Journal of Applied Science and Engineering, Vol. 19, No. 4, pp. 515–524 (2016) *DOI: 10.6180/jase.2016.19.4.15*

Group Events Ordering by Double Confirmations in Wireless Sensor and Actuator Networks

Yi-Chao Wu[1]* and Chiu-Ching Tuan[2]

[1]*Department of Information Management, Chihlee University of Technology, New Taipei City, Taiwan 220, R.O.C.*
[2]*Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan 106, R.O.C.*

Abstract

In wireless sensor and actuator networks, events ordering is an important issue since the ordering of monitored events actuator received could determine the correct interpretation of what event is occurring in the monitored environment. In the real situation, events may belong to different groups. Each group may be co-related with other groups based on the time ordering, such as the group-related events ordering we defined. However, the existing events ordering algorithms addressed nothing for group-related events ordering. Hence, we proposed a group events ordering by double confirmations, GOBDC, to treat the group-related events ordering in correct group order. Simulation results demonstrated that the rate of the correct events ordering of GOBDC could be up to 100%. Once the number of events or the probability of event with a delay increased, GOBDC still guaranteed the rate of the correct events ordering to be 100%.

Key Words: Wireless Sensor and Actuator Networks, Events Ordering, Correct Interpretation, Group-related Events Ordering, Double Confirmations

1. Introduction

Wireless sensor and actuator networks (WSAN) is a new kind of heterogeneous wireless sensor networks (WSN). It is composed with a large number of sensors and few of powerful actuators. Sensors are low cost and low power devices with limited sensing, computing, and wireless communication. Actuators are equipped with the powerful processing capabilities, transmission radius, and battery energy. Sensors gather and send the information to actuators. Actuators take decisions and perform the appropriate actions based on the received data. Hence, WSAN can perform both sensing and acting the environmental events [1,2].

Based on different types of applications, WSAN was classified into semi-automated and automated architectures [1,3,4]. The merit of semi-automated architecture is not required to develop new algorithms or protocols for communication and coordination. Nevertheless, actions may delay to be taken by actuator waiting commands from the sink.

In the automated architecture, sensors gather and forward the sensed data to the appropriate actuators. The actuators take the required decisions and perform the proper directly actions based on the sensed data. Sensors thus could reduce energy consumption without forwarding data to the sink through multiple hops. The latency of taking actions could be reduced and the lifetime of network could be prolonged. Therefore, we used the automated architecture in this paper.

Since actuators could take actions for the sensed data directly, an attractive application of WSAN was to monitor the physical environments subjected to critical conditions [5–8]. Context may be composed by more inter-

*Corresponding author. E-mail: alanwu@mail.chihlee.edu.tw

related events. Once events are related to each other, events ordering becomes important since incorrect ordering of events may cause incorrect actions and serious losses in some applications, e.g., tracking moving objects or monitoring fire. Whenever the relation among events was regarded as temporal, the time ordering of events occurring was critical to process them timely and correctly [5–8]. Hence, how to keep the track of events occurring accurately is requisite in WSAN. For example, the correct direction in which the object is moving can be tracked and determined by gathering its relative order of object moving and passing by different locations over time.

In the applications for events ordering, the correct events ordering could lead to perform the correct actions by actuators. For example, a correct path of man moving and passing by sensors 1, 2, and 3 (denoted as S_1, S_2, and S_3) sequentially is shown in Figure 1(a). Given that delays make the later event from S_3 prior to former *one* from S_2 to arrive at the collected node. An incorrect path was thus decided as 1-3-2, as shown in Figure 1(b). This error path may mislead the rescue team to move with an opposite direction compared to the actual path, 1-2-3.

However, events may be divided into different kinds of *ones*, such as smoke and motion detection, in a fire rescue application. A *group* was defined as a *kind* of events. These kinds of events were defined as the group-related events and may have the group correlation among them. In a fire rescue application, the ordering of smoking and motion detection captured by different source sensors may be related to each other. For example, once a fire is happened, the ordering of smoking events showed the fire path and the ordering of motion detection events showed the moving path of the escaped people. The rescuers need to consider both of the ordering of smoking events and *that* of motion detection events to determine the optimal rescuing path. Therefore, the group-related events ordering is important for the events ordering applications.

For example, in the above fire rescue application, two groups are defined, such as *smoking group* and *motion detection group*, respectively. Ordering of the events belonged to different groups was defined as the group-related events ordering in this paper.

However, the existing events ordering solution, such as ordering by confirmation, OBC [5–8], addressed nothing for group-related events ordering. In fact, events may belong to different groups. Each group may be co-related to another *one*. Therefore, we proposed a group events ordering by double confirmations, GOBDC, to guarantee the rate of the correct events ordering up to 100% for group-related events ordering.

Experiments were conducted to measure and compare the performance of ordering by simulation. The remainder of the paper was in the following sections. Section 2 presented the related work. Section 3 stated our scheme. Section 4 showed the simulation results. Section 5 concluded the paper, finally.

