time otherwise the receive node of node n may not receive data correctly due to signal interference;
M: a big number which is set to $|N|+1$;
T: T serves as an upper bound for the number of time slots in the static allocation period.

Decision variables:

y_{nt}: Binary variable. Its value is 1 if the sensor node n takes time slot to transmit data; otherwise its value is 0.
a_t: Binary variable. Its value is 1 if the time slot t is assigned to at least one sensor node; otherwise its value is 0.

The proposed linear programming model phase I (LP model I):
Minimise $\sum_{t\in T} a_t$ Subject to

(i) $\sum_{t\in T} y_{nt} = 1, \forall n \in N, n \neq 1$ (sink node);
(ii) $y_{nt} \in \{0,1\}, \forall t \in T$;
(iii) $(1-a_t) + \sum_{i\in \bar{N}_n} y_{it} \leq (1-y_{nt})M, \forall n \in N, t \in T$;
(iv) $a_t \in \{0,1\}, \forall t \in T$.

$|N|$, where the constraints are explained as follows. Constraints (i) and (ii) denote that all of nodes must be assigned at only one time slot. Constraint (iii) is to avoid the spatial reuse by interference nodes. That is, if the node n takes time slot t to transmit data then y_{nt} will be 1, which means the equation equals zero, the index set of interference node y_{it} cannot transmit data in time slot t, and a_t will be 1 which means time slot t is assigned at least one slot. In this equation, M needs to be greater than sum of $\bar{N}_n$, therefore we set M to $+1$. The objective function in (i) is to minimise the value of a_t or minimise the length of frame in TDMA schedule.

Since the time slot allocation of above linear programming model does not consider the routing problem, it may cause the scheduling of TDMA not to transfer data effectively. We illustrate this problem with the following two situations.

Suppose there is a node A whose path for sensing data transmission to Sink is A->B->C and the time slot allocation is T1(A), T2(B), T3(C), as shown in Fig. 9. In this first situation, time slot and routing work are at the same order. Since we use data aggregation to transfer data, node A only needs to transmit data to Sink within a single frame. However, if the time slot allocation is changed to T1(C), T2(B), T3(A), nodes B and C will transmit data in the slot before A, which causes the data of node A to pass through three frames to be transmitted to Sink, as shown in Fig. 10. In other words, the transmission efficiency is reduced in the second situation.

In order to enhance the transmission efficiency, we take the route into time slot allocation by the following two methods.

3.2 Ordering by hop counts

In this method, we use hop counts to represent the distance between the node and Sink. Then, the composition of the initial sequence is sorted according to the hop counts value, and the nodes of different hop counts cannot be arranged in the same slot. The larger the hop counts the nodes can be prioritised. For example, Fig. 11 shows that nodes A and C have the largest hop count 4 in the network. We assign these two nodes to the head of order, and then look for the second largest nodes, nodes B, D, and E, and then assign them after nodes A and C. Finally, we obtain the order 「A、C、B、D、E、F、G、H」.

For this consideration, we add two more given parameters and one more term to constraint (iii) in the proposed linear

