Event Handling Mechanism based on Active Rule for Internet of Things

Biao Dong[1, a], Jinhui Chen[2, b]

[1]School of Computer & Software, Nanjing Institute of Industry Technology, Nanjing, 210023, China

[2] School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing, 210044, China

[a]email:dongb@niit.edu.cn, [b]email:cjh@nuist.edu.cn

Keywords: Event; Active Rule; Internet of Things

Abstract. This paper presents an approach, called IECA, for modeling and implementing the architecture of Internet of Things(IoT) applications using an active rule-based model. Considering the effective collaboration and the quick response which are necessary for IoT information sharing and service provisioning, a common and extensible IECA execution environment is constructed to integrate the advantages of publish/subscribe paradigm and event-driven architecture. A formal definition is proposed for the IECA. The results show that the IECA-based IoT applications can easily be designed and developed, while enabling an efficient management by the IECA execution engine.

Introduction

The IoT application tends to integrate a large number of distributed enterprise information systems, and forms a large-scale complex service system, a large amount of sensed information must be collected and shared across multiple heterogeneous information systems. On the basis of the fusion processing on a large number of distributed and multi-source information, an IoT application system needs to coordinate its relations with users, business processes and physical entities, especially needs rapid interactions cross business domain and even cross organization. IoT application has the characteristics of being dynamic and real-time not only because there are various types of users, business processes and physical entities, but also because the physical world is constantly changing. Therefore, these characteristics bring the following new technical challenge to IoT application: based on real-time sensing information, how to establish more effective collaboration between users, business processes, physical entities and a large number of heterogeneous services to quickly respond to dynamic changes of the physical world.

Event-Condition-Active(ECA) rules is a short-cut for referring to the structure of active rules in event driven architecture and active database systems. Such a rule traditionally consisted of three parts: The event part specifies the signal that triggers the invocation of the rule; The condition part is a logical test that, if satisfied or evaluates to true, causes the action to be carried out; The action part consists of updates or invocations on the local data[1]. In IoT application, independent services can rely on active rule mechanism to complete certain service collaboration. The active rule mechanism not only improves the real time response capacity of IoT application to quickly changing business requirements, but also significantly reduces the impact on the existing application systems.

Alam proposed an event driven service oriented architecture paradigm which assists an IoT virtualization framework to leverage the monitoring process by dynamically sensing and responding to different connected objects sensor events[2]. Zappia proposed a lightweight stage-based event processor based on a layered architectural design. Core event processing logic is decoupled from low-level thread management issues[3]. Perera surveyed context awareness from an IoT perspective, and analyzed that IoT should be facilitated by a hybrid architecture which comprises many different architectures[4]. Aman detailed an event driven adaptive security model for IoT to approach the objective specified and explicated how it can be utilized in an eHealth scenario to protect against a threat faced at runtime[5]. Patti presented the design of an event-driven user-centric middleware for

monitoring and managing energy consumption in public buildings and spaces. The middleware allows an easy integration of heterogeneous technologies[6].

The above researches are focused on solving the problem of interconnection and interoperability between different entities in dynamic interaction environment, but a key problem for further study and implementation is how to distribute and aggregate information effectively in IoT computing environment, and how to realize dynamic collaboration across business domains and across heterogeneous systems. In the paper, we proposes IoT-oriented ECA(IECA) which is an active mechanism modeling method for supporting effective collaboration and quick response in IoT application, and construct an IECA execution environment which provide an IECA-based services provisioning approach.

ECA Rule for IoT Application

IoT's data is generated by interaction between IoT device and environment object. IoT's data has the following characteristics: (1) Simplicity. IoT's data can be expressed as three tuple <id, value, timestamp>, which ID is a device identifier, value refers to observation value of different types, and timestamp refers to the time when IoT device acts on an environment object. (2) Dynamic. In the majority of IoT applications, IoT device quickly and automatically generates the observed values in the form of data flow, so that a huge amount of data flow is generated. (3) Correlation. The observed value of IoT's data is related to the observation time, and IoT's data are related to each other. The correlation of IoT's data is divided into two categories: temporal correlation and spatial correlation. (4) Semantic implication. The simplicity of data means that IoT's data can't contain a large amount of information. However, the accumulated massive IoT's data can provide implicit knowledge and semantic information. In IoT application, IoT event is defined as a meaningful change in the time and the place of the environment object bound by IoT device. After the combination of a massive IoT's data with their background and application information, IoT event abstracts IoT's data, and gives semantic information of IoT's data. From hierarchical structure point of view, IoT event is divided into original event and complex event. After simple data cleaning, IoT's data are transformed into original events with simple semantic information. Complex event is composed of original events; it has higher semantic information, and contains business logic.