2. Related Work

In some applications of WSAN, the ordering of

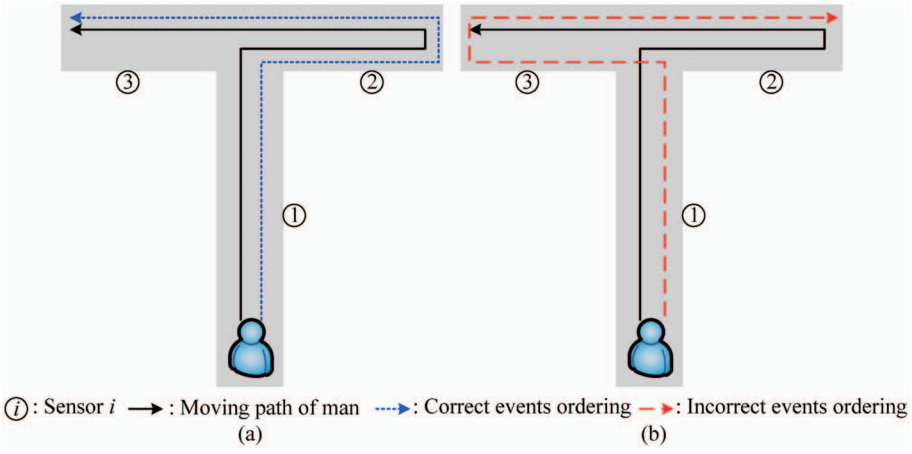

Figure 1. (a) Correct ordering; (b) Incorrect ordering.

events arrived at actuator needed to be the same as *that* of events captured from source sensors. Hence, to keep the ordering among events accurately becomes an important issue in WSAN. Unfortunately, the sequence of events may be fail due to the delays of transmitting events along the routing path to actuator from (i) message transmission, (ii) competition for accessing the communication medium, (iii) multiple hops along routing paths, (iv) MAC layer protocol, and (v) dynamic topology of networks [5–8]. To make the correct ordering, an actuator needs to ensure none of prior messages still in transit over the network while the received events are processing in order.

In [9,10], the authors proposed a causal ordering based on a "happened before" relation. The subscripts *i* and *j* were defined as the *i-th* and *j-th* events captured by source sensors, where $i < j$. Once event 1 (e_1) is happened before e_2, e_1 needs to be processed before e_2 even if any possible violation of event ordering at a receiver site.

The delaying techniques assumed an upper bound of delay D in which events could be routed back to the collected node, such as sink [11]. Here, D was defined as the maximum time taken by one sensor for sending an event to a collected node. The collected node kept a list of sensed events. After an elapsed time equals to D, the events ordered in the list are handed out to application in order. However, it is difficult to define an optimal D since the delay often varies over time. If D is longer, the collected node may waste redundant idle time on waiting events. A shorter D may result in missing some prior events that still are in transit.

To solve the above problems, a temporal message ordering in sensor networks, TMOS, was proposed [12]. However, TMOS needed to consume more energy because it constructed logical rings to forward the same messages in clockwise and anti-clockwise directions along the logical ring. It also used a longer time to confirm the messages along the m of nodes ring. In addition, it is not straightforward to construct a logical ring in sensor networks. Therefore, an ordering by confirmation, OBC, in WSAN was proposed [5–8].

OBC was a typical algorithm for ordering events in WSAN. It is based on the FIFO property of channels. Firstly, it constructed a tree cluster that each actuator acted as the cluster header and each sensor within this cluster transmitted the information of events to cluster header by PEQ [13]. When an actuator received an event i (e_i), it broadcasted a temporal acknowledgment (*TACK*) immediately to all leaf nodes. Each leaf node then replied *TACK* to the actuator along the original routing path. When the actuator received *TACKs*, the actuator defined that earlier events had been received completely, if any. These ordered events in the actuator's buffer will be removed in sequence. In OBC, none of nodes sent a duplicate event to the actuator, and none of logical rings was needed, such as TMOS.

However, OBC cannot guarantee that all co-related events could be treated in correct order once the number of events increased or the acknowledgment delayed. To address this issue, an event ordering by double confirmation, OBDC, was proposed [14]. In OBDC, the property of channels was FIFO. Sensors and actuators were clustered by PEQ [13]. While the actuator received e_i, it will broadcast a conformation i (m_i) to all leaf sensors. The m_i was the corresponding confirmation message for e_i. The t_{e_i} was defined as the timestamp of e_i captured by the source sensor. The t_{m_i} was defined as the happened timestamp of m_i as same as t_{e_i}. While the actuators receive all m_i from the leaf sensors, the events which their timestamps were less than or equal to t_{m_i} were treated in order.

We illustrated an example for OBDC. In Figure 2(a), S_{23} detects and sends e_0 to the actuator, and then S_{17} sends e_1 to the actuator. Assumed that e_1 arrives at actuator pervious to e_0 due to the delay of e_0, as shown in Figure 2(b). Next, the actuator broadcasts m_1 to leaf sensors (e.g., S_3, S_7, S_{11}, S_{14}, S_{15}, S_{21}, S_{22}, S_{23}, S_{24}, S_{25}, and S_{26}), as shown in Figure 2(c). Once e_0 arrives at actuator, actuator will save e_1 and e_0 sequentially and sort them decreasingly in buffer. Leaf sensors received m_1 must route m_1 back to the actuator along the cluster topology. While the actuator receives all m_1 from its one-hop neighboring sensors (i.e., S_1, S_5, S_8, S_{12}, S_{16}, and S_{17}), it will remove e_0 and e_1 orderly from buffer and hand them out to application, as shown in Figure 2(d). Although e_1 arrives in actuator previous to e_0, OBDC still could treat e_0 and e_1 in order to avoid generating an incorrect events ordering. To understand OBDC easily, the corresponding time sequence was shown in Figure 3.