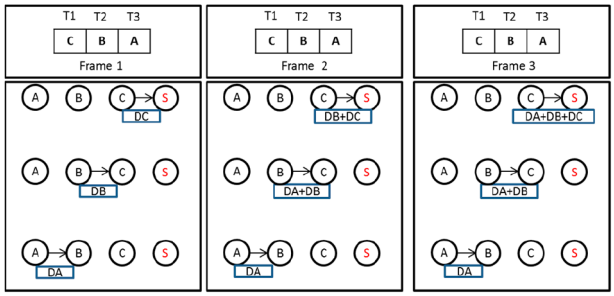

Fig. 10 *Time slot assignment without cooperate routing*

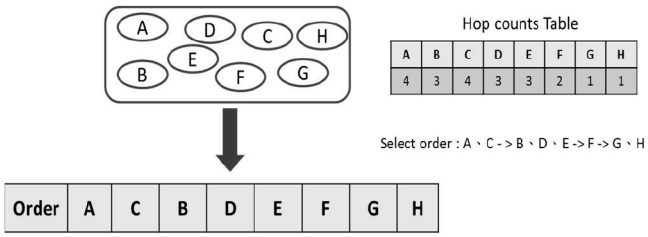

Fig. 11 *Ordering by hop counts*

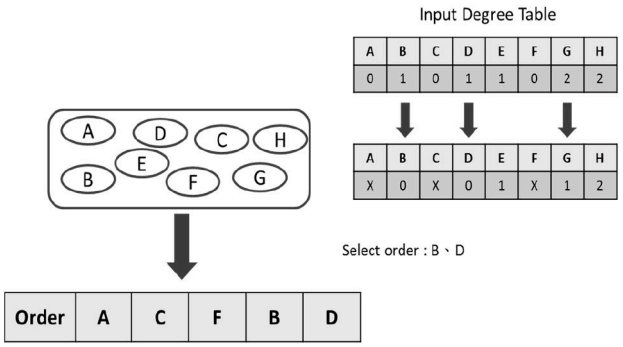

Fig. 12 *Ordering by input degree (a)*

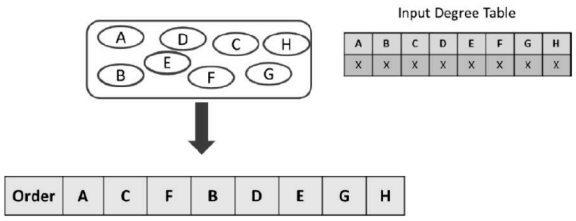

Fig. 13 *Ordering by input degree (b)*

programming model phase I to obtain linear programming model phase II.

Given parameters:

N: a set of nodes including sensor nodes and sink node in the entire network;
$\tilde{N}_n$: a set of interference node that we get from adjacency matrix; any member in the set cannot transmit data at the same time otherwise the receive node of node n may not receive data correctly due to signal interference;
H: a set of nodes in the network that are not currently scheduled to time slots and have the largest hop counts;
$\tilde{H}_n$: the set of nodes of different hop counts, and any node in the set cannot be arranged in the same time slot as node n;
M: a big number which is set to $|N|+1$;
T: T serves as an upper bound for the number of time slots in the static allocation period.

Decision variables:

y_{nt}: Binary variable. Its value is 1 if the sensor node n takes time slot to transmit data; otherwise its value is 0.
a_t: Binary variable. Its value is 1 if the time slot t is assigned to at least one sensor node; otherwise its value is 0.

The proposed linear programming model phase II (LP model II):
Minimise $\sum_{t\in T} a_t$ Subject to

(i) $\sum_{t\in T} y_{nt} = 1, \forall n \in N, n \neq 1$ (sink node);
(ii) $y_{nt} \in \{0,1\}, \forall t \in T$;
(iii) $(1-a_t) + \sum_{i\in \tilde{N}_n} y_{it} + \sum_{j\in \tilde{H}_n} y_{jt} \leq (1-y_{nt})M, \forall n \in N, t \in T$;
(iv) $a_t \in \{0,1\}, \forall t \in T$.

3.3 Ordering by node input degree

In this method, we use each node's input degree. For example, input degree table in Fig. 12 shows that nodes A, C, and F have zero input degree, which means there are no nodes behind them. So we first assign these three nodes to the head of order, and the input degree of these three node's forwarding node will subtract one.

Next, input degree table in Fig. 13 shows the new table after the first assignment. We find that nodes B and D have the zero input degree and assign them after nodes A, C, and F. Repeat the step mentioned before, finally we obtain the order 「A、C、F、B、D、E、G、H」.

For this consideration, we add two more given parameters and one more term to constraint (iii) in the proposed linear programming model phase I to obtain linear programming model phase III.

Given parameters:

N: a set of nodes including sensor nodes and sink node in the entire network;
$\tilde{N}_n$: a set of interference nodes that we get from adjacency matrix; any member in the set cannot transmit data at the same time otherwise the receive node of node n may not receive data correctly due to signal interference;
I: a set of nodes in the network that are not currently scheduled to time slots and have an input degree of 0;
$\tilde{I}_n$: a set of nodes with different input degree cannot be arranged in the same time slot as node n;
M: a big number which is set to $|N|+1$;
T: T serves as an upper bound for the number of time slots in the static allocation period.

Decision variables:

y_{nt}: Binary variable. Its value is 1 if the sensor node n takes time slot to transmit data; otherwise its value is 0.
a_t: Binary variable. Its value is 1 if the time slot t is assigned to at least one sensor node; otherwise its value is 0.

