Journal of The Korea Society of Computer and Information
Vol. 21 No. 10, pp. 99-105, October 2016
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2016.21.10.099

Active Rule Manager for the Mobile Agent Middleware System

Yon-Sik Lee*, Eun-Hong Cheon**

Abstract

The active rule system is a key element of the rule-based mobile agent middleware system for activeness and autonomy of the sensor network. The rule manager, which is the main components of active rule based mobile agent framework and active rule system, performs the control and management of the rule-related processes. In this paper, we design and implement the roles and functions of the rule manager in detail. The proposed rule manager plays an important role in the sensor network environment. The sensor data server loads the active rule on the mobile agent by the rule manager according to the situations, and the mobile agent migrates to the destination node and performs the designated action. This active rule-based mobile agent middleware system presents the usefulness for the various sensor network applications. Through the rule execution experiment using the rule-based mobile agent, we show the adaptability and applicability of rule-based mobile agent middleware system to the dynamic environmental changes in sensor networks.

▸ Keyword : Active Rule System, Rule Manager, Sensor Network Middleware, Mobile Agent

I. Introduction

IT-based control mechanism, using mobile agent framework with active rule, is a suitable paradigm that allows consumer-oriented control [1,2,4]. In daily life, the user's emotion-based illumination technique plays an important role in power saving control. To obtain the optimal power saving effect and illumination standards, the smart power saving techniques are required in sensor network environment [5,7,8]. It is suitable for the active rule system to derive smart power saving by adjusting the illuminance using active rules in accordance with the user's emotion. For this, we suggest the active rule-based mobile agent middleware system. The mobile agent in the system migrates to the destination sensor nodes, acquires and transmits sensor data according to the user's needs through the active rules, and directly executes the actions corresponding to the optional events. In this paper, we design and implement the roles and functions of the rule manager, which is a main module of the active rule system located in the sensor data server. The rule manager is the main component of the active rule-based mobile agent middleware system that provides activeness and autonomy of the sensor network.

II. Active Rule and Mobile Agent

ECA Rule is a rule of performing an action when an event occurs, after evaluating conditions and the result is satisfactory [1,2]. The basic structure of an event is as

• First Author: Yon-Sik Lee, Corresponding Author: Eun-Hong Cheon
*Yon-Sik Lee (yslee@kunsan.ac.kr), School of Computer Information and Communication Engineering, Kunsan National University
**Eun-Hong Cheon (ehcheon@woosuk.ac.kr), Department of Computer Engineering, Woosuk University
• Received: 2016. 05. 23, Revised: 2016. 07. 05, Accepted: 2016. 09. 30.
• This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (13A13907331) and R&D Ability Reinforcement Program funded by Jeollabukdo Business Agency (4-066)

follows; *<ID, Type, Start time, End time, Enable period, Location, Node ID, [(<Parameter>,…)]>*. Normally, an event begins according to the occurrence time of the sensing data. Conditions of the rule consist of random predicates or arithmetic operations. Action executor performs the actions specified in the rule according to condition evaluation, and these actions can be classified as single sentence, random sensor node control, or externally applied behavior [1,2,3]. Execution process of the rule is iterative in line with system request after an event has occurred.

Migration of the mobile agent with active rule takes place by radio communication and active message structure of TinyOS. TinyOS defines an active message structure corresponding to the packet and frame structure of IEEE 802.15.4 and provides it to users. Migration occurs using this function as TinyOS incorporates a mobile agent 'mobileAgent_t' into an active message 'message_t' before sending. Migration method may be in consecutive order or without order when necessary, and uses user-defined order and information from MetaTable, in accordance with received node ID, in sink node [7].

As Table 1 shows, one sink node (sink number 0) and a *SubMetaData* linked with *SubMetaTables* give information on six sensor nodes. In addition, *useSensorNumber* of *nodeInfo* shows the number of sensors in use at the current sensor node, and *useArea* shows the location of the sensor node.