However, OBC and OBDC addressed nothing for

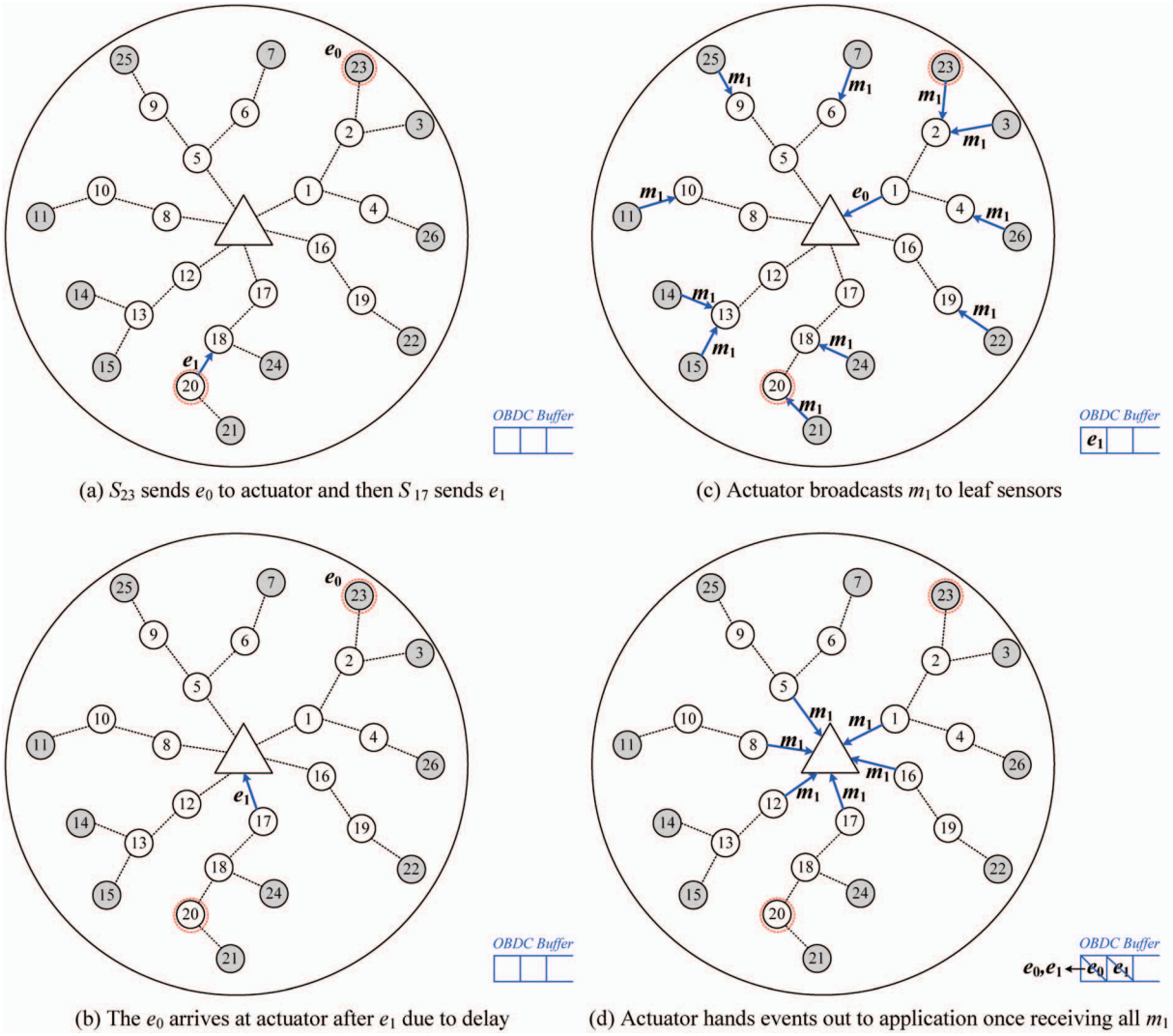

Figure 2. Example of OBDC.

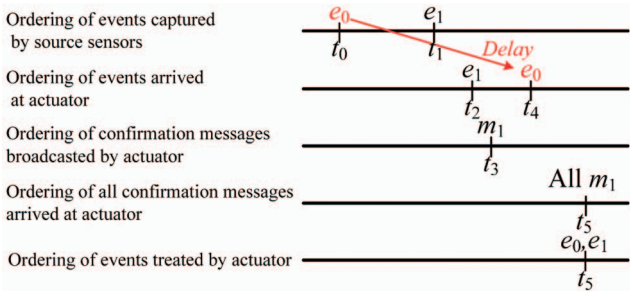

Figure 3. Time sequence diagram of events ordering for OBDC.

group-related events ordering. In fact, events may belong to different groups and each group may be related to other groups. Hence, we proposed GOBDC to ordering group-related events in correct order.

3. Group Events Ordering by Double Confirmations

In WSAN, to design an events ordering algorithm is an important issue since actuators need to interpret events in correct order. Without any delay, the ordering of events arrived at actuator should be identical to *that* of events captured by source sensors. However, delay is prone to be caused by transmission medium accessing, network dynamics, and signal interference in wireless network.

In OBDC [14], an actuator needed to broadcast a corresponding confirmation message to all leaf sensors when the actuator received an event from source sensors.

Once the non-leaf sensors received the corresponding confirmation message, they discarded to broadcast the confirmation message.

The sequence number for each event was assigned by *time-synchronization*. However, it only showed the ordering of events captured by the source sensors. While the delay existed in network, the latter events may arrive at the actuator before the prior events. Since the actuator cannot know whether any prior events were still in transit over network, the actuator may treat the latter events immediately. In this situation, the actuator may perform the incorrect actions because the ordering of events treated by the actuator was different from *that* of events captured by the source sensors. Hence, the events ordering issue cannot be only solved by the *time-synchronization*.

