A Novel Dynamic Service Architecture for RFID and WSNs Applications

Tao Ku, YunLong Zhu, KunYuan Hu, HanNing Chen
1 Shenyang Institute of Automation of CAS, Shenyang, 110016
2 Graduate school of the Chinese Academy of Sciences, Beijing, 100039
{kutao, ylzhu, hukunyuan}@sia.cn

Abstract - A novel dynamic service architecture used to integrate Radio-Frequency Identification (RFID) and Wireless Sensor Networks (WSNs) is proposed according to requirements on the ubiquitous computing. An infrastructure used to monitor devices and collect data at real time is proposed, which constitutes with monitor service and container service. The monitor service provides the mechanism based on service meta-model for dynamic deploying and fault-tolerance. The container service provides lifecycle management and runtime environment for sensors application services. An algorithm used to schedule and deploy application services dynamically is proposed, which is based on QoS capabilities provided by compute resource, such as CPU speed, memory capability and bandwidth. The results show that the service schedule approach is remarkable.

Index Terms - RFID, WSNs, Dynamic Service

I. INTRODUCTION

Over the past few years, a great deal of attention has been driven towards the pervasive computing for integrating RFID and WSNs [1]. Advances in the field of RFID and WSNs have brought tremendous benefits to bridge the gap between the physical and the virtual world. RFID systems have been applied in a number of applications, such as asset tracking, and supply chain management [5]. Meanwhile, WSNs offer a number of advantages over traditional RFID implementations, such as, the ability to rapidly deploy a multi-hop network, etc. However, this progress has also raised important problems, such as, the limited energy of sensor nodes, self-organizing network changes in the network topology, etc [4]. One of the most important aspects of integration is how to design middleware architecture for satisfaction requirements both RFID and WSNs. However, RFID and WSNs are almost under development separately. Few integration schemes and its related opportunities are investigated in detail.

There exist many challenges of middleware architecture for integrating RFID and WSNs, which are derived from the objective of supporting the development, maintenance, deployment and implementation based on the perception of applications. Another issue that needs to be considered is that the nodes of complex perceptual tasks mechanism, secure communication, and perceptual nodes synergies between multi-sensor nodes. Currently, there are already some projects underway that aim to develop middleware for RFID and WSNs, such as [7, 8, 9, 10]. In [11] a novel middleware approach for Pervasive Grid Environments is introduced, in [12] a Context-Aware middleware System for Mobile Applications is introduced. Especially, in [13] present policy-Driven methods for context-aware dynamic adaptation Environment. However, most of the projects are focusing on developing algorithms and components. Moreover, most of the current results are based on simulations or small-scale experiments.

The goal of this paper is to propose novel middleware architecture for integrating seamlessly RFID and WSNs. Firstly, we classify services as infrastructure service and application service based on Jini [2,3] service specification. Infrastructure service includes container service as executable environment, monitor service as management station and lookup service. RFID and WSNs are encapsulated the sensor services as application service beans, which can be executed on container service and monitored by monitor service. The service execute agent enable us to use adaptation-specific code of RFID and WSNs devices and compose services for directly executing under dynamic distributed environments. A dynamic service schedule algorithm is performance to cooperate services among limited computing resources. The algorithm is not only to support static topological structure but also support the specification of dynamic cooperation schedule; the results are shown that the advantages of the proposed schedule approach.

II. DYNAMIC SERVICE INTEGRATION INFRASTRUCTURE

The Rio [6] project based on Jini provided a dynamic adaptive architecture using for developing, deploying and managing distributed applications. The keys to the architecture are a set of dynamic capabilities, and reliance on policy-based and Quality of Service mechanisms. Rio reduces the complexity surrounding the development of dynamic services by a simple service specification model.

A. Integration Architecture

In our proposed integration architecture is based on Rio architecture, shown as Figure 1. The architecture includes six main components, viz. service discovery, service assembly, service configurations, service monitor, container and RFID/Sensor service execute agent. In the following, we will discuss the above components in details.
Service Discovery is responsible for discovering a lookup service using unicast or multicast protocols and match RFID and WSNs service requests with service template in lookup service entity database. The client prepares a Service Template, which is a list of class objects and a list of entries, to see if the lookup service has a service matching the template. If the match is successful, an object is returned that can be cast into the class required. The output of this component is a service request for RFID or WSNs services.