Event driven architecture technology is used to solve the following problems: on-demand distribution of sensed information across business domains or even across organizations, and dynamic service collaboration. The integration of active rule and event driven architecture enhances the flexibility of system, which defines a methodology for designing and implementing applications and systems in which events transmit between decoupled systems and services. Through calling the functions of real-time filtering, aggregation and event correlation diagnosis, the integration model can extract meaningful events for business applications, makes real-time and dynamic response to events, and provides enterprises with the ability to respond to business applications dynamically. Therefore, this integration model is suitable for IoT applications with high degree of automation. IECA uses ECA rules to describe the IoT event process model. Also IECA does some modifications to the model to adapt for the requirements of event handling technology on IoT computing environment. IECA shall meet the following requirements. First, it can describe both IoT events and active rules. Second, it can describe the relationships between IoT events and IECA rules. Based on the above requirements, IECA defines two types of files, such as event file and rule file. Event file is used to represent the composition and state of event, and rule file is used to represent association rules between the different events.

IECA Execution Environment

The IECA execution environment is an event-driven rule policy system, and includes four layers: access agent layer, active rule engine layer, publish/subscribe layer, and business application layer. In the execution environment, the upper business application can be viewed as a service subscriber or a service provider; the underlying access agent can be regarded as information access. The

execution environment achieves the service binding properties by publish/subscribe mechanism. Fig.1 shows IECA execution environment based on ECA rule.

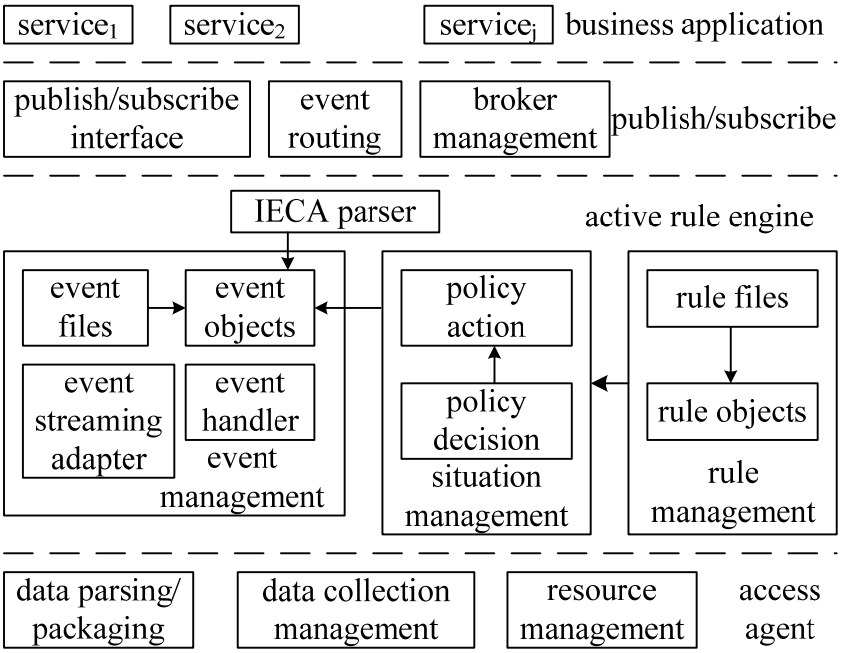

Fig.1. IECA execution environment based on ECA rule

The access agent layer includes three modules: data parsing/packaging, data collection management, and resource management. Data collection is a process of preprocessing and data collecting. The data collection management module provides an interface to help user configure collecting parameters. User can easily add or remove related collecting parameters in order to adapt to changes in application demand. The resource management module mainly implements data transformation. After receiving data from sensor, the access agent layer transforms heterogeneous raw sensor data into unified observation information according to resource description, and then generates original events.

The active rule engine layer includes four components: event management, situation management, rule management, and IECA parser. Firstly, the event management component treats data as different types of events. By analyzing the relationship between events, the events will be combined into more complex event with high-level semantics. Therefore, event becomes a signal that triggers the execution of the entire business process execution. In all, the component implements a framework to integrate the complex event processing engine, which is divided into four main modules: event files, event objects, event streaming adapter and event handler. Secondly, when events occur, the event management component transmits the events to the situation management component to make a decision. The situation management component is divided into two modules: policy action and policy decision. The policy decision module is responsible for parsing incoming events, and matches the events with the policies in rule files. If the match is successful, the corresponding rule is activated. The policy action module is responsible for executing the actions of matching rules to change the status of the target event objects. Thirdly, the rule management component is responsible for storing the rule details, including identification, event, condition and action. Rule files are parsed for structured storages. Finally, IECA parser analyzes event files, and extracts event objects.

