Appl. Math. Inf. Sci. 6 No. 2S pp. 713S-719S (2012)

Applied Mathematics & Information Sciences
An International Journal

@ 2012 NSP
Natural Sciences Publishing Cor.

A Reduce Identical Composite Event Transmission Algorithm for Wireless Sensor Networks

Jiun-Huei Ho[1], Hong-Chi Shih[2], Bin-Yih Liao[2], and Jeng-Shyang Pan[2,3]
[1]Department of Computer Science and Information Engineering, Cheng Shiu University, Kaohsiung, Taiwan
[2] Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan
[3] Innovative Information Industry Research Center (IIIRC), Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China
Corresponding author: Email: hqshi@bit.kuas.edu.tw

Received June 22, 2010; Revised March 21, 2011; Accepted 11 June 2011
Published online: 1 January 2012

Abstract: In this paper, a Reduce Identical Composite Event Transmission (RICET) algorithm is proposed to solve the problem of detecting composite events in wireless sensor networks. The RICET algorithm extends the traditional data aggregation algorithm to detect composite events, and this algorithm can eliminate redundant transmission and save power consumption, thereby extending the lifetime of the entire wireless sensor network. According to the experimental results, the proposed algorithm not only reduces power consumption by approximately 64.78% and 62.67%, but it also enhances the sensor node's lifetime by up to 8.97 times compared with some traditional algorithms.

Keywords: Wireless Sensor Network, RIET, Directed Diffusion, Grade Diffusion

1 Introduction

Recent advances in micro-processing, wireless and battery technology, and new smart sensors have enhanced data processing, wireless communication [1, 3, 4], and detection capabilities. The wireless sensor network is usually constituted by hundred or thousand nodes because sensor nodes are very cheap devices. In the wireless sensor network, multiple sensor nodes might detect the same event when they are randomly deployed on a large scale. The sensor nodes will consume a large amount of power if they send the same event to the sink node independently, as in the traditional algorithm [2, 7]. In general, the detected device is a small and cheap device. Hence, the wireless sensor node usually has several detected devices, and it can detect large amounts of different information in an area when an event takes place. Moreover, an event usually includes several different pieces of data. For example, we might detect the light, temperature, and CO2 data when a fire accident takes place. In this paper, a composite event can include several atomic events, and a sensor node can detect several atomic events when a composite event takes place because it has several different detected devices.

Several routing [8, 9, 10], clustering [4, 6], and aggregating [7] algorithms for wireless sensor networks have been proposed in recent years. However, they assume that sensor nodes all have the same detection abilities and devices and that they can detect, analyze, and transmit their detected event to the sink node by their algorithm. Their algorithms cannot aggregate detected data or cluster their sensor nodes into those that have several detection abilities or even into those with similar or different ones. The sensor nodes send their detected composite data to the sink node independently and thus consume a large amount of power in this case.

Hence, this paper proposes a Reduce Identical Composite Event Transmission (RICET) algorithm to improve the RIET [12] algorithm. The RICET algorithm extends the traditional data aggregation algorithm to detect composite events, and it can eliminate redundant transmission, save power consumption, and therefore extend the lifetime of the entire wireless sensor network.

2 Related Works

We used a Reduce Identical Composite Event Transmission (RICET) algorithm for a wireless sensor network, which improves upon the traditional RIET algorithm to detect composite events. In this section, we introduce some of the traditional algorithms and the RIET algorithm.

2.1 Directed Diffusion Algorithm (DD)

C. Intanagonwiwat et al. [5] introduced the directed diffusion (DD) protocol in 2003. DD aims to reduce the data relay to better manage power consumption and is, basically, a query-driven transmission protocol. The collected data are transmitted only if they fit the query from the sink node, thereby reducing the power consumption from data transmission. First, the sink node provides interested queries in the form of attribute-value pairs to the other sensor nodes by broadcasting the interested query packets to the entire network. Subsequently, the sensor nodes only send the collected data back to the sink node if they fit the interested queries.