Service Assembly is used by service container to register its availability to support Jini service beans and receive requests of service provision to instantiate a Jini service graph of constituent services listed in Lookup Service. In general, a service request is expanded into constituent services until no more services need to be expanded. Services are expanded based on the inputs they require to run.

Service Configure will configure services as operational string in Monitor Service, which provides service synergies functionalities while satisfying certain constraints, for example on the number and locations of the nodes. Service configure requires frequent topology change information of the services provided by nodes as well as their properties, such as location, remaining power, etc. The output of Service configure consists of service configuration messages which are sent to cybernode for composing the service dependable relationship.

RFID/sensor Service Execute Agent is the infrastructure services that are part of the Rio architecture. Rio services are also Jini services, and they are manageable objects that providing administrative user interfaces and logging and tracking of activity, these objects can be used for event tracking, process diagnostics or basic historical analysis. They provide essential support to enable some of the capabilities for integration RFID and WSNs in the Rio architecture. Such as, perceptual nodes synergies between nodes, RFID&Sensor data process, secure communication, and complex perceptual tasks mechanism, etc. Each of these capabilities is implemented by the services described in Reader or Sensor service. Of course any other Jini service can be discovered and used as well. Developers are not constrained to the services available through the Rio architecture.

Monitor Service is responsible 1) configuring the node services based on an operational String, 2) deploying service to cybernodes based on quality of service (QoS) capabilities, and 3) managing runtime service and resource information required by failure recovery mechanism. The Provision service provides the capability to deploy and monitor Operational Strings. Monitoring an Operational String allows Service Provision component to detect and recover from service failure on the network. An operational string is a collection of service elements that together constitute a complex service in a distribute system. To manage an operational string, a service monitor determines whether a service instance corresponding to each service element in the operational string is running on the network and monitors the service instance corresponding to each service element in the operational string to ensure that the complex service represented by the operational string is provided correctly.

Container Service provides a lightweight container that turns heterogeneous compute resources into available services through the network, which includes service instantiate and service bean instantiator. Cybernode may also include one or more JSBs. the container represents the capabilities of compute resource they run on through quantitative and qualitative mechanisms. Container instances dynamically
discover and enlist with the Rio provisioning services, and provide a lightweight container to instantiate mobile services. The relationship between monitor and container may be one-to-one or one-to-many.

III. FORMAL DESCRIPTION

In this section, we will discuss some preliminaries needed to evaluate proposed integration architecture before presenting dynamic services schedule algorithm. The essential problem of dynamic services schedule is to schedule active entities (sensor service) to execute in a set of passive entities (Cybernode) under satisfying QoS requirement. A set of sensor service denoted by \( S = \langle S_1, S_2, \ldots, S_j \rangle \). Each service is composed of tasks and operations. A set of cybernode is a set of virtual computing resources denoted by \( V = \langle V_1, V_2, \ldots, V_n \rangle \). The term quality of service (QoS) may mean different for different types of resource. In our study, a three dimension QoS is considered denoted by QoS =\(<\)cpu, memo, disk\(>)\. We first give related definition of components and their properties:

**Definition 1.** \( E = \sum_{i=1}^{j} E_i \) Present the accumulated computation workload in terms of the expected time to complete the execution of those service tasks on \( V_i \)

**Definition 2.** \( O = \sum_{i=1}^{j} O_i \) Present the accumulated I/O workload in terms of the expected time to complete service transfers for those tasks on \( V_i \)

**Definition 3.** \( W_i = E_i + O_i \) Present the total cumulative workload on \( V_i \).

According to above definition, we assume \( j \) services in \( n \) cybernodes, \( W \) can be calculated by following equations:

\[
W = \sum_{i=1}^{j} \left( e_i^q - \left( t^q - t^q_i \right) \right) + \sum_{i=1}^{j} \left( o_i^q - \left( t^q - t^q_i \right) \right)
\]

\( e_i^q \) The computation cost of \( q \) in term of execution time on \( V_i \), \( q \) is operation set of service

\( o_i^q \) The I/O cost of \( q \) in terms of the service transferring time on network when it is assigned on \( V_i \)

\( t^q \) The current time point when the service scheduled

\( t^q_i \) The time point for service operation \( q \) to end its executes

\( t^q_e \) The time point for service operation \( q \) to end its network transfer

IV. ALGORITHM FRAMEWORK

The services are schedulable if the system completes the allocation of all services to cybernode resources within deadline and budget. The schedulable sequence of service corresponds to QoS route table. The algorithm for scheduling services in dynamic environment is follows:

**Algorithm**: dynamic service schedule.

**Input:**
- \( S \), a application services set
- \( V \), a service container set.
- \( QoS \), a quality of service set.

