J Inf Process Syst, Vol.17, No.1, pp.28~36, February 2021
https://doi.org/10.3745/JIPS.03.0155

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Simulation of a Mobile IoT System Using the DEVS Formalism

Jung Hyun Im*, Ha-Ryoung Oh*, and Yeong Rak Seong**

Abstract

This paper proposes two novel methods to model and simulate a mobile Internet of Things (IoT) system using the discrete event system specification (DEVS) formalism. In traditional simulation methods, it is advantageous to partition the simulation area hierarchically to reduce simulation time; however, in this case, the structure of the model may change as the IoT nodes to be modeled move. The proposed methods reduce the simulation time while maintaining the model structure, even when the IoT nodes move. To evaluate the performance of the proposed methods, a prototype mobile IoT system was modeled and simulated. The simulation results show that the proposed methods achieve good performance, even if the number of IoT nodes or the movement of IoT nodes increases.

Keywords

Discrete Event System Specification, Mobile IoT System, Modeling and Simulation

1. Introduction

Internet of Things (IoT) systems are widely used in many different ways. There are various methods available to analyze the performance and to verify the operation of these systems; one of the most common is modeling and simulation. There are many modeling and simulation methods [1-4], including the Discrete Event System Specification (DEVS) formalism [3,4], which are typically used for discrete event systems. The DEVS formalism can be used to model and simulate IoT systems because they are discrete event systems. This formalism defines a system in a modular and hierarchical manner. It specifies the structure of the system using coupled models, and the behavior of the system using atomic models. Thus, the DEVS formalism can fully represent a system [4-6].

One of the most difficult tasks in the simulation of IoT systems is simulating the broadcasting property of wireless communication. Im et al. [7] proposed a useful method to simulate “stationary IoT systems” in which the locations of IoT nodes are fixed. In this method, the simulation target area is recursively partitioned into smaller areas to reduce the number of events generated while simulating wireless communication between IoT nodes. However, the method has difficulty simulating a “mobile IoT system” in which IoT nodes move freely, and the simulation time increases greatly. The reason it is difficult to simulate a mobile IoT system is that the structure of the simulation model may change as the

Manuscript received September 4, 2020; first revision November 18, 2020; accepted December 4, 2020.
Corresponding Author: Yeong Rak Seong (yeong@kookmin.ac.kr)
* Dept. of Secured-Smart Electric Vehicle, Kookmin University, Seoul, Korea (ijh227@kookmin.ac.kr, hroh@kookmin.ac.kr)
** Dept. of Electric Engineering, Kookmin University, Seoul, Korea (yeong@kookmin.ac.kr)

IoT node moves. To address the difficulties, many studies have been attempted to formally express these systems by extending the DEVS formalism [8-13]. However, even modelers familiar with formal modeling methodologies, including the DEVS formalism, may find another difficulty in learning and understanding the new formalism. It is also a burden to adapt to a new modeling and simulation environment implementing the new formalism.

This paper proposes two practical solutions for simulating a mobile IoT system using the DEVS formalism. For simplicity, we assume here that IoT nodes can only receive communication messages, but the proposed methods can be easily extended and applied even if IoT nodes transmit messages.

2. Previous Work

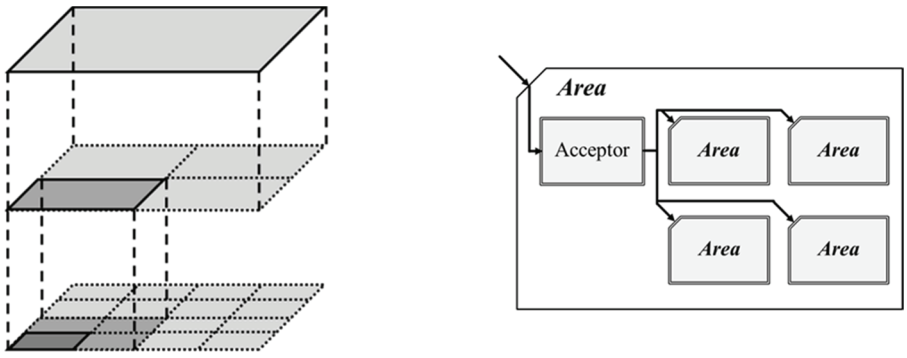
When an IoT system is simulated using the DEVS formalism, external transition events can be massively over-generated, and the simulation time increases accordingly. The fundamental reason for this over-generation is that an IoT node communicates wirelessly with other nodes in many IoT systems. In the real world, when a device transmits a wireless packet, the packet is only transferred to neighboring IoT nodes, because the transmission power of the device is limited. In modeling and simulating this situation, however, external transition events representing the packet would be transferred to every ***IoT*** model (Note that the names of the DEVS models appear in bold italics). This is because the model of the transmitter device does not know the locations of the receiver IoT nodes due to the modular nature of the DEVS formalism. After receiving the external transition event, an ***IoT*** model can decide whether or not the IoT node associated with the model receives the wireless packet in the real world by using the location information of the transmitter contained in the external transition event message. If the associated IoT node can receive the packet, the ***IoT*** model processes the event; otherwise, the ***IoT*** model discards the event. The problem is that every ***IoT*** model should always process these external events, regardless of whether the associated IoT node is able to receive the packets. Thus, the number of over-generated external transition events tends to be enormous.