In DD, all of the sensor nodes are bound to a route when broadcasting the interested queries, even if the route is such that it will never be used. In addition, several circle routes, which are built simultaneously when broadcasting the queries, result in wasted power consumption and storage. In the real world, the number of the sensor nodes in a system is in the hundreds or even thousands. Such a waste of power consumption and storage becomes worse and the circle route problem becomes more serious with larger-sized systems.

2.2 Grade Diffusion (GD)

In 2011, H. C. Shih et al. [11] proposed a Grade Diffusion algorithm that improves upon the LD-ACO algorithm [3] to enhance node lifetime, raise transmission efficiency, and solve the problem of node routing and power consumption. The GD algorithm broadcast grade completely and quickly creates packages from the sink node to every node in the wireless sensor network by the LD-ACO algorithm. Then the GD algorithm proposes a routing algorithm to reduce the nodes' average load, enhance the nodes' lifetimes and reduce the nodes' power consumption.

2.3 Reduce Identical Event Transmission Algorithm (RIET)

The DD and GD algorithms consider only a sensor node that sends an event to the sink node when the event takes place. Hence, sensor nodes consume large amounts of power to send the same event if detected. H. C. Shih et al. [12] proposed a Reduce Identical Event Transmission Algorithm (RIET), which can reduce the probability of sending a same event and save the sensor nodes' power. Moreover, the RIET algorithm is based on the GD algorithm.

The RIET algorithm that is used in the wireless sensor network can reduce the probability of sending a same event, save the nodes' power, and enhance the sensor nodes' lifetime by sensor node communication. The sensor node not only has the ability to sense and transfer events, but also can commute with its neighbor nodes when it senses an event. The RIET algorithm uses the finite state machine (FSM) that has a "Sensing State", "Delay State", "Query State", and "Receive Query State" to avoid simultaneous sensor node commutation with neighbor nodes in the algorithm. Hence, a sensor node can query or respond to its neighbor nodes by our algorithm and by the FSM. The FSM's transformation is shown in figure 1.

In figure 1, the sensor node has 4 states, which are "Sensing State", "Delay State", "Query Start", and "Receive Query State". In general, the sensor node is under the "Sensing State" process while waiting for an event to take place. When sensor nodes detect an event, their state changes to the "Delay State" and a delay time Wd is evaluated. Wd is illustrated in section 2.3.1. After the delay time, the sensor node's state changes to the "Query State". If a sensor node's state is the "Query State", it will commute with its neighbor nodes. Hence, the delay time Wd can prevent sensor nodes from changing their state to the "Query State" at the same time and thus from independently sending an event to the sink node.

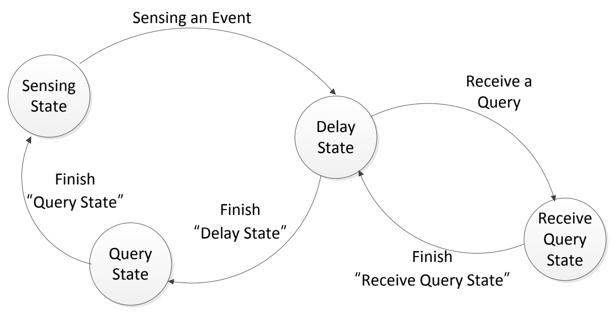

Figure 1: RIET algorithm's finite state machine

After the "Delay State", a sensor node's state changes to the "Query State", and the node sends a query message to its neighbor nodes, which asks whether they detect the same event. Then the sensor node state changes to the "Receive Query

State” when it is under the “Delay State” and receives a query message. If the sensor node’s delay time is shorter than those of the other nodes, the sensor node’s state changes to the “Query State” faster than those of the others. Afterwards, the neighbor nodes receive the query message and change their state to the “Receive Query State” from the “Delay State” if they detect the same event. Illustrations of the “Query State” and of the “Receive Query State” are in sections 2.3.2 and 2.3.3, respectively.

The “Delay State”, “Query State”, and “Receive Query State” are introduced in the following sections.

