Secure Cooperation in a Distributed Robot System using Active RFIDs

Makoto Obayashi
Dept. of Product Development
Tokyo Metropolitan Industrial Technology
Research Institute
115-8586, Nishigaoka 3-13-10,
Kitaku, Tokyo, Japan.

Hiroyuki Nishiyama and Fumio Mizoguchi
Information Media Center
Tokyo University of Science
270-1163, Yamazaki 2669-1,
Noda, Chiba, Japan.

Abstract

In this paper, we developed a distributed robot system that can provide various services in a real environment using ad-hoc networked active radio frequency identifications (RFIDs). These services are derived from a large amount of data from sensors connected to active RFIDs. The primary advantage of this method is that it allows the construction of a real environment monitoring system easily. Moreover, human status and position are easily identified by outfitting active RFIDs to subjects. However, a security system is required for a radio ad-hoc network and cooperation system between active RFIDs. In our research, we developed a multi-robot cooperating system as a multi-agent system and applied it to an active RFID, which has limited resources. We also developed a security system for active RFID communication that can be executed using limited resources. We then integrated the multi-agent system and security system. We also constructed a robotic environment that can provide various services using active RFIDs and then evaluated it.

1 Introduction

In recent years, the significance of developing a ubiquitous computing environment by establishing information terminals in various places has been recognized. This purpose aims to construct a highly developed information society[4] and build a barrier-free society for the elderly and the physically handicapped. Autonomous control of various devices in the real environment enables access to information and various services[2]. For these purposes, active Radio Frequency Identification (RFID) is both important and convenient recognized. Unlike a passive RFID represented by an IC tag, an active RFID has a CPU and a battery and can operate autonomously. It becomes possible to easily build a sensor network by developing the ad-hoc interconnection between active multi-sensor RFIDs. Generally, each active RFID is small and inexpensive. Therefore, it minimizes the risk when applied to outdoor or hazardous areas. Also, since it can attach to a person without incongruity, it is possible to collect varied information using sensor technology and to realize various services. It is also easy to make a home or office adapt to a robotic environment by arranging active RFIDs equipped with actuators (Fig.2). However, there are several problems in operating active RFIDs. First, all communications are carried out by an ad-hoc radio network, and packets are intercepted by neighboring active RFIDs. Second, in order to achieve low power consumption and downsizing, RFIDs have limited computer resources and transmission speed. For this reason, it is impossible to implement the usual security techniques. When developing a robotic environment using active RFIDs, anticrime and disaster prevention must be considered. While communicating is crucial, malicious hackers must be intercepted. In order to build a robot environment, dynamic cooperation between active RFIDs is necessary. Examples of this cooperation include invader authentication and the control of information appliances according to sensors in the real environment. In this case, processing of Cooperation and security are interrelated, making it difficult to deal with these problems. Therefore, we developed a secure dynamic active RFID (Fig.1) cooperation system suitable to limited computer resources. Our system incorporates a multi-robot cooperation system produced in our previous work[3] into an active RFID ad-hoc network system. We also implemented the framework for cryptography and authentication to facilitate dynamic cooperation between active RFIDs. Finally, we discuss the applicability of our system based on the experiments.

2 TinyMRL

We developed the multi-agent language "TinyMRL" to solve the problems described in

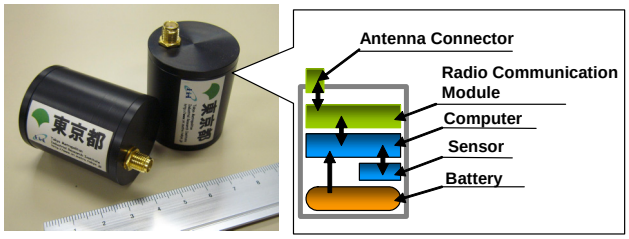

Figure 1: This figure shows active RFIDs in our research: It is small computer which has battery and ad-hoc radio network module. It can get various information using sensors and control actuators.