In [7], we proposed the hierarchical partitioning (HP) method to reduce the over-generated external transition events for stationary IoT systems, in which the locations of IoT nodes are fixed. The method comprises two key ideas.

First, to exploit the hierarchical and modular nature of the DEVS formalism, the entire simulation target area is recursively partitioned into smaller areas, and then each area is modeled as an ***Area*** coupled model. Fig. 1(a) shows an example. The entire area in the top layer is partitioned again four sub-areas in the middle layer, and each of these sub-areas is partitioned into four smaller sub-areas in the bottom layer. Consequently, the entire simulation target area is partitioned into $N^{(L-1)}$ sub-areas by this process, where N is the number of sub-areas in each partition process, and L is the number of layers (including the original area). Each ***Area*** coupled model, except those in the bottom layer, contains N children ***Area*** coupled models. A bottom-layer ***Area*** coupled model includes ***IoT*** models instead of ***Area*** coupled models. Note that because the number of IoT nodes in a restricted area is not fixed, the number of ***IoT*** models in a bottom-layer ***Area*** coupled model is also not fixed.

Second, an ***Acceptor*** atomic model is employed in the ***Area*** model. The role of the ***Acceptor*** model is the selective forwarding of external transition events, which are produced for simulating wireless communication, to every sibling model (***Area*** model or ***IoT*** model on the same level). Namely, when

receiving such an external transition event, an ***Acceptor*** model determines whether or not any descendent ***IoT*** models of its parent Area model can receive the broadcasted communication packet represented by the external transition event. If they can, the ***Acceptor*** model forwards the event to every sibling model; otherwise, it discards the event. For this purpose, an ***Acceptor*** atomic model stores the coverage of its parent ***Area*** model. The coverage of an ***Area*** model is defined as the position of the partitioned area, which is specified by the ***Area*** model. Because a coupled DEVS model cannot specify behavior nor store information, the ***Acceptor*** atomic model was devised and included in each ***Area*** coupled model, except for that in the top layer, as shown in Fig. 1(b). Using the above methods, the number of external transition events is drastically reduced, and the simulation time is correspondingly decreased.



(a) (b)

Fig. 1. (a) Hierarchical partitioning of the simulation area and (b) structure of an ***Area*** model from [7].

3. Proposed Methods

This section proposes two novel methods to model and simulate a mobile IoT system using the DEVS formalism. Here the locations of IoT nodes are changed during the simulation time. Our study is based on the HP method, with the IoT nodes moving among the partitioned areas. Because the HP method is devised for a stationary IoT system, it has an underlying limitation for simulating mobile IoT systems. First, we explain the limitation and then propose two novel methods to overcome it.

3.1 Limitation of the HP Method

The DEVS formalism separately specifies the behavior and structure of a system by using atomic models and coupled models, respectively. This is of great benefit for complexity control in modeling and simulation. Unfortunately, however, this becomes a major barrier for simulating mobile IoT systems using the HP method.

To model an IoT system using the DEVS formalism, it is natural that each IoT node is specified by an atomic or a coupled model as in [7], since an IoT node displays a complex, dynamic behavior in processing wireless communication packets. In [7], each partitioned area was modeled with an ***Area*** model, and an IoT node located in a partitioned area was modeled as a component model of the ***Area*** model for that specific area. Thus, if an IoT node moves from one area to another, the components of the ***Area*** models specifying the two areas must be updated. This means that the structure of the coupled

model must be changed. In the DEVS formalism, the model structure is specified statically, so it is difficult to reflect the dynamism. Many modeling methodologies have been proposed to address dynamism in the model structure [8-13]. These methods are extensions of the DEVS formalism. However, even modelers familiar with formal modeling methodologies, including the DEVS formalism, may have difficulty learning and understanding a new formalism. It is also a burden to adapt to a new modeling and simulation environment implementing the new formalism.