2.3.1 Delay State

When a sensor node senses an event, the sensor node will record the event status into E_s and evaluate a delay time Wd. The Es has three states: “Un-processing (-1)”, “Processed (+1)”, and “In-processing (0)”. Wd is evaluated by equation (1), and it is the delay time mean when the sensor node can change its state from the “Delay State” to the “Query State”.

$$W_d = \left(\alpha + N_l + \left(1 - \frac{E_{now}}{E_{original}}\right) + (1-\alpha) * P_r\right) * T_d \qquad (1)$$

$$N_l = \begin{cases} \left(\frac{l_v - 2}{l_v}\right), & \text{lv} > 2 \\ 0, & \text{otherwise} \end{cases}$$

In equation 1, α is set by the user to a value between 0 and 1, T_d denotes the shortest delay time, P_r is a random value between 0 to 1, l_v is the sensor node’s grade value, E_{now} is the sensor node’s residual power, and $E_{original}$ is the sensor node’s initial power. In the RIET algorithm, the sensor node needs to wait a delay time of at least α*T_d. Additionally, if a sensor node’s E_{now} is more than that of the others, the sensor node’s state can quickly change to the “Query State”. Moreover, the sensor node whose grade value is smaller than that of the others will have a shorter delay time W_d than the others because the event can be detected by different grade sensor nodes.

2.3.2 Query State

After the delay time, a sensor node first changes its status to the “Query State” from the “Delay State” and checks its E_s. If the E_s is “Processed (+1)”, the sensor node will do nothing and change its state to the “Sensing State” because the event has already been processed by other nodes. Otherwise, the sensor node will query its neighbor nodes and analyze data by the RIET algorithm because the event has not been processed if the sensor node’s E_s is “Un-processing (-1)” or “In-processing (0)”. The sensor node that sent query messages sets a minimum waiting time T_w to wait for its neighbor nodes’ responses, and the sensor node will check the received information, which contains the E_s and W_d of its neighbor nodes.

The query node checks its and its response node’s E_s. If the response node’s E_s is “Processed (+1)”, the query node will do nothing and change its state to the “Sensing State” because its neighbor nodes have already sent an event to the sink node. However, if the response node’s E_s is “Un-processing (-1)” or “In-processing (0)”, the query node will check its and its response node’s W_d. The query node sends the event to the sink node, modifies its E_s to “Processed (+1)”, and changes its state to the “Sensing State” when its W_d is smaller than that of the response node. However, the query node just changes its state to the “Sensing State” when its W_d is greater than or equal to that of the response node’s because its neighbor nodes will then send the event to the sink node.

2.3.3 Receive Query State

When a sensor node is under the “Delay State” and receives a query message, it changes its status to the “Receive Query State”. After that, the sensor node checks whether it has detected the same event as the query node. The sensor node will do nothing and change its state to the “Delay State” if it does not detect the same event. Otherwise, if the sensor node detects the same event, it responds with its E_s and W_d to the query node.

When the sensor node receives the query node’s W_d and E_s, it first checks them. If the sensor node’s E_s is “Processed (+1)”, it will do nothing and change its state to “Delay State”. Otherwise, when its E_s is “Un-processing (-1)” or “In-processing (0)”, the sensor node will check its W_d because the event has not been sent. If the sensor node’s W_d is smaller than or equal to that of the query node, the sensor node changes its E_s to “Processing (0)” and its state to the “Delay State” because the sensor node might later send the event to the sink node. The query node will change its E_s to “Processed (+1)” and finish its “Query State” if its W_d is greater than that of the receiving node.

The sensor node remains under the “Delay State” until the delay time W_d ends when it returns from “Receive Query State” processing. If the sensor node under the “Delay State” receives a query message again, the sensor node’s state will be changed to the “Receive Query State” again.

In sections 2.3.2 and 2.3.3, the sensor node needs to query its neighbor nodes and to receive their response to decide which sensor nodes can send event data to the sink node when it is under the “Query State”. However, if the sensor nodes have different detected devices and the event is a composite event, they cannot analyze the event using the RIET algorithm. Additionally, the algorithm cannot easily decide to which sensor node to transfer data when each sensor node has a different detected ability. This paper proposes a RICET algorithm to solve this problem.