The proposed linear programming model phase III (LP model III):
Minimise $\sum_{t\in T} a_t$ Subject to

(i) $\sum_{t\in T} y_{nt} = 1, \forall n \in N, n \neq 1$ (sink node);
(ii) $y_{nt} \in \{0,1\}, \forall t \in T$;
(iii) $(1-a_t) + \sum_{i\in \tilde{N}_n} y_{it} + \sum_{i\in \tilde{I}_n} y_{jt} \leq (1-y_{nt})M, \forall n \in N, t \in T$;
(iv) $a_t \in \{0,1\}, \forall t \in T$.

3.4 Simulated annealing

SA is a generic probabilistic metaheuristic for the global optimisation problem by using a good approximation algorithm to find out the global optimum of a given function in a large search space [20].

As shown in Fig. 14, there are several steps for SA to solve the LP models which we build above. We describe the given parameters setting and the details of each step are as follows.

Given parameters:

T_i: initial temperature. We set T_i to 500;

T_c: current temperature;

T_f: stopping temperature. When $T_c < T_f$, it means the SA is completed;

L: the maximum iteration number at current temperature. We set L to 50;

l: current iteration;

β: cooling rate, when cooling condition is met, the current temperature will multiply the cooling rate. Cooling rate = 0.999 is adopted for our system;

E: the energy function or cost function;

E = number of time slots;

P: probability function P is applied to accept the new solution or not:

$$p = \begin{cases} 1 & \text{if } \Delta E \le 0 \\ \exp\left(\dfrac{-\Delta E}{\tau_c}\right) & \text{if } \Delta E \ge 0 \end{cases},$$

Step 1. Initial configuration setting: The configuration in SA for our model is an order of sensor nodes which is generated randomly.

Step 2. Time slot assignment: Based on the order of sensor nodes in step 1, we select one node at a time and then check time slot from small to large for the constraints in the proposed LP model. If there is conflict between nodes, then some nodes are assigned to a new time slot.

Step 3. Iteration: To obtain the optimal solution, we change the nodes' position in the current order to get the new order and thus have a neighbour solution. Moreover, we generate the new cost function nearby_E and calculate the difference between nearby_E and current_E (E_nearby – E_current). If the difference is smaller or equal zero which means the nearby_E is better than the current_E, we will accept this solution to be a best solution. Otherwise, the nearby_E is worse than the current_E. In order to avoid being trapped in local minima, we will accept the solution with a certain probability. This step will repeat until the iteration number is satisfied.

Step 4. Cooling: If the iteration number in step 3 is satisfied, then the current temperature will multiply the cooling rate, and step 3 will repeat again until met the stop condition.

Step 5. The stopping condition: When the current temperature is down to the stopping temperature. At this point, we may meet the global optimal solution that we want so that we will stop the algorithm and output the current time slot assignment.

3.5 Particle swarm optimisation

Particle swarm optimisation (PSO) is a heuristic algorithm that optimises a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search space to iteratively improve the candidate solutions according to simple mathematical formula over the particle's position and velocity [21, 22]. Suppose $x_{j,h}(t)$ and $v_{j,h}(t)$ are the position and the velocity of hth dimension in jth particle at time t. The algorithm of PSO can be expressed as

$$v_{j,h}(t) = v_{j,h}(t-1) + c_1 r_1 (x^*_{j,h} - x_{j,h}(t-1)) + c_2 r_2 (x^{\#}_h - x_{j,h}(t-1))$$

$$x_{j,h}(t) = x_{j,h}(t-1) + v_{j,h}(t)$$

where x^*_j and $x^{\#}$ denote the best position solution of jth particle and all particles as of time $t-1$; r_1 and r_2 are both random numbers; c_1 and c_2 are individuality coefficient and sociality coefficient, usually