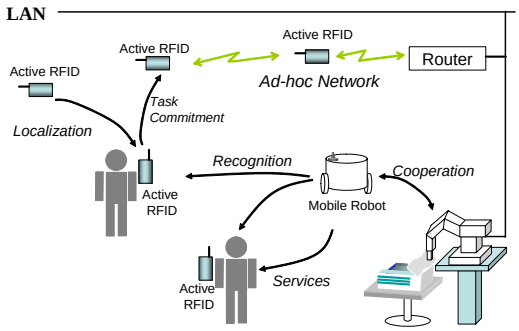

Figure 2: Our model of Communication between devices. Our system detects the position and status of persons by sensors equipped to active RFID.And it realizes various services by cooperating with devices in the environment via ad-hoc network.

the previous section. TinyMRL enables dynamic secure cooperation between active RFIDs. We achieved dynamic cooperation and competitive resolution between robots by developing a multi-agent system "MRL"[3]. When heterogeneous robots cooperate in a real environment, efficient task assignment and competition resolution are required. These are indispensable when multiple robots cooperate in executing a task. MRL allows robot behavior to be defined as a set of rules, where an action is determined by its internal state and environment alteration. Moreover, functions of shared variables and streams enable the simple description of competition resolution between robots. TinyMRL, which we developed in this research, is a multi-agent language and its execution environment. TinyMRL is applied MRL, which is then applied to active RFIDs. TinyMRL is executable on embedded computers, even those with limited resources (including 8-bit CPUs).

Although TinyMRL has a small system, it isn't

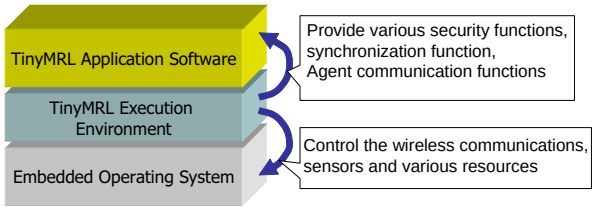

Figure 3: TinyMRL System Configuration

a subset of MRL. MRL is a processing system based on the assumption of parallel computing. It is designed based on a concurrent logic language, which enables it to efficiently unify functions and synchronization between parallel processes. It is confirmed by heterogeneous robots executing cooperative tasks[3]. However, the concurrent logic language has several shortcomings. First, it needs a parallel computer and a UNIX OS. This is not necessary for a large-scale robot system, but it is redundant or superfluous equipment to a sensor network and an information appliance network. Second, concurrent logic language is more difficult than procedural language. Operation of an undefined variable and stream make debugging difficult. Third, MRL does not allow some branch and loop instructions ("'if," "for," or "while"), therefore implementation of control calculations is difficult. TinyMRL is implemented using the advantages of MRL to the embedded computer, and is also designed for devices that have a network communication module. Moreover, it functions as a script language for rapid behavior setup of active RFID. Also, TinyMRL can describe an agent state transition as a set of defining rules that inherits the features of MRL. Moreover, in TinyMRL, "for", "while", and "if" sentences can be written in a description of a predicate. These are necessary for implementing data processing. Furthermore, TinyMRL execution environments provide a security system as the system call, so the developer can communicate securely. Figure 3 illustrates the TinyMRL system configuration. The bottom block is an embedded OS, such as μITron. The top block is the agent software, which is defined by TinyMRL syntax. The middle block is the TinyMRL execution environment that interfaces with the OS, ecurity functions, and the cooperation framework between active RFIDs.

3 Implementation

We adopted MOTE developed by the Crossbow Corporation as the active RFID. It has an 8-bit CPU, 128Kbyte ROM and a radio network module that can constitute an ad-hoc network.

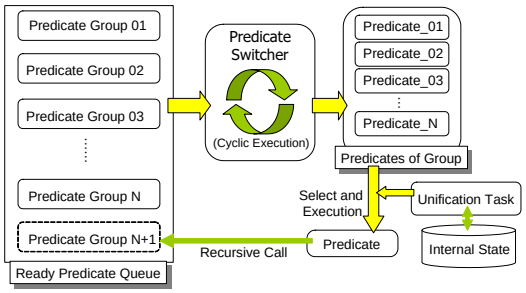

Figure 4: Processing flow of selecting and switching predicates.