3 Reduce Identical Composite Event Transmission Algorithm

This paper proposes a Reduce Identical Composite Event Transmission (RICET) algorithm to detect a composite event because a sensor node only has the ability to detect a single event in the RIET algorithm, and a composite event can include several atomic events. The RICET algorithm modifies the “Query State” and “Receive Query State” to analyze composite events.

3.1 Query State

In the RICET algorithm, sensor nodes have a multi-detection device and different devices, which are indicated by a state $\{E_i \mid i=1{\sim}N\}$. Moreover, each detected device has a state $\{D_i \mid i=1{\sim}N\}$ to record atomic events. The N means that there are N detected devices in total. The E_i is the data on the atomic event, and its i corresponds with D_i. Each sensor node has several different detected abilities, and a detected ability corresponds with its E_i and D_i to indicate different atomic events that have been detected. This means that the sensor node’s D_i and E_i are $\{D_1, D_2, D_3\}$ and $\{E_1, E_2, E_3\}$ if the sensor node has 3 detected devices. Moreover, the E_is are recorded as “Un-processing (-1)” or “Processed (+1)” to indicate the atomic event’s processing state.

A sensor node can find whether a composite event takes place and detect the atomic events that have been included in the composite event according to the sensor node’s detected device. The sensor node records the atomic events into its E_i, and all of the initial E_i states are “Un-processing (-1)”. After that, the sensor node evaluates a delay time W_d by equation (1) and changes its state from “Delay State” to “Query State”.

The sensor node first checks all its E_i states when it is under “Query State” processing. If all of its E_i states are “Processed (+1)”, the sensor node will do nothing because all of the atomic events have been processed by other sensor nodes. However, the sensor node still sends a query message to its neighbor nodes to notify them that the event has been processed. Then the neighbor nodes process the message as in section 3.2, and the query sensor node changes its state to the “Sensing State” from the “Query State”.

However, if its E_i is “Un-processing (-1)”, the sensor node queries its neighbor nodes when there is an atomic event that has not been processed. Firstly, the sensor node sets a minimum waiting time T_w to wait for its neighbor nodes’ responses, and the sensor node checks the E_i, D_i, and W_d from the responses of its neighbor nodes. If a neighbor node responds with an E_i and D_i, it has detected an atomic event that is the ith detected device. After that, the query node checks whether its neighbor node’s E_i is “Processed (+1)” or not. The query node sets its E_i to “Processed (+1)” when its neighbor node’s E_i is “Processed (+1)” because the neighbor node’s detected device has detected the same atomic event, which has been sent to the sink node. The query node checks all of its E_i if all of the E_i are the same as those of its neighbor nodes’ responses, and if all neighbor nodes’ E_i are “Processed (+1)”, the query node modifies its E_i to “Processed (+1)” and finishes “Query State” processing.

However, if E_i is “Un-processing (-1)” for all of the neighbor nodes detecting this atomic event by the ith detected device, the query node will check its and its response node’s W_d. The query node will send the atomic event to the sink node, modify this E_i to “Processed (+1)”, and change its state to “Sensing State” when its W_d is smaller than or equal to the response node. Before sending the atomic event to the sink node, the sent node sets a threshold E_{th} to combine and average the D_i data if its neighbor nodes’ D_is are under the threshold. If any of its neighbor nodes’ D_is are over the threshold, the D_i will be destroyed. After the sensor node sends atomic event data to the sink node, the sensor node modifies the E_i’s status to “Processed (+1)” and changes the node’s status to the “Sensing State”. Otherwise, when its W_d is larger than response node, the query node merely changes its state to the “Sensing State” because its neighbor nodes will send this atomic event to the sink node.

3.2 Receive Query State

When a sensor node is under the “Delay State” processing and receives a query message, the sensor node’s state will be changed to “Receive

Query State”. The sensor node checks whether it detects the same atomic events E_i as the query node had. The sensor node will do nothing and change its state to “Delay State” if it does not detect the same atomic events E_i. However, if the sensor node has detected the same atomic event, it responds with its E_i and W_d to the query node and receives the query node’s E_i and W_d.