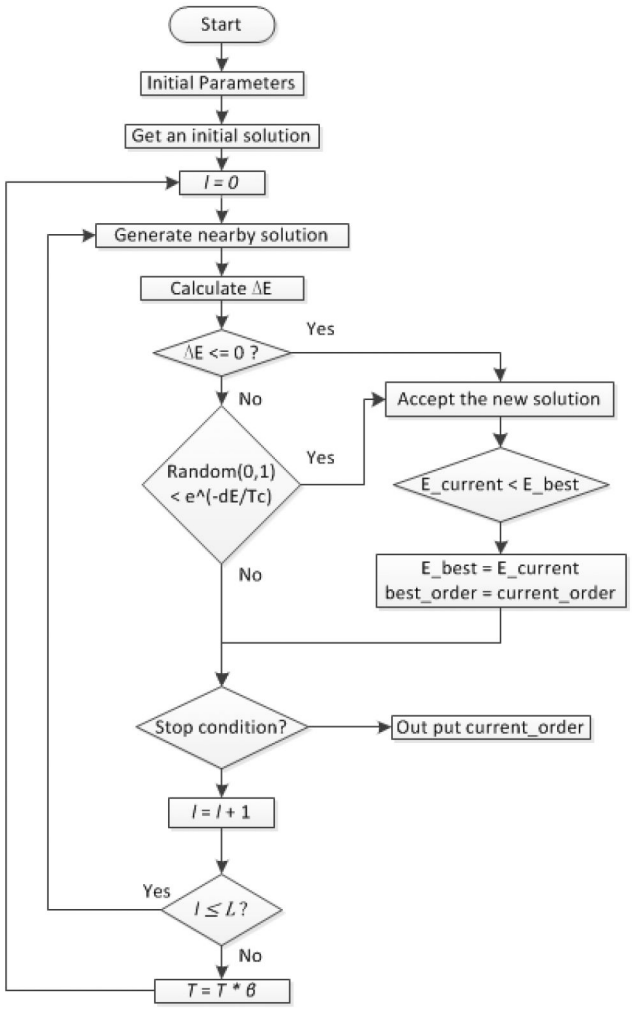

Fig. 14 *Flow chart of SA*

set to 2. There are several steps for PSO to solve the LP models we build above.

Step 1: Determine the population size 20 and set the initial value randomly for the position and velocity of each particle.
Step 2: Evaluate the fitness of the potential position solution for each particle *j*. If the fitness of the position solution is found to be better than the individual position solution in the previous memory, then the individual position solution is updated.
Step 3: Find the new best position solution in the entire particle swarm. If the fitness of this position solution is better than the position solution in the previous memory, then the position solution is updated.
Step 4: If the stopping condition has been met, the PSO is ended. Otherwise, go to step 5.
Step 5: Apply the algorithm of PSO to update the velocity and the position solution of each particle and go back to step 2 to continue.

4 Time synchronisation

Due to the fact that TDMA has a constraint which is sensor nodes, it must be time synchronised with sink node, otherwise the schedule will not work well or worse, therefore we proposed a time synchronisation algorithm that cooperates with TDMA.

Sensor nodes are mostly equipped with a hardware oscillator-assisted computer clock and the angular frequency of the hardware oscillator determines the rate at which the clock runs. We can implement an approximation $C(t)$ of real time t. It can be approximated with good accuracy by an oscillator with fixed frequency [Z]. Then, for some node i in the network, we can approximate its local clock as

$$C_i(t) = \omega_i t + \varphi_i \tag{1}$$

where ω_i is the clock drift and φ_i is the offset of node i's clock. Drift denotes the frequency of the clock and offset is the difference in value from real time t. Using (1), we can compare the local clocks of two nodes in a network, say node 1 and node 2 as

$$C_1(t) = \omega_{12} C_2(t) + \varphi_{12} \tag{2}$$

where ω_{12} is the relative drift between nodes 1 and 2, and the φ_{12} is the relative offset between nodes 1 and 2, if two nodes are perfectly synchronised, then their relative drift is 1 which means their oscillator has the same frequency rate and relative offset is 0 which means they have the same time at this moment.

The time synchronisation problem in sensor network of n sensor nodes is trying to equalise $C_i(t)$ for $i = 1..n$, and it should repeatedly correct the offsets to keep the node synchronised over a time period.

As the network is consisted by lots of sensor nodes, the basic idea is that the synchronisation starts at the sink, which synchronises with each downstream nodes, and these nodes that are synchronised by sink will synchronise their downstream nodes too, and repeat it until all nodes are synchronised. In the following, we utilised two schemes to synchronise the sensor network and the details are described as follows.

4.1 Pair-wise synchronisation

As shown in Fig. 15, the basic scheme to synchronise a pair of nodes, sink and sensor nodes is to synchronise their lock clock by exchanging the timestamp packet with the following procedure:

At first, sink transmits the first packet at T1, which this packet containing the timestamp T1 respects the sink local clock.

When the sensor node received the first packet at the T2 and records it, T2 is equal to T1 plus the D1, which is transmission time from sink to sensor node, then the sensor node transmits the second packet at T3, which contains the timestamps T1、T2、T3.