The publish/subscribe layer includes three modules: publish/subscribe interface, event routing, and broker management. The layer is responsible for publishing, subscription and routing of event. Events which are published in a unified message space can be subscribed by other related IoT applications. Control commands can be used as events by IoT applications, and can be sent to some relevant devices and sensors. Similarly, IoT applications can also publish/subscribe to business events in application layer, that is, the events generated by IoT applications, in order to achieve

service collaboration between different applications.

Formal Definitions of IECA Execution Environment

IECA execution environment is composed of several services which cooperate with each other to complete the corresponding the whole function of system. These services provide external interfaces which are called application programming interfaces(APIs).

Definition 1 Event. An event is a five-tuple: E=(event-name, I-in, I-out, B-in, B-out), where event-name is the name of an API, and is a unique identifier for each API. Let I-in={p_1, p_2, …, p_m}(m≥0) and I-out={p_1, p_2,…, p_n}(n≥0) denote input interface and output interface of the API respectively, we use the elements to define the input parameters and the output parameters. Let B-in and B-out denote the event sets for the input interface I-in and the output interface I-out, respectively.

Definition 2 IECA rule. An IECA rule is a four-tuple: R=(event-name, condition, active-APIs, active-parameters), where event-name is the name of an event, and is a unique identifier for each event. Condition is the precondition for firing rule, which is a boolean expression. If the value of the expression is true, the rule is triggered. When the rule is triggered, active-APIs is the set of APIs that are executed, active-APIs can be null. Parameter represents the real parameters when calling the APIs.

In all active-APIs, due to the division of functions, there are some key active-APIs that act as primary entry points to main functionality. According to different input parameters, these active-APIs enter different processes. We define a set of parallel active-APIs as an active group.

Definition 3 Active group. An active group is a three-tuple: AG=(rule-name, active-APIs, t). AG is a set of active-APIs that have been called at time t, and the number of APIs in the set can be 0, 1 or more.

Definition 4 Active group tree. An active group tree is a four-tuple: AGT=(AGS, ag_0, E, $C_{W,E}$), where AGS is the set of active groups, and an active group ag∈AGS is called a node. The parameter ag_0∈AGS is an initial activity group for IoT business application. E is a set of events that trigger changes in active groups. Let $C_{W,E} \subseteq AGS \times AGS \times E$ denote the transition relationship between active groups. If there is $(c_i, c_j, e) \in C_{W,E}$, this means there is the father-son relationship between the active group c_i and c_j; active group c_i and active group c_j are parent node and child node, respectively; the parameter E is an event that triggers the transition from c_i to c_j.

In an IoT application system, a finite sequence of events is called an event sequence. The process of performing activities in rule can be regarded as an ordered composition of active groups. One active group is converted to another by one or more events at run time. In the transitions of active groups, IoT business applications can make different API calls according to different parameters. So IECA execution environment requires only a few simple elements, such as IECA rule sets and active groups, but can solve the transition problem of different active groups.

Conclusion

In this paper, we propose an IoT-oriented approach for architecting IoT applications using active rule mechanism. We derive conclusions of IECA model from analysis, and construct an IECA execution environment from four layers. The results show that the approach effectively realizes service collaboration and the quick response for IoT applications.

Acknowledgement

This work was sponsored by Qing Lan project(Jiangsu province, china).

References

[1] H.Weigand, A.Paschke. The pragmatic web: Putting rules in context[M]. Rules on the Web:

Research and Applications, Springer Berlin Heidelberg, 2012, pp.182-192.

[2] S.Alam, M.M.R.Chowdhury, J.Noll. Senaas: An event-driven sensor virtualization approach for internet of things cloud[C]. NESEA 2010, pp.1-6.

[3] I.Zappia, F.Paganelli, D.Parlanti. A lightweight and extensible Complex Event Processing system for sense and respond applications[J]. Expert Systems with Applications, 39(12), 2012, pp.10408-10419.

[4] C.Perera, A.Zaslavsky, P.Christen, et al. Context aware computing for the internet of things: A survey[J]. Communications Surveys & Tutorials, IEEE, 16(1), 2014, pp.414-454.

[5] W.Aman, E.Snekkenes. Event driven adaptive security in internet of things[C]. UBICOMM 2014, 2428:715.

[6] E.Patti, A.Acquaviva, M.Jahn, et al. Event-driven user-centric middleware for energy-efficient buildings and public spaces[J]. Systems Journal, IEEE, 99, 2014, pp.1-10.