When the sensor node receives the query node’s E_i, D_i and W_d, it checks its and its query node’s E_i. The sensor node just checks the same E_i as those of the query node. If all of the sensor node’s E_is are “Processed (+1)”, it does nothing and changes its state to the “Delay State”. Otherwise, the sensor node compares its W_d with that of the query node for when its E_i is “Un-processing (-1)” because the atomic event would not have been sent to the sink node. If sensor node’s W_d is greater than or equal to the query node’s W_d, the sensor node simply changes the E_i to “Processed (+1)” because the query node will send the atomic event. Otherwise, when its W_d is smaller than that of the query node, the sensor node will do nothing. After the sensor node checks all of the same E_i as that of the query node, the sensor node finishes “Receive Query State” processing and returns to “Delay State” processing.

The sensor node is still under the “Delay State” until the end of the delay time W_d when it returns from “Receive Query State” processing. If the sensor node under the “Delay State” processing receives a query message again, the sensor node’s state will be changed to the “Receive Query State” again.

Additionally, the sensor node will not be in “Query State” and “Receive Query State” processing at the same time because they must avoid changing E_i at the same time.

4 Simulation and Analysis

We simulate and analyze the RICET algorithm in this section. In the simulation, we built a 3-D space of 100*100*100, and the scale of the coordinate axis for each dimension was limited from 0 to 100. The radio limit of the nodes was set to 15 units. There were three non-overlay nodes in each 10*10*10 space, and their Euclidean distance from each other was at least 2. Thus, the space included 3000 nodes, and a center node of (50,50,50) was assigned to the sink node. Moreover, each sensor node had 1~3 detected devices, and there were 3 kinds of detected device in our simulation.

There were 300 cycles, and each cycle had a composite event. Thus, there were 90,000 composite events in our simulation, and each composite event had 3 atomic events that could be detected. Every sensor node’s initial power was set at 3,600 mw, and 1.6 mw was consumed at each commutation with other sensor nodes and at each event data transmission. Moreover, we set the E_{th} at 10%. The simulation decided that multiple sensor nodes detected a same atomic event if the atomic event values differed by more than E_{th}.

Firstly, we compared the active nodes of the Directed Diffusion (DD), Grade Diffusion (GD), and RIET algorithm in each cycle, and the result is shown in figure 2. The active node means that the sensor node has enough power to send an atomic event and has relay nodes in its routing table.

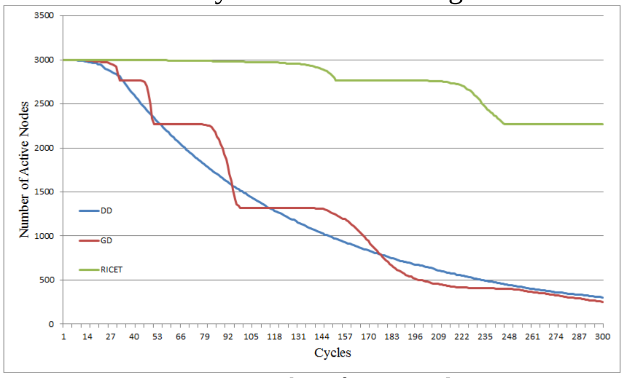

Figure 2: Number of active nodes

In figure 2, the DD and GD algorithms only had 303 and 253 active nodes after 300 cycles, but the RICET algorithm still had 2270 active sensor nodes that had enough power to detect and send an event. The RICET algorithm enhanced the sensor nodes’ lifetimes by approximately 7.5 times and 8.97 times those of the DD and GD algorithms, respectively, because the RICET algorithm can reduce the transferred numbers of sensor nodes when they detect the same composite event. In the DD and GD algorithms, the sensor node independently sends atomic events to the sink node and consumes a large amount of power if they detect the same composite event. This means that the sensor node sends 3 times the amount of atomic data to the sink node when it detects a composite event because a composite event includes 3 atomic events. Hence, the RICET algorithm can save a large amount of power and enhance the lifetimes of sensor nodes. Thus, the RICET algorithm has more active nodes than the DD and GD algorithms in figure 2.