Next, sink received the second packet at T4 and records it. Then transmit the third packet at T5, which contains T1、T2、T3、T4、T5. After sensor node received the third packet at T6, firstly, the sensor node calculates the relative clock drift ω_{12}:

$$\omega_{12} = (T5 - T1)/(T6 - T2) \tag{3}$$

Secondly, we assume the transmit time is the same between sink to sensor node, then we can calculate the relative clock offset φ_{12} by subtracting $t2$ from $t4$:

$$\varphi_{12} = ((T3\omega_{12} - T1) - (T4 - T2\omega_{12}))/2 \tag{4}$$

Finally, sensor node has ω_{12} and φ_{12} now, so it can utilise (2) to synchronise with sink if the relative clock drift is fixed.

4.2 Multi-hop synchronisation

Multi-hop is an extension of the pair-wise synchronisation algorithm. Several important considerations for multi-hop synchronisation are listed as follows:

- *Synchronise route*: Sink according to the routing table, synchronises one downstream node at a time. Also these synchronised nodes will synchronise their downstream node too.
- *Global reference time (sink local clock)*: Due to the pair-wise synchronisation we proposed that node synchronised by sink can calculate approximate sink time, which means the sensor nodes that are not single hop between sink can actually synchronise with sink.
- *Resynchronisation rate*: In the actual environment, the frequency of oscillator will change with time or affected by the temperature、humidity etc., therefore the relative clock drift between each node will not be fixed for good. The synchronised errors on node must be increased when the work time increases; for solving this problem, the sensor network must be resynchronised again to regulate the relative clock drift.

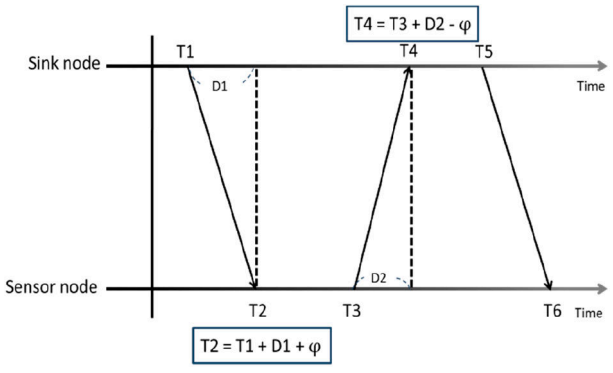

Fig. 15 *Timestamp packet exchange between two nodes*

Table 1 Simulation parameters for topology construction

	Environment 30 m × 30 m Resources	
	Sink node	Sensor node
numbers	1	30–60
location	(15, 15)	random deployment
sensing radius	9 m	9 m

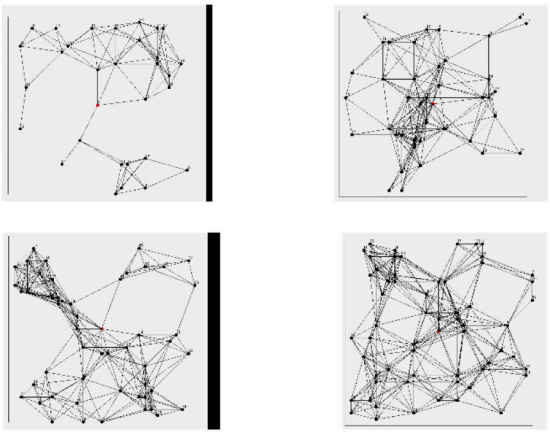

Fig. 16 *30–60 nodes topologies*

Table 2 Optimal number of time slots for different models obtained by SA and PSO

Number of nodes	Number of time slots					
	Model I		Model II		Model III	
	SA	PSO	SA	PSO	SA	PSO
30	7	7	14	14	9	9
40	18	18	25	25	18	18
50	12	12	26	26	12	12
60	15	15	28	28	15	15

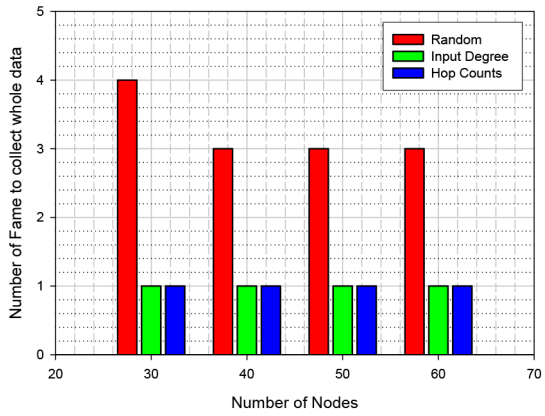

Fig. 17 *Number of frames to collect whole data for different models*

5 Experimental results

5.1 Topology construction

We utilised ZigBee sensor's parameters for simulation, and built four topologies for simulations. We list some parameters in Table 1 and Fig. 16.