3.1 Agent definition using TinyMRL

In TinyMRL syntax, behavior of an active RFID is controlled by a set of predicates. Each predicate consists of a predicate name, condition part, and execution part (the same as a Guarded Horn Clause GHC)). Figure 4 illustrates the TinyMRL execution process diagram. A predicate is classified by its number of arguments. Namely, predicates with the same name and same number of arguments are classified in the same group, and predicates are launched from each predicate group. When a predicate group is launched, only the predicate whose internal state of a definition formula and active RFID in a condition paragraph is called. If several predicates satisfy the launch condition, only one predicate is called according to its priority. In general, a predicate priority is determined by the number of conditions. When two or more predicates have the highest priority, one predicate is chosen at random. Finally, in the predicate execution, a series of processes can be constituted by calling a predicate recursively at the end of the execution paragraph. Hence, TinyMRL syntax easily describes the state transition of agent behavior. A predicate that belongs to the same predicate group expresses a state, and a recursive call means continuing the state. Furthermore, when an agent transitions to another state, it is realized by the predicate definition that corresponds to arbitrary events and calls the other predicate group. The end of a state is also realized by not calling any other predicate. In this case, predicates are not queued into the ready predicate queue, resulting in a simultaneous completion of the series of state transition operations. Figure 5 shows an example of a TinyMRL program.

3.2 Security function of TinyMRL execution environment

This section describes the details of TinyMRL security functions. Generally, an active RFID has poorer

```
00:start() :- true {     /* Start execution */
01:  run(1);             /* Launch a predicate */
02:  run(1,2);           /* Launch a predicate */
03:}
    /* When the variable a is equal to 1,
       followng predicate is executed */
04:run(int a) :- a == 1 {
05:  syscall:dataRequest(Turtle);
06:  run(2);               /* Recursive Call */
07:}
    /* When a message is received,
       following predicate is executed */
08:run(int a) :- a==2 && syscall:receiveMsg(){
09:  Massage msg = syscall:getMsg();
10:  run(msg);           /* Recursive Call */
11:}
```

Figure 5: TinyMRL Example Code

resources than a common computer. It also has low throughput corresponding to low power consumption. The radio signal is therefore weak. This restricts the speed and range of radio communication. μTESLA and SNEP provide techniques for achieving network security with limited resources[1]. We developed the system by fusing these security techniques to TinyMRL. The TinyMRL execution environment has a secure communication framework between active RFIDs and allows developers to construct dynamic secure cooperation without special operations. TinyMRL provides cryptography, packet authentication and semantic security functions. In our system, we assume that all active RFIDs a developer employs are reliable. We also assume that malicious hacking attempts are elicited from participating in the ad-hoc network by the cracker's active RFID. Also, all active RFIDs a developer initializes have a common secret key. Keys used by data encryption, decryption, and authentication are generated from a secret key, which all reliable active RFIDs possess. When $Agent_a$ tries to communicate with $Agent_b$, $Agent_a$ has to launch the following system call. Its procedure is then changed to the TinyMRL execution environment.

$$\texttt{system:sendMsg(}Agent_b\texttt{, <Message>)} \quad (1)$$

In this case, the communication content processed by TinyMRL between $Agent_a$ and $Agent_b$ is expressed as follows.

$$Agent_a \rightarrow Agent_b : N_a, \texttt{<Message>} \quad (2)$$

$$Agent_b \rightarrow Agent_a : \{\texttt{<Message>}\}_{<K_{ba}>}, \texttt{MAC}(K_{ba}, N_a || E_b) \quad (3)$$

$$E_b = \{\texttt{<Message>}\}_{<K_{ba}>} \quad (4)$$

Table 1: Time required to commitment in contract net protocol

The number of predicates	10	20	30	40
Time required to commition [msec]	2988	3092	3156	3733