Then, we compared the average power consummation of the DD, GD, and RICET algorithms, and the result is shown in figure 3. In figure 3, we see that the RICET algorithm has the lowest average power consumption because it can reduce the transferred numbers of sensor nodes that detect the same event. Hence, the average power consumptions of the DD and GD algorithms are 3497.83 mw and 3300.13 mw after 90000 composite events take place, but the RICET algorithm only consumes 1232.02 mw. Hence, the RICET algorithm can reduce the average power consumption by approximately 64.78% and 62.67% of those of the DD and GD algorithms, respectively.

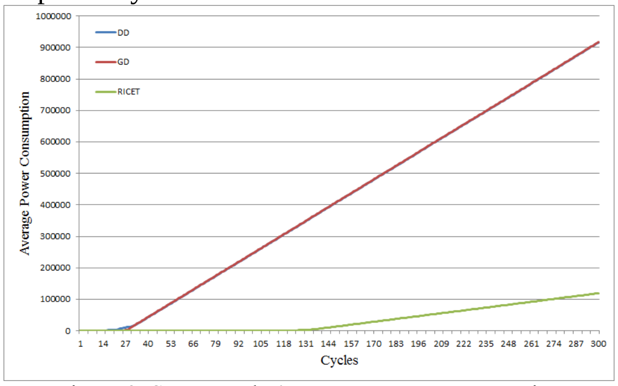

Figure 3: Sensor nodes' average power consumption

Finally, we compared the total data loss of the DD, GD, and RICET algorithms. In this simulation, we counted the number of composite events that could not be sent to the sink node or those transferred 20 times, in which the composite event would be dropped.

In figure 4, we see that there are many data losses in the DD and GD algorithms. The DD and GD algorithms send a same composite event to the sink node when a composite event takes place and many sensor nodes detect the event. Hence, some sensor nodes close to the sink node exhaust their power quickly, which we call the inner nodes. All of the inner nodes had been exhausted at 33 and 29 cycles, and there were not any events that could be sent to the sink node by the DD or GD algorithms. After 300 cycles, the DD and GD algorithms had data losses of 915766 and 916753, respectively. In our simulation, there were only 90000 composite events, but the data losses of the DD and GD algorithms were greater than that because they re-sent same composite events. The RICET algorithm had a data loss of only 119213 and extended the lifetime of the inner nodes until the 135th cycle. Hence, the RICET algorithm can reduce the data loss by approximately 86.98% and 87% of those of the DD and GD algorithms, respectively.

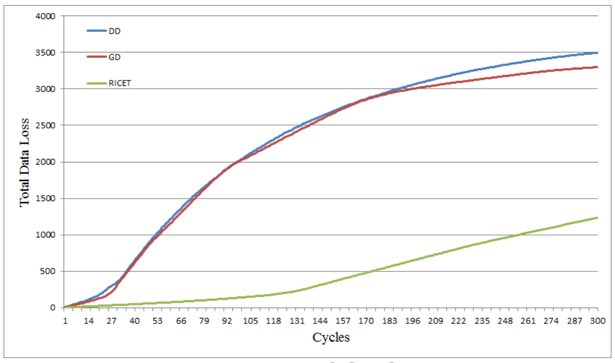

Figure 4: Total data loss

5 Conclusions

This paper proposed a Reduce Identical Composite Event Transmission Algorithm (RICET). The RICET algorithm extends the traditional data aggregation algorithm to detect a composite event, and this algorithm can eliminate redundant transmission, save power consumption, and therefore extend the lifetime of the entire wireless sensor network.

In our simulation, we found that the RICET algorithm can save sensor node power by up to 64.78% of that of traditional algorithms. Moreover, the RIET algorithm can enhance a sensor node's lifetime by up to 12.9 times and reduce data loss by approximately 87% after sending 900000 composite events. Hence, the RICET algorithm can enhance the nodes' lifetime when they detect a same composite event and thus reduce the data loss rate.

References

[1] S. Corson and J. Macker, Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. RFC 2501, (1999).

[2] Z. Hea, B. S. Lee, and X. S. Wang, Aggregation in Sensor Networks with a User-Provided Quality of Service Goal. Information Sciences, Vol.178, No. 9, (2008), 2128-2149.