5.2 Time slot assignment

As shown in Table 2, model I has the minimum number of time slots under different number of nodes. For model I (random), model II (hop counts), and model III (input degree), SA and PSO have the same number of time slots.

Figs. 17 and 18 show the number of frames to collect the whole data and the duration of whole data collection for different models. Since frame is the duration for collecting all sensing data once, it can be seen from the figure that model III, which uses input degree, not only achieves good time slot allocation and also improves the network transmission efficiency. Accordingly, we compare model III with the method [19], as shown in Table 3. One can see that the proposed method has better performance than the method [19].

5.3 Time synchronisation

Time synchronisation is an important issue in TDMA. The synchronisation errors directly affect the TDMA schedule efficiently. We used the time slot assignment result by input degree and the topology is 30 nodes. We list some parameters in Table 4 for simulated synchronisation errors. Figs. 19 and 20 perform the average synchronisation errors in the different resynchronisation rates in 30 and 300 s. The results were averaged over every possible pair. We can see the average synchronisation errors in 30 s results are only 0–5 μs, which means we utilised global reference time that makes every sensor node directly synchronised with sink works very well.

5.4 Power consumption

Finally, the last simulation is power consumption, which is very important in the wireless sensor network. For the simulation of power consumption, we used the time slot assignment result by input degree; list time parameters are given in Table 5.

As shown in Fig. 21, we can see that the network lifetime for 30–60 nodes can perform 6 years or more. As ZigBee sensor is a low data and low consumption device and if the ZigBee sensor

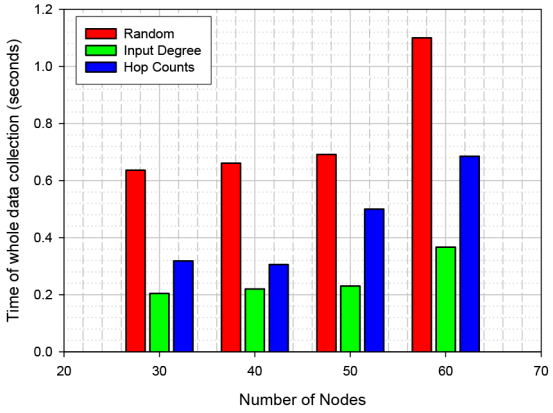

Fig. 18 *Duration of whole data collection for different models*

Table 3 Comparison for duration of whole data collection between method [19] and the proposed method

	Duration of whole data collection for different models, s			
	30 nodes	40 nodes	50 nodes	60 nodes
method [19]	0.39	0.56	0.76	0.84
the proposed	0.2	0.22	0.24	0.36

Table 4 Simulation parameters for time synchronisation

	Resources	
	Sink node	Sensor node
initial time, s	5	1–10
initial clock drift	1	1 ± 40 ppm
initial clock offset	0	sink time–sensor time
clock drift change rate	0	±0.5–1 ppm/10

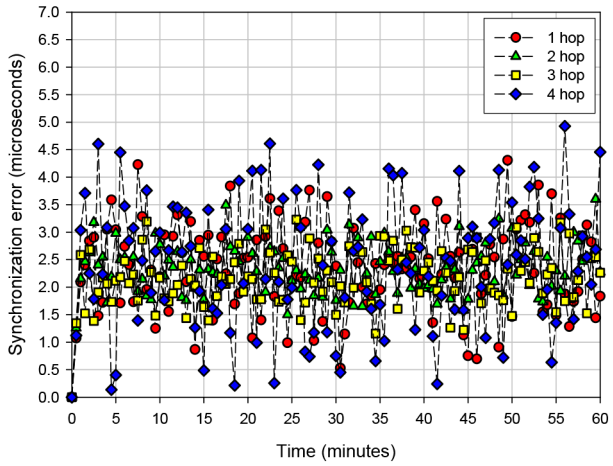

Fig. 19 *Resynchronisation rate in 30 s*

stays at the receive mode all the time, the battery (3 V, 1000 mAH) will run down in 40 h, but if the ZigBee sensor can change to the sleep mode when they are not scheduled to listen or transmit, the consumption will be down to 1 µA from 24 mA, these make the network lifetime prolong so much.