If e_i is delayed to be sent back, the later e_j may arrive in actuator before the delayed e_i ($i < j$). It is due to the unique path and the property of FIFO channel. However, OBDC could not guarantee the rate of the correct events ordering up to 100% for group-related events ordering. Hence, we proposed GOBDC to guarantee the rate of the correct events ordering up to 100% for group-related events ordering.

3.1 Definition of Events Ordering

Here, $(e_i \rightarrow e_j)_h$ was defined that e_i was *happened* before e_j; $(e_i \rightarrow e_j)_a$ was defined that e_i *arrived* at actuator before e_j; and $(e_i \rightarrow e_j)_t$ was defined that e_i was *treated* before e_j. The subscripts h, a, and t were defined as the "happened", "arrived", and "treated", respectively. The m_i was the corresponding confirmation message for e_i. The t_{e_i} and t_{m_i} denoted the happened timestamp of e_i and m_i.

For group-related events ordering, $e_{i,j}$ was defined that the order of event was i^{th} and this event belongs to j^{th} group. The $e_{last,\,j}$ was defined as that the *event* belongs to *last* event of the j^{th} group in the buffer. The $(e_{i,j} \rightarrow e_{i,\,k})_t$ must be true firstly and then $(e_{i,j} \rightarrow e_{k,\,j})_t$ must be true, $i < j < k$. The $m_{i,j}$ was the corresponding confirmation message for $e_{i,\,j}$. The $t_{e_{i,\,j}}$ and $t_{m_{i,\,j}}$ denoted the happened timestamp of $e_{i,\,j}$ and $m_{i,\,j}$.

3.2 Our Scheme GOBDC

OBC and OBDC only considered $(e_{i,\,j} \rightarrow e_{k,\,j})_t$ without addressing $(e_{i,\,j} \rightarrow e_{i,\,k})_t$. Once it had the group ordering relation among events, they could not interpret these events in correct order. Since events ordering may be group-related, GOBDC was designed to guarantee the rate of the correct events ordering up to 100% for group-related events ordering.

GOBDC was based on FIFO property of channels. A pre-configuration clustering was needed. In OBDC [14], it assumed that each sensor could be covered by at least one actuator. Routing path between any two sensors must be existed. Hence, all sensors could be clustered by PEQ [13]. In fact, no one could guarantee that each sensor could be covered by at least one actuator and routing path between any two sensors was existed. To address this issue, we proposed a coverage and connectivity aware clustering algorithm within k-hops, CCAC-k. In CCAC-k, it proved that all sensors could be covered by at least one actuator and no isolated sensor existed. Hence, in this paper, we constructed a cluster topology by CCAC-k [15] around an actuator being as cluster header of cluster. Each sensor was given to be covered by at least one actuator for transmitting events in this cluster. To understand the process of GOBDC easily, we illustrated in the following examples.

In Figure 4, while $e_{0,0}$ arrived at actuator, $m_{0,0}$ was broadcasted to all leaf sensors by the actuator. When the actuator received all $m_{0,0}$, only $t_{e_{0,\,0}}$ was less than $t_{m_{0,\,0}}$. Hence, only $e_{0,0}$ was treated in t_4. While $e_{0,1}$ arrived at the actuator, $m_{0,1}$ was broadcasted to all leaf sensors. After the actuator received all $m_{0,1}$, $e_{0,1}$ and $e_{1,1}$ were in buffer. Since only $t_{e_{0,\,1}}$ was less than $t_{m_{0,\,1}}$, $e_{0,1}$ was treated in t_{12}. In the same way, the ordering of events treated by actuator in OBDC, $(e_{0,0} \rightarrow e_{0,1} \rightarrow e_{1,1} \rightarrow e_{1,0} \rightarrow e_{2,0} \rightarrow e_{2,1})_t$, is incorrect, since $e_{0,1}$ and $e_{1,1}$ are treated before $e_{2,0}$. It was caused that OBDC addressed nothing for group-related events ordering.

In GOBDC, it considered both of events ordering and groups ordering. Hence, $e_{0,1}$ was not treated in t_{12}, because the $e_{last,0}$ was not treated. It violated the rule that $(e_{i,\,j} \rightarrow e_{i,\,k})_t$ must be true firstly and then $(e_{i,\,j} \rightarrow e_{k,j})_t$ must be true, $i < j < k$. Hence, $e_{0,1}$ was not treated in t_{18}. In the same way, ordering of events treated by actuator in GOBDC was correct, such as $(e_{0,0} \rightarrow e_{1,0} \rightarrow e_{2,0} \rightarrow e_{0,1} \rightarrow e_{1,1} \rightarrow e_{2,1})_t$. It showed that GOBDC could ensure the rate of the correct events ordering up to

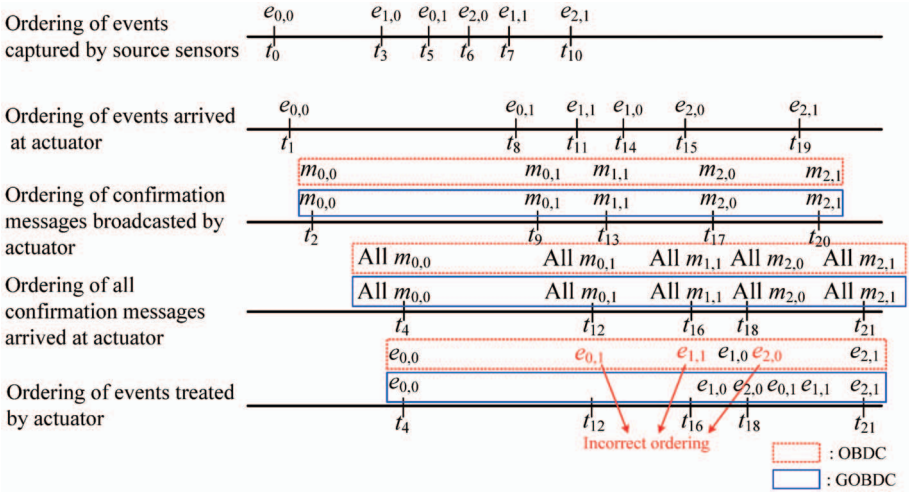

Figure 4. Group-related events ordering in GOBDC and OBDC.