[3] J. H. Ho, H. C. Shih, B. Y. Liao and S. C. Chu, A Ladder Diffusion Algorithm Using Ant Colony Optimization for Wireless Sensor Networks. Information Sciences, (2011).

[4] T. P. Hong and C. H. Wu, An Improved Weighted Clustering Algorithm for Determination of Application Nodes in Heterogeneous Sensor Networks. Journal of Information Hiding and Multimedia Signal Processing, Vol. 2, No. 2, (2011), 173-184.

[5] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, Directed Diffusion for Wireless Sensor Networking. IEEE/ACM Transactions on Networking, Vol. 11, (2003), 2-16.

[6] V. Kawadia and P. R. Kumar, Power Control and Clustering in Ad Hoc Networks. The Twenty-second

Annual Joint Conference of the IEEE Computer and Communications Societies, (2003), 459-469.

[7] W. H. Liao, Y. Kao and C. M. Fan, Data Aggregation in Wireless Sensor Networks Using Ant Colony Algorithm. Journal of Network and Computer Applications, Vol. 31, (2008), 387-401.

[8] J. Pan, Y. Hou, L. Cai, Y. Shi and X. Shen, Topology Control for Wireless Sensor Networks. The Ninth ACM International Conference on Mobile Computing and Networking, (2003), 286-299.

[9] C. E. Perkins and E. Royer, Ad Hoc On-Demand Distance Vector Routing. Proceedings of IEEE WMCSA, (1999), 90-100.

[10] E. M. Royer and C. K. Toh, A Review of Current Routing Protocols for Ad-Hoc Mobile Networks. IEEE Personal Communications, Vol. 6, (1999), 46-55.

[11] H. C. Shih, J. H. Ho, B. Y. Liao, and J. S. Pan, Grade Diffusion Algorithm. Proceedings of the Information Technology and Applica-tion in Outlying Islands (2011)

[12] H. C. Shih, S. C. Chu, John Roddick, J. H. Ho, B. Y. Liao, and J. S. Pan, A Reduce Identical Event Transmission Algorithm for Wireless Sensor Networks. Proceedings of the third International Conference on Intelligent Human Computer Interaction, (2011)

Jiun-Huei Ho received the B.S. degree in Electronic Engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, Republic of China in 1990, and the M.S. degree in the Institute of Computer and Information Engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, Republic of China in 1994, and a Ph.D. in the Department of Computer Science and Engineering at National Sun Yat-Sen University, in 2005. He joined the faculty of Department of Electrical Engineering, Cheng Shiu Institute of Technology, Taiwan, as an Lecturer in 2000. Since 2005, he has been an assistant professor of the Department of Computer Science and Information Engineering, Cheng Shiu University, Taiwan. His research interests include Wireless Sensor Network, mobile computing systems, Internet technology, and distributed system.

Hong-Chi Shih received the M.S. degree from National Kaohsiung University of Applied Science. He is currently a Ph.D. student with the Department of Electrical Engineering at National Kaohsiung University of Applied Science, Taiwan.

Bin-Yih Liao received the Ph.D. degree in Department of Electronic Engineering from National Cheng Kung University. He is currently a Professor with the Department of Electrical Engineering at National Kaohsiung University of Applied Science, Taiwan. His research interests are in the areas of image processing, information security and embedded system.

Jeng-Shyang Pan received the B.S. degree in Electronic Engineering from the National Taiwan University of Science and Technology in 1986, the M.S. degree in Communication Engineering from the National Chiao Tung University, Taiwan in 1988, and the Ph.D. degree in Electrical Engineering from the University of Edinburgh, UK in 1996. Currently, he is a Professor in the Department of Electronic Engineering, National Kaohsiung University of Applied Sciences, Taiwan. He joined the editorial board of International Journal of Innovative Computing, Information and Control, LNCS Transactions on Data Hiding and Multimedia Security, International Journal of Hybrid Intelligent System, Journal of Information Assurance and Security, International Journal of Computer Sciences and Engineering System, Journal of Computers, International Journal of Digital Crime and Forensics, and ICIC Express Letters. His current research interests include soft computing, information security and signal processing.