3.2 The Variable Coverage Method

To overcome this limitation, we propose a simple solution referred to as the variable coverage (VC) method. The method is built by modifying the HP method. The main difference is that the coverage of an ***Area*** model is no longer fixed. With the VC method, the coverage is dynamically changed when one of the descendent ***IoT*** models of the ***Area*** model moves to a different location. For that, the ***Acceptor*** model is modified to manage the coverage of its parent ***Area*** model. For simplicity, the coverage is represented by a rectangle in which all IoT nodes included in the ***Area*** model are placed. Whenever a descendent ***IoT*** model of an ***Area*** model moves to a different location, the Acceptor model included in the ***Area*** model updates the coverage.

Unfortunately, the VC method has a disadvantage. As one can predict, even if only a single ***IoT*** model roams the entire simulation area, the coverage of its ancestor ***Area*** models could be expanded to the entire simulation area; thus, the benefits of hierarchical partitioning of the simulation area would be reduced.

3.3 The Active Event Method

This section proposes a practical and efficient solution for the simulation of mobile IoT systems referred to as the active event (AE) method. The key idea of this method is to specify IoT nodes as external transition events instead of models. In this way, wireless communication can be efficiently simulated without changing the structure of the coupled models.

The AE method was developed in response to the following points. First, to model the complex and dynamic behavior of an IoT node, it is natural to specify an IoT node by a model. However, the structure of the simulation model might not remain fixed during the simulation. Second, an external transition event can be moved freely between models in the DEVS formalism. Finally the DEVS abstract simulator algorithm [14,15], implemented in DEVS simulation environments, interprets the dynamics of a typical DEVS model. Within DEVSim++ [14,15], a well-known DEVS simulation environment, each model is associated with a corresponding abstract simulator in a one-to-one manner. There are two types of abstract simulators: a simulator for an atomic model and a coordinator for a coupled model.

From these points, we developed two ideas: (1) if an IoT node is modeled using the DEVS formalism, but implemented as an external transition event rather than a model, the movement of the IoT node can be simulated while preserving the structure of the coupled models; and (2) if the ***Area*** model on the bottom layer has the simulator algorithm embedded, it can interpret the behavior of the IoT node implemented as an external transition event. To perform the second idea efficiently, ***Area*** coupled models are replaced with ***Cell*** atomic models, explained later. An advantage of this method is that an IoT node is still specified by the atomic model's DEVS formalism until it is transformed into an event. In other words, we can reduce unnecessary external transitions in the simulation of wireless communication while maintaining the benefits of the DEVS formalism.

The transformation from an IoT model to an external transition event is easily accomplished. In the DEVS formalism, an atomic model is specified by a set of input ports, a set of output ports, a set of state variables, an external transition function δ_{ext}, an internal transition function δ_{int}, an output function λ, and a time advance function *ta*. With this transformation, the input and output port sets are omitted. However, even though the IoT is no longer an atomic model, the state variables set as well as the four characteristic functions are preserved.

Fig. 2 shows the structure of the ***Cell*** atomic model, which replaces the ***Area*** coupled model on the bottom layer in the VC method. The external transition function of the ***Cell*** model receives two types of external transition events. The first type represents the migration of an IoT node from a neighboring ***Cell*** model. For migration, a ***Cell*** model is connected to four neighboring ***Cell*** models through north, east, west, and south input/output ports. When an external transition event containing an IoT node arrives at a ***Cell*** model through N_{in}, E_{in}, W_{in} or S_{in} ports, the external transition function of the ***Cell*** model adds the IoT node to the *IoT_pool*, a state variable of the ***Cell*** model. As a result, the *IoT_pool* stores all the IoT nodes located in the partitioned area specified by the ***Cell*** model. The second type of external transition event indicates the arrival of a wireless communication packet. These events arrive through the P_{in} input port. When an event of this type arrives at a ***Cell*** model, the external transition function of the ***Cell*** model forwards the event to each IoT node stored in its *IoT_pool*. The external transition function of the IoT node is then executed to simulate wireless communication.