Table 1. *MetaTable* information

host name	host URL	keyWord	pointers to SubMetaTables
Data Server	"210.118.34.37"	"Data"	s1
sink nodeID	sink Number	sinkInfo	Sink URL
s1	0	0	"210.118.34.37::0"
sensor node ID	node Number	nodeInfo: use Sensor Number	nodeInfo: use Area	nodeAddr.
1	1	2	8	"210.118.34.12::0.1"
2	2	2	2	"210.118.34.12::0.2"
3	3	1	3	"210.118.34.12::0.3"
4	4	2	4	"210.118.34.12::0.4"
5	5	1	5	"210.118.34.12::0.5"
6	6	1	7	"210.118.34.12::0.6"

The mobile agent sent from a sink node executes the active rule as it follows along the migration pathway. Active rules (gathering, transmitting, managing of sensor data and device control) loaded on the mobile agent are executed according to the value of sensing data at the sensor node. Furthermore, if an additional node for rule execution exists, migration and rule execution process continue in accordance with the user-defined order, and upon termination, the user-defined order is removed automatically [7,8].

III. Active Rule System

Active rule-based mobile agent middleware system uses mobile agent to perform active tasks in relation to sensor data such as remote automatic operation and control of resource utilization.

Fig.1 shows the active rule system in sensor data server that is linked with mobile agent middleware system [2,7]. Active rule system as an applied module is composed of subsystems of rule components (event, condition, and action) and processes sensing information coming from sensor nodes or sink nodes as an event and executes necessary actions by linkage with existing database and rule base[1,2]. And, its type may vary depending on the characteristics of a system [3].

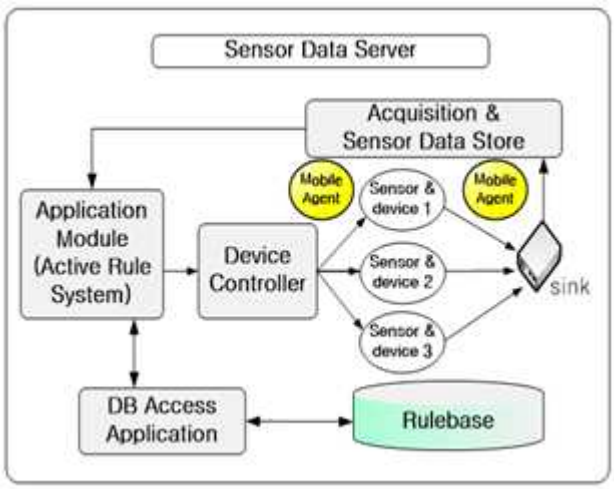

Fig. 1. Active rule system in sensor data server

The role of sensor data server is to handle sensor data storage and process, execute user's requests, and modify rule based on the command of the manager. Gathered data is used for event detection and situation classification, and if the data is unnecessary, it is removed or used by rule manager according to the rule base. Action executor performs actions, such as device control and data store in the database, after confirming

the network and related situations [7].

Rule manager controls the entire procedures about the rules. The main functions of the rule manager are registration/deletion of a rule, regulation among each rule components, management of rule process, providing rule-related interface to users, and enabling/disabling rule at the occurrence of an event [2,3]. The event process is divided into following two types, and its flow is shown in Fig.2.

1) Process of active rule loaded on mobile agent: A mobile agent periodically collects data from illumination sensor and sends it to sink node. Then, the sensor data server classifies the information by different situations (timeslot, presence of user, etc.) and loads the active rule on the mobile agent. Then, it migrates the mobile agent for execution of the action (on/off or dimming control) to the destination node.

2) Process of active rule stored in rule base: As shown in Fig.2, a rule is activated according to event-condition and condition-condition coupling mode based on the sensing data, temporal event and rule base event.

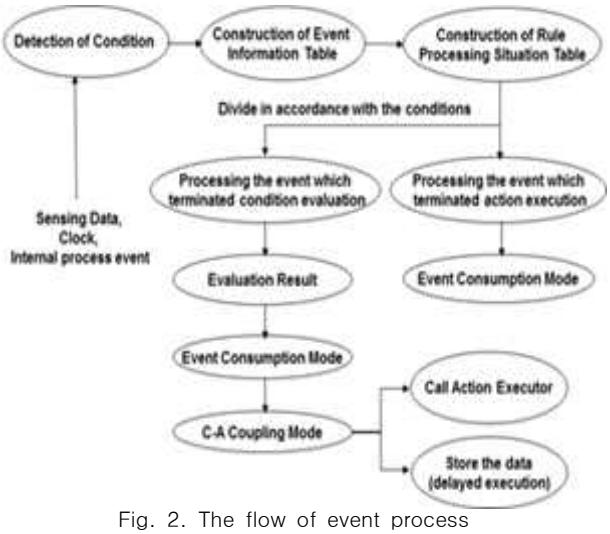

Fig. 2. The flow of event process

The rule manager proposed in this paper is related with another modules and functions for the overall processes of the active rules. The components of the active rule system including the proposed rule manager is shown in Fig.3.