6 Conclusion

This work proposed three linear programming models to construct TDMA frames which combine spatial reuse and routing problem. The number of time slots in different models is minimised. A time synchronisation algorithm is also proposed to make the sensor work for all synchronised TDMA MAC. Experimental results reveal that the model with input degree demonstrates advanced

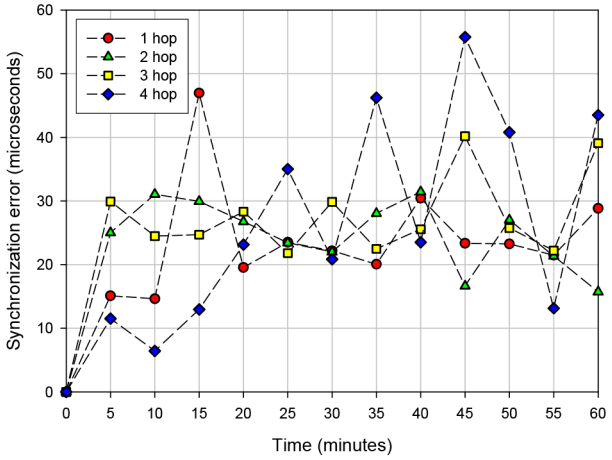

Fig. 20 *Resynchronisation rate in 300 s*

Table 5 Simulation parameters for power consumption

	Resources	
	Sink node	Sensor node
power supply	power cable	battery powered (3 V, 1000 mAH)
data packet	N/A	45 bytes
TDMA schedule		
frame intervals		60 s
frame length		using simulated data
power consumption		
transmit		29 mA
receive		24 mA
sleep		1 µA

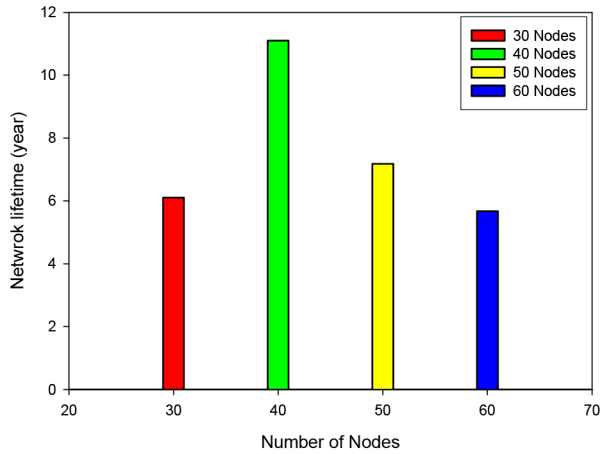

Fig. 21 *Network lifetime for different nodes*

outcomes in time slots assignment and network lifetime and the time synchronisation algorithm demonstrates advanced synchronisation errors. These results improve traditional TDMA in IWSN.

7 Acknowledgments

This work was financially supported by the 'Intelligence Recognition Industry Service Research Center (IR-IS Research Center)' from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

8 References

[1] Zhang, H., Soldati, P., Johansson, M.: 'Optimal link scheduling and channel assignment for converge cast in linear wireless HART networks'. Proc. of the WiOPT 2009 7th Int. Symp. on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, Seoul, Korea, 23–27 June 2009, pp. 1–8