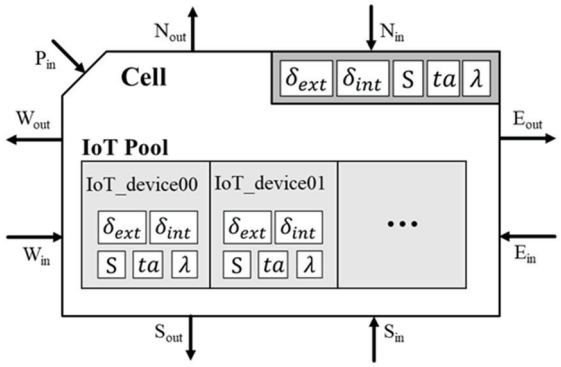


Fig. 2. Structure of the ***Cell*** model.

When an IoT node stored in the *IoT_pool* needs to move from the corresponding partitioned area to one of the neighboring partitioned areas, an internal transition of the IoT node is scheduled. The scheduled time is announced to the ***Cell*** model by using the time advance function of the IoT node. This function of the ***Cell*** model then sets its next event time to the minimum value of the announced IoT nodes' scheduled times stored in the *IoT_pool*. When an internal transition event arrives at a ***Cell*** model, the output function of the ***Cell*** model is executed first. This function invokes the output function of the imminent IoT node, this is the IoT node that has the minimum scheduled time, attaches this node to the external transition event, and sends the event to the N_{out}, E_{out}, W_{out}, or S_{out} port. Finally, the internal transition functions of the ***Cell*** model is executed. This function invokes the internal transition function of the imminent IoT node, removes this node from the *IoT_pool*, and schedules the next internal transition.

4. Experiments and Analysis

To evaluate the performance of the proposed VC and AE methods, the mobile IoT system described in Section 2 was implemented and simulated using DEVSim++ [14,15]. The HP method is excluded from these experiments because it is fundamentally inadequate for simulating mobile IoT systems. To express the main idea of the proposed method as simple as possible, we set $N = 4$ and $L = 3$ in this simulation. That is, the top-level area, i.e., the whole simulation target area, was partitioned into four middle-level sub-areas, and then each of the middle-level sub-area is partitioned again into four bottom-level sub-areas as shown in Fig. 1(a). However, these parameters may vary depending on the number of simulated mobile IoT nodes and the characteristics of the application.

The actual simulation time may vary according to environmental conditions. Thus, we measured the total number of external transition events. The average number was computed over 10 simulation runs in which a total of 1,000 wireless communication packets were generated.

Two major factors affect the number of external transition events generated by wireless communication. The first is the number of IoT nodes. Although the number of wireless communication packets is fixed, the number of external transition events depends on the number of IoT nodes receiving each packet. The second is the mobility of the IoT nodes. The mobility of an IoT node indicates the probability that the node migrates from the current ***Area*** model to one of the neighboring ***Area*** models. In this experiment, an IoT node had between 100 and 200 opportunities to move its location during a simulation run. If the mobility is 0.3, an IoT node migrates to a neighboring ***Area*** model in 30% of the opportunities. Recall that the ***Area*** model in the VC method and the ***Cell*** model in the AE method send an external transition event representing a wireless communication packet to all children models, even if only one of the children models is located in the communication range of the packet. As IoT nodes move, the coverage of an ***Area*** model grows, and the number of external transition events for simulating wireless communication also increases. On the other hand, the coverage of a ***Cell*** model is fixed during the simulation, although the number of IoT nodes included in the *IoT_pool* of the ***Cell*** model might change. Therefore, in the AE method, as the mobility increases, the number of external transition events representing wireless communication is almost unchanged, but the number of external transition events representing the movement of IoT nodes increases. Note that the smaller the increase in the number of external transition events with increasing number of nodes, the better the algorithm is at reducing the simulation time.

4.1 Experiment 1: Number of External Transition Events vs. Number of IoT Nodes

In the first experiment, we compared the performance of the VC and the AE methods as a function of the number of IoT nodes. The number of IoT nodes was varied from 100 to 1,000 in steps of 100. The mobility was set to its maximum value of 1, to maximize the effect of the movement of IoT nodes on the simulation time.