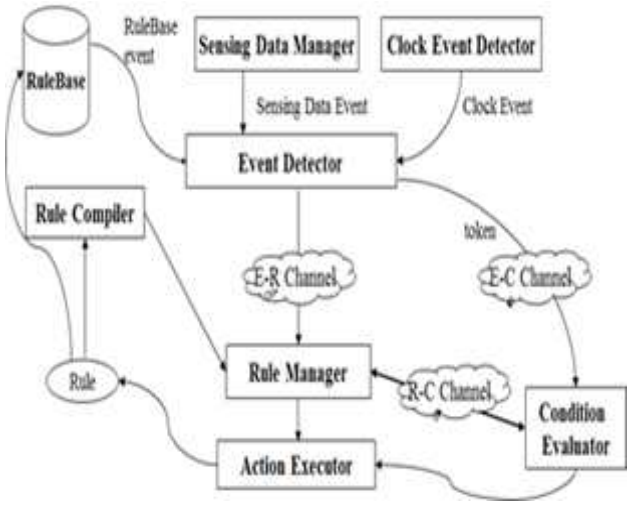

Fig. 3. Components of active rule system

IV. Rule Manager

As shown in Fig.3, the rule manager controls components of rule system such as event detector, condition evaluator, and action executor, and manages rule base. The rule managing procedure performs addition, deletion, and modification of a rule as well as enabling and disabling. Details on such rule managing and processing procedures handled by the rule manager are described as follows.

1. Roles of rule manager

Rule manager creates and maintains internal rule information table, rule processing situation table and event information table to manage internal data. The functions related to rule management are addition, deletion, modification, enabling, and disabling of rules.

- **Addition of rule information:** A new rule information is loaded on the internal structure of rule manager and it is notified to event detector and condition evaluator. The relevant rule information is added to rule information table by rule identifier. For the proper processing of each modules, the parts of event and condition of the rule are stored in the channels of event detector and condition evaluator, respectively, and it is notified to event detector and condition evaluator.

- **Deletion and modification of rule information:** Information about the deleted or modified rule is

eliminated or modified in the rule information table and it is notified to event detector and condition evaluator.

- **Enabling/Disabling rule**: A particular rule is set to respond or not to respond to an event, and it is also notified to event detector and condition evaluator.

The following Fig.4 represents fetch-oriented relationship between the rule manager and other modules.

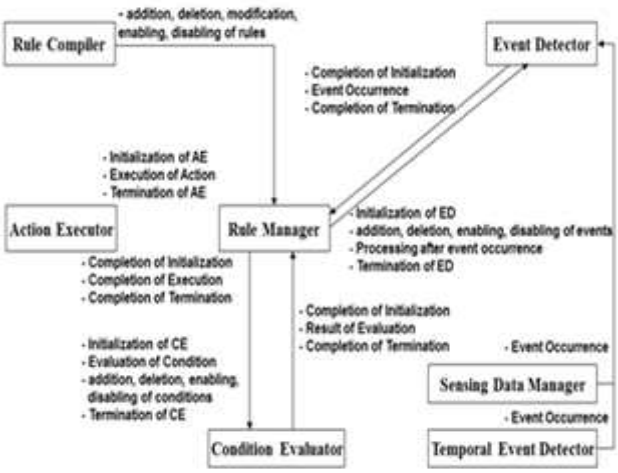

Fig. 4. Relationship between rule manager and other modules (fetch-oriented)

2. Procedure of rule processing

The procedures of rule processing involve initialization of event detector, condition evaluator, and action executor, and the process performs handling of internal structure and event occurrence. The control strategy of the procedure is as follows:

```
Rule_Execute()
{
Event Detection(sensing data, temporal event);
Store Event in EventInfoTable;
Identification/Store/Load(Agent) Triggered Rule;
repeat until no rules are triggered {
    Fetch Condition Evaluator according to E-C coupling mode;
    Processing to Event Consumption mode(after condition evaluation);
    Selection the Executing Rule by applying collision resolution strategy;
    Fetch Action Executor according to C-A coupling mode;
    Execution of Action;
    Processing to Event Consumption mode;
} }
```

- **Main module of rule processing:** This module performs initialization of rule manager and proper processing of each event that occurred. If an event is detected after initialization, the main module calls for functions that process each event.