100% but OBDC could not for group-related events ordering.

In OBDC, it addressed nothing for the group-related events ordering. Hence, the events in the same group could be treated in correct order. However, the events in different groups may be treated in incorrect order, as same as OBDC. Hence, the rate of the correct events ordering for group-related events ordering in OBDC and OBDC all could not be up to 100%.

The flow chart of GOBDC was shown in Figure 6. In OBDC, it only addressed the events ordering without the groups ordering. In GOBDC, it considered the events ordering and groups ordering, jointly. While all confirmation messages from the leaf sensors arrived at actuator, the additional blocks needed to be processed for ensuring the groups ordering, such as the grey blocks, as shown in Figure 5. Hence, GOBDC required more time to treat events in correct group-related events ordering, especially while the number of groups increased. However, GOBDC could guarantee the rate of the correct events ordering for group-related events ordering up to 100%.

Once the number of groups was set to one, OBDC was just a special case of GOBDC. Hence, GOBDC could be also applied for non-group-related events ordering and the overhead of GOBDC for non-group-related events ordering was only a little increment.

Although GOBDC could solve the group-related events ordering problem, it may spend more time to complete it. Once the number of different groups or the frequency of delay in network increased, the time of treating the events also increased. It was the limitation of GOBDC.

4. Simulation Results

In the baseline WSN simulation, NS2, OPNET, and OMNET are the general simulators. However, the documentation is constrained and usually lags behind the present simulator release. Moreover, these simulators have mixed compilation and interpretation. Therefore, it is difficult to analyze or understand previous codes when using different versions. Few of them could study the simulation scenario trace files for performance evaluation. However, the people may evaluate different results for the same performance metrics. In addition, these tools constrained the number of nodes and required the large amount of memory. Once the number of nodes exceeds a few hundred, the simulation may be very slow or not even run [16]. Finally, they cannot be applied for the special issues in WSAN, such as events ordering. Hence, many authors [17] have built their own simulators.

The effectiveness of our proposal is effective through simulator we coded in a C# language with the .NET platform. The performance of GOBDC was measured and compared with OBC and OBDC by the simulator. In this section, we will set required simulation parameters, define performance metrics, and analysis the experimental results, respectively. Then we will analyze and discuss the experimental results. Simulation parameters and the

corresponding value(s) were listed in Table 1.

The real scenario could consist of many sub-regions of sensors with different number of nodes. In each sub-region, the sensors were clustered by an actuator by CCAC-k [15]. Since the processing of group-related events ordering in each sub-region was the same, the simulation scenario was for a sub-region that one actuator was included in the simulation. To compare GOBDC with OBC and OBDC fairly, most of the simulation parameters were defined based on OBC [5–8] and OBDC [14], such as the simulation area, number of sensors, transmission radius of sensor, number of events (N), period time for generating a event (T_g), one hop delay, delay time (T_d), packet size (S), and energy consumption per byte (E) were defined as listed in Table 1. Based on CCAC-k, the value of k was set to 4. Hence, the transmission radius of actuator was set to 80 m. Since the number of events (N) could affect R_c, we evaluated R_c in different N from 20 to 80. In the same way, the probability of events with a delay in each link (P) was also an important factor for R_c. Hence, we evaluated R_c in different P from 20% to 80%. To evaluate R_c for group-related events ordering, the number of events in a group (N_g) was defined from 1 to 10. Once N_g increased, the number of groups (NG) may also increase.

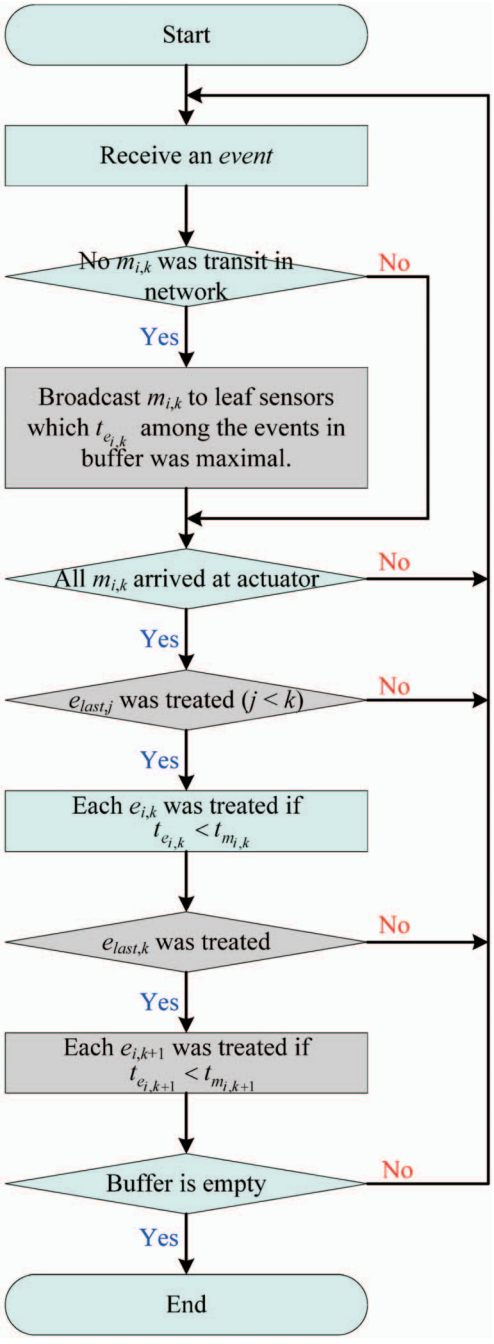

Figure 5. Flow chart of GOBDC.