[2] Soldati, P., Zhang, H., Johansson, M.: 'Deadline-constrained transmission scheduling and data evacuation in WirelessHART networks'. Proc. of the 2009 European Control Conf., Budapest, Hungary, 23–26 August 2009
[3] Zhang, H., Osterlind, F., Soldati, P., *et al.*: 'Time-optimal convergecast with separated packet copying: scheduling policies and performance', *IEEE Trans. Veh. Technol.*, 2015, **64**, pp. 793–803
[4] Chipara, O., Lu, C., Roman, G.C.: 'Real-time query scheduling for wireless sensor networks', *IEEE Trans. Comput.*, 2013, **62**, pp. 1850–1865
[5] Saifullah, A., Xu, Y., Lu, C., *et al.*: 'Real-time scheduling for WirelessHART networks'. Proc. of the 2010 IEEE 31st Real-Time Systems Symp. (RTSS), San Diego, CA, USA, 30 November–3 December 2010, pp. 150–159
[6] Saifullah, A., Xu, Y., Lu, C., *et al.*: 'End-to-end delay analysis for fixed priority scheduling in WirelessHART networks'. Proc. of the 2011 17th IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS), Chicago, IL, USA, 11–14 April 2011, pp. 13–22
[7] Saifullah, A., Xu, Y., Lu, C., *et al.*: 'Priority assignment for real-time flows in WirelessHART networks'. Proc. of the 2011 23rd Euromicro Conf. on Real-Time Systems (ECRTS), Porto, Portugal, 5–8 July 2011, pp. 35–44
[8] Vestal, S.: 'Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance'. Proc. of the RTSS 2007 28th IEEE Int. Real-Time Systems Symp., San Antonio, TX, USA, 3–6 December 2007, pp. 239–243
[9] Baruah, S., Vestal, S.: 'Schedulability analysis of sporadic tasks with multiple criticality specifications'. Proc. of the ECRTS '08 Euromicro Conf. on Real-Time Systems, Prague, Czech Republic, 2–4 July 2008, pp. 147–155
[10] Burns, A., Davis, R.I.: 'Mixed criticality on controller area network'. Proc. of the 2013 25th Euromicro Conf. on Real-Time Systems (ECRTS), Paris, France, 9–12 July 2013, pp. 125–134
[11] Burns, A., Fleming, T., Baruah, S.: 'Cyclic executives, multi-core platforms and mixed criticality applications'. Proc. of the 2015 27th Euromicro Conf. on Real-Time Systems (ECRTS), Lund, Sweden, 8–10 July 2015, pp. 3–12
[12] Lee, J., Phan, K.M., Gu, X., *et al.*: 'Mc-fluid: fluid model-based mixed-criticality scheduling on multiprocessors'. Proc. of the 2014 IEEE Real-Time Systems Symp., Rome, Italy, 2–5 December 2014, pp. 41–52
[13] Burns, A., Harbin, J., Indrusiak, L.S.: 'A wormhole noc protocol for mixed criticality systems'. Proc. of the IEEE Real-Time Systems Symp. (RTSS), Rome, Italy, 2–5 December 2014, pp. 184–195
[14] Tobuschat, S., Axer, P., Ernst, R., *et al.*: 'IDAMC: a Noc for mixed criticality systems'. Proc. of the 2013 IEEE 19th Int. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA), Taipei, Taiwan, 19–21 August 2013, pp. 149–156
[15] Cros, O., Fauberteau, F., George, L., *et al.*: 'Mixed-criticality over switched ethernet networks', *Ada User J. Proc. Workshop Mixed Crit. Ind. Syst.*, 2014, **35**, pp. 138–143
[16] Carvajal, G., Fischmeister, S.: 'An open platform for mixed-criticality real-time ethernet'. Proc. of the IEEE Design, Automation & Test in Europe Conf. & Exhibition, Grenoble, France, 18–22 March 2013, pp. 153–156
[17] Addisu, A., George, L., Sciandra, V., *et al.*: 'Mixed criticality scheduling applied to jpeg2000 video streaming over wireless multimedia sensor networks'. Proc. of the 2013 19th Int. Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA) Workshop on Mixed Criticality Systems (WMC), Taipei, Taiwan, 19–21 August 2013, pp. 55–60
[18] Shen, W., Zhang, T., Barac, F., *et al.*: 'PriorityMAC: a priority-enhanced MAC protocol for critical traffic in industrial wireless sensor and actuator networks', *IEEE Trans. Ind. Inf.*, 2014, **10**, pp. 824–835
[19] Jin, X., Xia, C., Xu, H., *et al.*: 'Mixed criticality scheduling for industrial wireless sensor networks', *Sensors*, 2016, **16**, p. 1376, doi:10.3390/s16091376
[20] Granville, V., Krivanek, M., Rasson, J.-P.: 'Simulated annealing: a proof of convergence', *IEEE Trans. Pattern Anal. Mach. Intell.*., 1994, **16**, (6), pp. 652–656
[21] Zhan, Z-H., Zhang, J., Li, Y., *et al.*: 'Adaptive particle swarm optimization (PDF)', *IEEE Trans. Syst. Man Cybern.*, 2009, **39**, (6), pp. 1362–1381
[22] Shen, M., Zhan, Z.-H., Chen, W.-N., *et al.*: 'Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks (PDF)', *IEEE Trans. Ind. Electron.*., 2014, **61**, (12), pp. 7141–7151