N_a in equations 1 and 2 means the nonce is generated by the active RFID defined as $Agent_a$. The TinyMRL execution environment generates a nonce at every communication instance between agents. Nonce is a randomly generated bit array. It is used to maintain the order and novelty of packets. In our system, a 64-bit-long nonce is implemented. When a nonce is generated according to the system call of communication, the TinyMRL execution environment stores the nonce in internal memory. After this procedure, it guarantees the packet order and launches the agent communication process by checking whether the received packet includes its own generated nonce. In this implementation, we use the RC5 algorithm developed by RSA for cryptography. We use the OFB mode in the RC5 algorithm processing. Therefore, one function achieves encryption and decryption. We use the CBCMAC and the RC5 algorithm for generating a Message Authentication Code (MAC). A MAC is generated as the final block of the CBC mode. In our system, a MAC is generated using the received packet nonce and encrypted message, shown as Equation (3).

4 Experiment

We examined the performance of a contract net protocol using several active RFIDs to evaluate our system. A contract net protocol is used as the typical negotiation technique for task assignment. Figure 6 shows agent task behavior when executing a contract net protocol. We use four active RFIDs, and NODE1 serves as the client. The other three NODEs bid for requests from NODE1. USER is a person's process of information terminal. The processing of the contract net protocol begins when indicated by USER. In this experiment, we measured the change of average time required to negotiation by increasing predicate in each active RFID. The experimental results are shown in Table 1. As the shown in the table, even the result of the least predicate number takes about 3 seconds. Therefore, the experiment result shows that our system can't be applied to reactive system. But, our system is available to secure cooperating system of active RFIDs.

5 Conclusion

In this research, we developed a multi-agent language and its execution environment to resolve cooperation problems and negotiation for task assignment of a distributed robot system using active RFIDs. Moreover, we developed the security system to guarantee the secure communication between active RFIDs, which have limited resources. We also developed an active RFID behavior definition system that allows the developer to easily construct an ad-hoc network robot system by fusing these functions. We then evaluated the applicability of our system through experimentation.

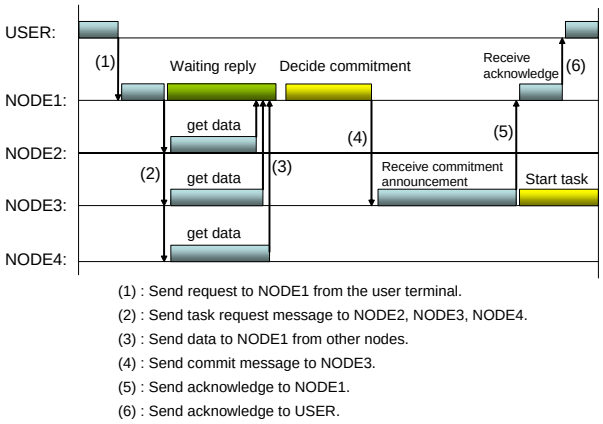

Figure 6: Time line of the execution of contract net protocol between active RFIDs.

References

[1] Adrian Perrig , Robert Szewczyk , Victor Wen , David Culler , J. D. Tygar, "SPINS: security protocols for sensor netowrks," Proceedings of the 7th annual international conference on Mobile computing and networking, pp.189-199, July 2001, Rome, Italy.

[2] F.Mizoguchi, H.Nishiyama, H.Ohwada and H.Hiraishi, "Smart office robot collaboration based of a multi-agent programming," Artificial Intelligence, Vol.114, 1999.

[3] Hiroyuki Nishiyama, Makoto Obayashi, Hayato Ohwada, Fuimo Mizoguchi, "A Concurrent Logic Programming Language for Multiple Robot Cooperation," Journal of the Robotics Society of Japan, vol.19, No.5, pp.620-631, 2001.

[4] Ora Lassila and Mark Adler: "Semantic Gadgets: Ubiquitous Computing Meets the Semantic Web," in: Dieter Fensel et al (eds.): Spinning the Semantic Web, pp.363-376, MIT Press, 2003