4.1 Performance Metrics

Rate of correct event ordering (R_c) is the major metric in the events ordering since an error event ordering easily misleads actuator to perform fail actions that further cause serious losses or damage. As defined in (1), R_c is defined as the number of correct ordered events (N_c) divided by N as (1).

$$R_c = \frac{N_c}{N} \times 100\% \tag{1}$$

Table 1. Simulation parameters

Parameter name	Value (Units)
Simulation area	100 × 100 (m^2)
Number of sensors	50
Transmission radius of sensor	20 (m)
Transmission radius of actuator	80 (m)
Number of events (N)	20, 40, 60, 80
Period time for generating a event (T_g)	50–100 (ms)
One hop delay	20 (ms)
Delay time (T_d)	500–1500 (ms)
Packet size (S)	50 (Bytes)
Energy consumption per byte (E)	400 (nJ/Byte)
Probability of events with a delay in each link (P)	20%, 40%, 60%, 80%
Number of events in a group (N_g)	1–10

4.2 Experimental Results

R_c of GOBDC was always up to 100%, but R_c of OBDC and OBC were below 40% averagely, as shown in Figures 6(a)–(d). Next, we varied N to evaluate R_c of OBC, OBDC, and GOBDC, respectively. In Figures 7(a)–(d), it showed that R_c of GOBDC always kept on 100%, but R_c of OBC and OBDC were below 40%. Once N increased, R_c decreased. It proved that GOBDC could ensure that R_c could be up to 100% in group-related events ordering.

The difference of R_c under GOBDC, OBDC, and OBC were larger while N increased. It was caused that OBDC and OBC addressed nothing for group-related events ordering. It showed that GOBDC could outperform OBDC and OBC regarding to R_c even though N or P increased over time.

In our simulation, each *point estimate* was calculated under 1000 rounds. For example, in Figure 6(a), R_c of OBDC was calculated as 32.5% and 31.9%, respectively, in the average case scenario and worst case scenario after 1000 rounds. In the same way, the *standard error* of R_c of GOBDC, OBDC, and OBC all could be within 1%. The *confidence interval* of R_c of them was 95%.

In OBC and OBDC, they never considered the group-related events ordering. Hence, R_c in OBC and OBDC were lower than *that* in GOBDC. In our simulation, the maximal N_g was 10 and the minimum N_g was 1. The probability of each event belonged to a different group was random. Once the number of events (N) increased, the number of groups (NG) may also increase. For example, if N was set to 2, the maximal NG was 2 and the minimum NG was 1. In our simulation, the maximal N was 80 and the minimum N was 20. Hence, the maximal NG may be 80 and the minimum NG may be 2. Once NG increased, R_c in OBC and OBDC decreased. Hence, R_c in GOBDC was over the much low precision of OBC and OBDC for group-related events ordering.

5. Conclusions

WSAN is a new type of heterogeneous wireless sensor networks, where sensors gather environmental information and actuators take actions based on the sensed data from sensors. The past researches always assumed