- **Initialization for rule processing:** In order to initialize the rule processing module, this module creates processes of condition evaluator, event detector, and action executor, generates a channel (shared memory) for data exchange, and loads internal rule information table, etc. If initialization is successful, then returns TRUE. The initialization process is as follows:

```
int Initialize_RuleManager()
{
Construct_REChannel()
Create_EventDetectorProcess()
Construct_RCChannel()
Create_ConditionEvaluatorProcess()
Construct_RAChannel()
Create_ActionExecutorProcess()
Load_RuleInfoTable()
if (there are no errors) {
        return TRUE
}
        return FALSE
}
```

- **Database related event processing:** This module calls for the functions related to the event that can occur in the database (DB_EVENT). It stores evaluation information according to both collision resolution strategy of the event and event-condition coupling mode of the selected rule. Then, it sends signal for condition evaluation to the condition evaluator.

- **Processing for the completion of condition evaluation:** This module brings evaluation result from the condition evaluator upon receiving signal for the completion of condition evaluation, and handles processes according to event consumption mode. If the condition evaluation result is TRUE, it refers to the condition-action coupling mode of the rule. If the mode is IMMEDIATE, calls action executor, and if it is DELAYED, stores the data. The

procedure for the completion of condition evaluation is as follows:

```
Process_for_CONEVALUATED(rule R)
{
if (sig_num = SIGUSR1) {
Evaluation_Result = GetEvaluationResult (R)
if (consumption-time=CONSIDERATION) {
   if (consumption-scope = LOCAL)
      Delete_ACTIVE(E_id, R)
   else if (consumption-scope = GLOBAL) {
      Delete_EVENT(E_id);
      Delete_ACTIVE(E_id, R)
}}
if (Evaluation_Result = TRUE) {
   if (R.CA_Cmode = IMMEDIATE) {
      WriteToRAChannel(R_id); SignalToActionExecutor()
      }
   else if (R.CA_Cmode = DELAYED)
      Save_DelayedRule(R_id, ACTION_PHASE)
} } }
```

- **Processing of the event consumption mode after completion of action execution:** This module performs the process for event consumption in relation to the IMMEDIATE and DELAYED mode of action execution, and it can be changed once the event consumption mode supported by the system is determined. The procedure for the event consumption after the execution of action is as follows:

```
Process_for_ACTCOMPLETED()
{
if (signum = SIGUSR2) {
if (consumption-time = EXECUTION) {
   if (consumption-scope = LOCAL)
         Delete_ACTIVE(E_id, R)
   else if (consumption-scope = GLOBAL) {
         Delete_EVENT(E_id)
         Delete_ACTIVE(E_id, R)
} } } }
```

- Other modules: Modules capable of the following are required.

▪ Construct shared memory between event detector and rule manager, and obtain information about an event in the memory.

▪ Generate a collision set of rules triggered (Triggered_Rules) by the event, and select one rule after applying a resolution strategy for the collision set.

▪ Send signals for condition evaluation to the condition evaluator, and call condition evaluator with the token that triggered the event.

▪ Send signals for action execution to the action executor, execute the action, and if the coupling mode is DELAYED, classify the stages by event-condition or condition-action firstly, then store the tokens of the event.

V. Experiments of Rule Execution

Generally, in the sensor network, the mobile agent migrates to the sensor node and acquires current sensor data. If the agent has rules to execute on the sensor nodes, it evaluates the corresponding condition and executes the active rules. The mobile agent and active rule execution are based on the information of the *MetaTable* proposed in [7]. Then, it searches adjacent node, following the order in the migration list, migrates and executes the loaded active rule and returns to the sink node after completing the migration (Experimental result: Fig. 5). The mobile agent periodically collects data from sensors and sends it to sink node. Then the sensor data server loads the active rule on the mobile agent by the rule manager according to the situations (threshold, timeslot, presence of user, etc.). Next, it migrates the mobile agent to the destination node and performs the rule(adjustment priorities of migration, on/off or dimming control).

Fig. 5. Experimental results: Sequential migration following the order in migration list

The Fig. 6 shows that the migrated mobile agent acquires current sensor data, transmits it to the sink node and executes the loaded rule. The loaded active rule is that the mobile agent migrates to the highest priority node following the given priorities and eliminates the redundant sensing data at the corresponding sensor node using the *Redundant Data Elimination Algorithm*[9]. For the experiment, we defined the elimination of sensor data as the difference between the sensing data(*localData*) with two previous sensed data(pData, ppData) is less than the given threshold (*THRESHOLD=3*).