**Output**
- A service pair set. \( Pair < S_i, V_j > \)

**Method**:

\( S_i \leftarrow \) Single-elements in the application services set

\( V_j \leftarrow \) Single-elements in the service containers set

Call Dynamic-schedule \( ( S_i, V_j, QoS ) \)

**Procedure** Dynamic-schedule \( ( S_i, V_j, QoS ) \)

Begin

(1) \( S_i \leftarrow \Phi \) Read from application set

(2) \( V_j \leftarrow \Phi \) Read from container set

(3) for each application service \( S_i \in S \) do

(4) for each container \( V_j \in V \) do

(5) Begin

(6) if \( \text{Lookup}(s_i) = \text{true} \)

(7) then \( S = S - S_i \)

(8) \( \text{Monitor}(V(S_i))// \text{monitor container of } s_i \)

(9) if \( QoS_{S_i} > QoS_{V(S_j)} \) Then \( S = S + S_i \)

// reassign \( S \) according QoS ability

else \( V_i = \text{Schedule}(S_i, V) \)

(11) // schedule service to containers according QoS ability.

(12) \( \text{Update}(S_i, V_i) // \text{update container QoS ability} \)

(13) if \( S \neq \Phi \) then

(14) \( \text{Dynamic-schedule} \ (S_i+1, V_j+1, QoS) \)

(15) End

(16) return;

**Figure.2. Dynamic Services Schedule Algorithm**

A. Algorithm Performance Analysis

The algorithm tests were made on the three PCs with 1.4GHz Pentium IV processor, running Window 2003 as the operating system with 1 GB of RAM. In our experiment, the prototype is implemented in Java language based on jini specification, there are the eight services were defined constructing dynamic service architecture. Especially, the algorithm is performed to figure out the container with the least load from the available resource list, however, RFID reader service and WSNs service are rather intensive for input and output. The algorithm used scalability policy handler to increase service instances to run since many sensors produced potentially infinite service requests. Schedule policy is an automated mechanism that measures actual QoS values and increases/decreases instances in proportion to the SLA requests. Because the potential load increase differentiates
greatly for different machines due to their various disk cache states, therefore, it must take into count the currently existing workload on the machine and potential workload resulting from the tasks assignment, Figure3 is shown that low CPU, memory utilization and disk usage in our algorithm when services are scheduled among containers.

![Broken line graph representation of times in terms of QoS observed at schedule service](image)

Figure.3. Broken line graph representation of times in terms of QoS observed at schedule service

In figure 3 presents the processing time of service requests, memory utilization and disk usage. The horizontal axis corresponds to the quantity of service requested, and the vertical axis corresponds to the amount of time (in milliseconds). In general, the results are shown that low CPU, memory utilization and disk usage.

**B Implementation**

Currently, services encapsulation components are developed to support multi-protocol sensors hardware, and middleware system may be implemented with both object-oriented and non-object-oriented programming systems. Our target integration architecture represented as a collection of service elements running on the Rio infrastructure. Service elements include both infrastructure services and application services. The infrastructure services are responsible to service lifecycle management and sensor nodes management. The application services provide sensors core services, such as, RFID and WSNs data processing, events processing et.al. Moreover, we will improve the performance of dynamic services schedule algorithm by using heuristic method.

**V. CONCLUSION**

In this paper, a novel dynamic service architecture for integrating RFID and WSNs is discussed in detail. An algorithm of dynamic services schedule is proposed. The performance evaluation on multi-nodes is shown that proposed architecture has the capabilities to cooperate with RFID and WSNs under dynamic environments. It can be concluded that dynamic service architecture based on jini can bring more sophisticated application functionality to satisfy the requirement of pervasive computing, therefore, we believe that proposed architecture can provide a robust infrastructure layer to simplify RFID and WSNs application deployment and scaling.

**ACKNOWLEDGMENT**

This work is supported by the National 863 key project of the Advanced Technology development of RFID based Information integration platform under contract No 20060104A1118. National Science and Technology Key Project of China under contract No. (2006BAH02A09).

**REFERENCES**