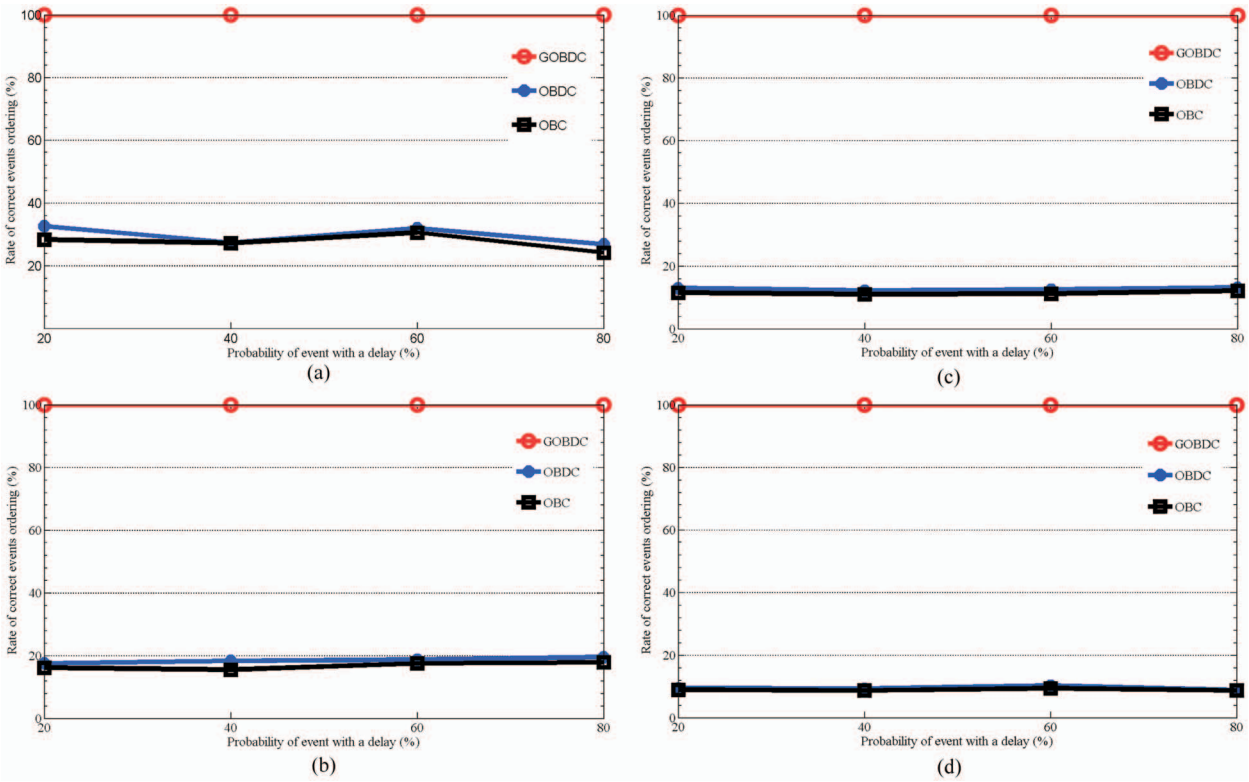

Figure 6. (a) R_c ($N = 20$); (b) R_c ($N = 40$); (c) R_c ($N = 60$); (d) R_c ($N = 80$).

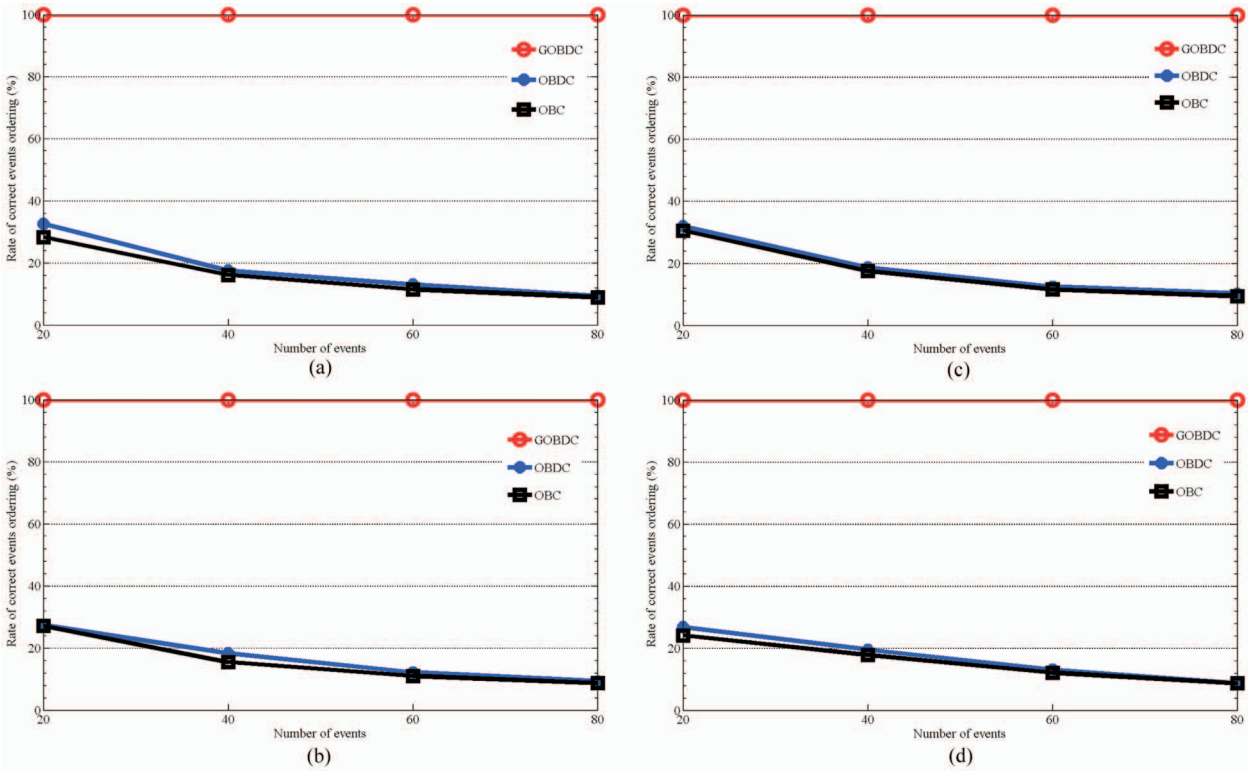

Figure 7. (a) R_c (P = 20%); (b) R_c (P = 40%); (c) R_c (P = 60%); (d) R_c (P = 80%).

that events are not co-related to each other. Hence, most of proposals in WSAN focused on reducing energy consumption and transmission time for routing protocols. Events are interpreted by actuators independently. However, events may be time-related based on the ordering of occurrences of events, since delay in wireless network is prone to be caused, such as network medium access, MAC layer protocol, and dynamic network topology, respectively. Once events are time-related, the correct ordering of events becomes important since actuators may take error actins due to incorrect events ordering.

In the existing solutions for ordering events, such as OBC and OBDC, only the complete-related ordering was considered. However, the co-related events may belong to different groups. Each group may be related to other groups. In this paper, this kind of ordering was defined as group-related events ordering. To address the group-related events ordering issue, GOBDC was proposed in this paper. Differed from OBC and OBDC, GOBDC could guarantee R_c up to 100% for group-related events ordering.

The simulation results demonstrated that R_c of GOBDC was always up to 100%, but R_c of OBC and OBDC was below 40% even if N or P varied for group-related events ordering. It proved GOBDC could be applied for the group-related events ordering.