Fig. 6. Experimental results Optimal migration following the priority and rule execution (redundant data elimination)

In these manners, the active rule-based mobile agent operates by itself between the sensor nodes and sensor data server. When the mobile agent arrives at the sensor data server, the rule manager loads the appropriate active rule on mobile agent as necessary. The experiment was performed with Hmote2420 sensor nodes. The communication frequency band was 2405MHZ and the RF power was [Output Power 0 dBm, Current Consumption 17.4mA]. The operating System used was TinyOS-2.x and Cygwin tool was used for the development.

VI. Conclusion and Further Research

This paper is part of the research intended to induce power-saving by using active rule in controlling illumination without affecting the emotion of the users. It is suitable for the active rule system to derive smart power saving by adjusting the illuminance using active rules in accordance with the user's emotion. For this, we suggest the active rule-based mobile agent middleware system. To construct a practical environment for the active rule that is applied with user's emotional margin, we proposed a detailed design and implementation method of the role and function of the rule manager within the active rule system which utilizes metatable function of the existing Java RMI-based multi-agent system as well as migration and active rule-loading function of a mobile agent. The active rule system collects, sends, and processes sensor data through migration of a mobile agent when necessary. Through the operational experiments on migration, rule execution of a mobile agent and the active rule system resulting from an external event, we showed both potential and efficiency of further applications of the active rule system that uses a mobile agent within the sensor network environment. Continuous researches on the efficient linkage between the strategies for the design, execution, and management of an active rule that is practical for additional applications and mobile agent middleware system are required.

REFERENCES

[1] D. W. Lee, "Rule-Based Cooperation of Distributed EC Systems," International Journal of Contents, Vol. 5, No. 3, pp. 79-85, 2009.

[2] N. Kalanat, M. R. Kangavari, "Data Mining Methods for Rule Designing and Rule triggering in Active Database Systems," International Journal of Database Theory and Application, Vol. 8, No. 1, pp. 39-43, 2015.

[3] R. Trepos, et al., "Building Actions from Classification Rules," Knowledge and Information System Journal, Vol. 34, pp. 267-298, 2013.

[4] Konstantopoulos C., "Effective Determination of Mobile Agent Itineraries for Data Aggregation on Sensor Networks," IEEE Transactions on Knowledge and Data Engineering, Vol. 22, pp. 1679-1693, 2010.

[5] Heimfarth T., "Experimental Analysis of a Wireless Sensor Network Setup Strategy Provided by an Agent-oriented Middleware," 2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 820-826, 2010.

[6] D. Ballari, M. Wachowicz, and M. A. M. Callejo, "Metadata behind the Interoperability of Wireless Sensor Networks," Sensors, Vol. 9, No. 5, pp. 3635-3651, 2009.

[7] Y. S. Lee, M. S. Jang, S. K. Kang, "Active Rule System based on User's Emotional Margin for Power Saving Control," The Journal of The Institute of Internet Broadcasting, and Communication, Vol. 14, No. 3, pp. 119-124, 2014.
[8] P. patel, et al., "Context Aware Middleware Architecture for Wireless Sensor Network," 2009 IEEE International Conference on Services Computing, pp. 532-535, 2009.
[9] Y. S. Lee, J. H. Lee, "Forward Migration of an Active Rule Mobile Agent using the Meta_data," The Journal of KIICE, Vol. 1, No. 7, pp. 1567-1574, 2012.

Authors

Yon Sik Lee received the B.S. and M.S. degrees in Computer Science from Chonnam National University, Korea, in 1982 and 1984, respectively. And, his Ph.D. degree in Computer Application Engineering from Chonbuk National University, Korea, in 1994.
Dr. Lee joined the faculty of the School of Computer Information and Communication Engineering, Kunsan National University, Kunsan, Korea, in 1986. He is interested in sensor network middleware, active rule system, agent system and cloud computing.

Eun Hong Cheon received the B.S. degrees in Electronic Engineering from Kwangwoon University, Korea, in 1981. And, his M.S. and Ph.D. degrees in Electronic Engineering and Computer Engineering from Ajou University, Korea, in 1985 and 1998, respectively.
Dr. Cheon joined the faculty of the Department of Computer Engineering, Woosuk University, Wanju, Korea, in 1988. He is interested in wireless sensor network, computer and network security, and cloud computing.