Fig. 3 shows the results of Experiment 1. As the number of IoT nodes increases, the AE method can significantly reduce the number of external transition events compared to the VC method. The number of external transition events generated by the VC method increases slowly when the number of IoT nodes is small, but then increases rapidly after the number of IoT nodes reaches a certain threshold (400 in this experiment). In contrast, using the AE method, as the number of IoT nodes increases, the number of external transition events generated increases almost linearly.

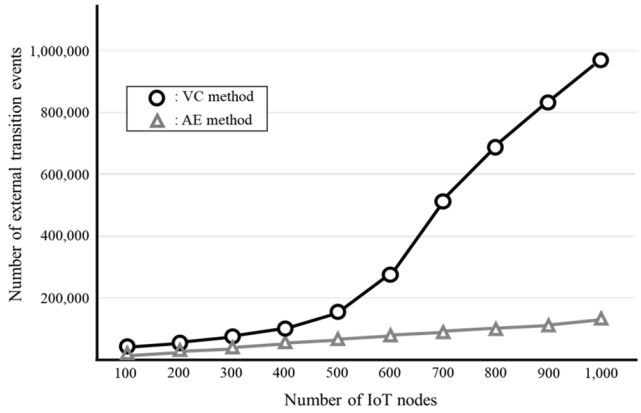


Fig. 3. Number of external transition events as a function of number of IoT nodes.

4.2 Experiment 2: Number of External Transition Events vs. Mobility of IoT Nodes

In the second experiment, the performance of the proposed methods was tested as a function of mobility. We increased the mobility from 0.1 to 1.0 in steps of 0.1, and set the number of IoT nodes to 800 for all simulation runs. For completeness, a trivial simulation method that does not partition the simulation area was compared with the proposed methods.

Fig. 4 shows the results of Experiment 2. As the mobility of IoT nodes increases, the number of external transition events in the AE method hardly increases. This is because the number of external transition events due to the movement of IoT nodes increases, but the number of external transition events due to wireless communication packets hardly changes. In contrast, the number of external transition events in the VC method increases with increasing mobility. Note that in the non-partitioning method, an external transition event representing a wireless communication packet is forwarded to all IoT nodes. Thus, in this method the number of external transition events does not depend on the mobility, and provides an upper limit of the number of external transition events. With low mobility, the VC method performs only slightly worse than the AE method. However, as the mobility increases, the performance of the VC method rapidly degrades.

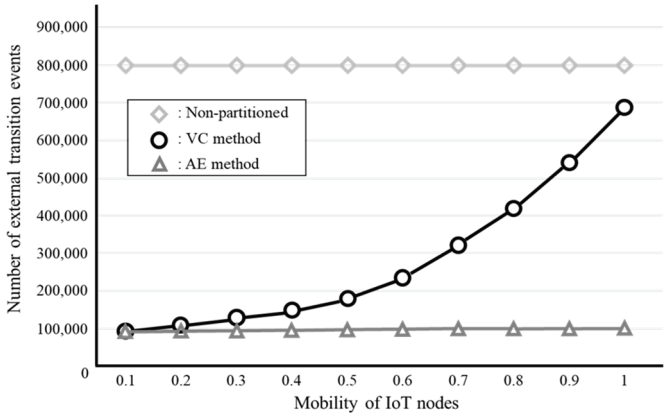


Fig. 4. Number of external transition events as a function of mobility of IoT nodes.

5. Conclusion

In this paper, we proposed two novel methods for modeling and simulation of mobile IoT systems using the DEVS formalism. These methods are based on hierarchical partitioning of the simulation area. In the VC method, each partitioned area is specified by an ***Area*** coupled model, and each IoT node is also specified by a DEVS model. To simulate the movement of IoT nodes between the partitioned areas, the coverage of the ***Area*** model is dynamically extended. In the AE method, each sub-area within the bottom layer of the partition hierarchy is specified by a ***Cell*** atomic model and each IoT node is represented by an external transition event. The coverage of the ***Cell*** model is fixed, while the IoT nodes move between the partitioned areas. An advantage of the AE method is that an IoT node is still specified by the atomic DEVS formalism until it is transformed into an event. Experimental results show that the AE method always outperforms the VC method. If the number of IoT nodes is small and the mobility of IoT nodes is low, the performance gap between the two methods is small. However, as the number of IoT nodes or the mobility of IoT nodes increases, the AE method provides much better performance. So, we conclude that the AE method is a powerful method for simulating mobile IoT systems, but that the simpler VC method is useful when the number of IoT nodes is small or the mobility of IoT nodes is low.