In the real environment, the relation of events ordering may include other kind of ordering. Therefore, we will further investigate other events ordering issues and propose more efficient events ordering algorithms to address these issues.

References

[1] Akyildiz, I. F. and Kasimoglu, I. H., Wireless Sensor and Actor Networks: Research Challenges. Ad Hoc Networks, Vol. 2, No. 4, pp. 351–367 (2004). doi: 10.1016/j.adhoc.2004.04.003

[2] Munir, M. F. and Eurecom, I., Wireless Sensor and Sensor-Actuator Networks Research Trends, Protocols, and Applications, 2005 Networking and Communications Conference, 2005 May 1–3; Lahore, Pakistani, IEEE; pp. 6–6 (2008). doi: 10.1109/INCC.2008.4562673

[3] Melodia, T., Pompili, D. and Akyildiz, I. F., A Communication Architecture for Mobile Wireless Sensor and Actor Networks, 2006 Sensor and Ad Hoc Communications and Networks Conference, 2006 September 28; Reston, VA, IEEE, pp. 109–118 (2006). doi: 10.1109/SAHCN.2006.288415

[4] Melodia, T., Pompili, D., Gungor, V. C. and Akyildiz, I. F., "Communication and Coordination in Wireless Sensor and Actor Networks," IEEE Transaction on Mobile Computing, Vol. 6, No. 10, pp. 1116–1129 (2007). doi: 10.1109/TMC.2007.1009

[5] Boukerche, A., Silva, F. H. S., Araujo, R. B. and Pazzi, R. W. N., A Low Latency and Energy Aware Event Ordering Algorithm for Wireless Actor and Sensor Networks, 2005 Modeling Analysis and Simulation of Wireless and Mobile Systems Conference, 2005 October 10–13; Quebec, Canada, ACM, pp. 111–117 (2005). doi: 10.1145/1089444.1089465

[6] Boukerche, A. and Martirosyan, A., An Efficient Algorithm for Preserving Events' Temporal Relationships in Wireless Sensor Actor Networks, 2007 Local Computer Networks Conference, 2007 October 15–18, Dublin, IEEE, pp. 771–780 (2007). doi: 10.1109/LCN.2007.153

[7] Boukerche, A., Araujo, R. B. and Silva, F. H. S., "An Efficient Event Ordering Algorithm that Extends the Lifetime of Wireless Actor and Sensor Networks," *Performance Evaluation*, Vol. 64, No. 5, pp. 480–494 (2007). doi: 10.1016/j.peva.2006.08.009

[8] Boukerche, A., Araujo, R. B., Silva, F. H. S. and Villas, L., "Wireless Sensor and Actor Networks Context Interpretation for the Emergency Preparedness Class of Applications," *Computer Communications*, Vol. 30, No. 13, pp. 2593–2602 (2007). doi: 10.1016/j.comcom.2007.05.053

[9] Lamport, L., "Time, Clocks and the Ordering of Events in a Distributed System," *Communications of the ACM*, Vol. 21, No. 7, pp. 558–565 (1978). doi: 10.1145/359545.359563

[10] Samani, M. M. and Sloman, M., "GEM: A Generalized Event Monitoring Language for Distributed Systems," *Distributed Systems Engineering Journal*, Vol. 4, No. 2, pp. 96–108 (1997). doi: 10.1088/0967-1846/4/2/004

[11] Raynal, M., Schiper, A. and Toueg, S., "The Causal Ordering Abstraction and a Simple Way to Implement It," *Information Processing Letters*, Vol. 39, No. 6, pp. 343–350 (1991). doi: 10.1016/0020-0190(91)90008-6

[12] Romer, K., Temporal Message Ordering in Wireless Sensor Networks, 2003 Ad-Hoc Networks Conference, 2003 June 25–27; Mahdia, Tunisia, IFIP, pp. 131–142 (2003).

[13] Boukerche, A., Pazzi, R. N. and Araujo, R. B., A Fast and Reliable Protocol for Wireless Sensor Networks in Critical Conditions Monitoring Applications, 2004 Analysis and Simulation of Wireless and Mobile Systems Conference, 2004 October 4–6; Venice, Italy, ACM, pp. 157–164 (2004). doi: 10.1145/1023663.1023692

[14] Tuan, C.-C. and Wu, Y.-C., "Event Ordering by Double Confirmation in Wireless Sensor and Actor Networks," *IEEE Sensors Journal*, Vol. 11, No. 3, pp. 829–836 (2011). doi: 10.1109/JSEN.2010.2085431

[15] Tuan, C.-C. and Wu, Y.-C., "Coverage and Connectivity Aware Clustering Algorithm within *k*-Hops in Wireless Sensor and Actor Networks," *Science China Information Sciences*, Vol. 57, No. 6, pp. 120–136 (2014). doi: 10.1007/s11432-013-5022-3

[16] Mallapur, S. V. and Patil, S. R., "Survery on Simulations Tools for Mobile Ad-Hoc Networks," *International Journal of Computer Networks and Wireless Communications*, Vol. 2, No. 2, pp. 241–248 (2012).

[17] Abedi, R. H., Aslam, N. and Ghani, S., Fault Tolerance Analysis of Heterogeneous Wireless Heterogeneous Sensor Network, 2011 Electrical and Computer Engineering Conference, 2011 Niagara Falls, ON, IEEE, pp. 175–179 (2011). doi: 10.1109/CCECE.2011.6030433

Manuscript Received: Mar. 2, 2016
Accepted: Jun. 17, 2016