Although the proposed methods focus on mobile IoT systems in this paper, they can be used for many other applications that suffer from a long simulation time due to dynamic changes in the model structure. We are currently developing an intelligent traffic information system. The proposed AE method is being utilized to evaluate the performance of the system. In the future, we would like to apply the proposed method to many other applications.

References

[1] B. P. Zeigler, A. Muzy, and E. Kofman, *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*, 2nd ed. San Diego, CA: Academic Press, 2000.

[2] B. P. Zeigler, A. Muzy, and E. Kofman, *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*, 3rd ed. San Diego, CA: Academic Press, 2018.

[3] G. A. Wainer and P. J. Mosterman, *Discrete-Event Modeling and Simulation: Theory and Applications*. Boca Raton, FL: CRC Press, 2016.

[4] B. K. Choi and D. H. Kang, *Modeling and Simulation of Discrete Event Systems*. Hoboken, NJ: John Wiley & Sons, 2013.

[5] B. P. Zeigler, "DEVS representation of dynamical systems: event-based intelligent control," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 72-80, 1989.

[6] T. G. Kim and S. B. Park, "The DEVS formalism: Hierarchical modular systems specification in C++," in *Proceedings of the 1992 European Simulation Multiconference*, York, UK, 1992, pp. 152-156.

[7] J. H. Im, H. R. Oh, and Y. R. Seong, "Reducing the number of events in simulation of wireless networks," in *Proceedings of Symposium of the Korean Institute of Communications and Information Sciences*, Seoul, Korea, 2015, pp. 676-677.

[8] F. J. Barros, "Dynamic structure discrete event system specification: a new formalism for dynamic structure modeling and simulation," in *Proceedings of Winter Simulation Conference*, Arlington, VA, 1995, pp. 781-785.

[9] F. J. Barros, "Modeling formalisms for dynamic structure systems," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 7, no. 4, pp. 501-515, 1997.

[10] A. M. Uhrmacher, "Dynamic structures in modeling and simulation: a reflective approach," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 11, no. 2, pp. 206-232, 2001.

[11] H. Khalil and G. Wainer, "Cell-DEVS for social phenomena modeling," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 3, pp. 725-740, 2020.

[12] M. Zhu, C. Foucher, V. Albert, and A. Nketsa, "Applying the model-driven architecture approach to dynamic structure applications," in *Proceedings of the 31st European Simulation and Modelling Conference (ESM)*, Lisbon, Portugal, 2017.

[13] M. J. Blas, S. M. Gonnet, H. P. Leone, and B. P. Zeigler, "A conceptual framework to classify the extensions of DEVS formalism as variants and subclasses," in *Proceedings of 2018 Winter Simulation Conference (WSC)*, Gothenburg, Sweden, 2018, pp. 560-571.

[14] T. G. Kim, "DEVSIM++ user's manual," Department of Electrical Engineering, KAIST, Daejeon, Korea, 1994.

[15] T. G. Kim and C. Choi, "DEVSim++ ME: HLA-compliant DEVS modeling/simulation environment with DEVSim++," in *Model Engineering for Simulation*. Cambridge, MA: Academic Press, 2019, pp. 355-392.

Jung Hyun Im **https://orcid.org/0000-0001-8626-301X**

Im received a B.S. degree in Electrical Engineering from Kookmin University and an M.S. degree in Secured-Smart Electric Vehicles from Kookmin University. He is currently a graduate student in the Department of Secured-Smart Electric Vehicles at Kookmin University, Seoul, Korea. His current research interests are in the areas of discrete event system modeling and simulation, embedded systems, and machine learning.

Ha-Ryoung Oh **https://orcid.org/0000-0002-7991-4010**

Oh received a B.S. degree in electrical engineering from Seoul National University, Seoul, Korea, in 1983 and M.S. and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 1988 and 1992, respectively. Since 1992, he has been a professor with Kookmin University, Seoul, Korea. His current research interests include RFID systems, wireless sensor networks, machine learning and embedded systems.

Yeong Rak Seong **https://orcid.org/0000-0003-2453-2653**

Seong received a B.S. degree in electrical engineering from Hanyang University, Seoul, Korea, in 1989 and M.S. and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 1991 and 1995, respectively. Since 1996, he has been a professor with Kookmin University, Seoul, Korea. His current research interests include RFID systems, wireless sensor networks, machine learning and embedded systems.