RFID에 기반한 유비쿼터스 환경에서의 어플리케이션 프레임웍 구조*

김기현°, 이정태, 류기열
아주대학교 정보통신전문대학 정보통신공학과
{misozigi°, jungtae, kryu}@ajou.ac.kr

Application Framework Architecture In ubiquitous environment based on RFID

Kee-Hyun Kim°, Jung-Tae Lee, Ki-Yeol Ryu

Dept. of Information and Communication Engineering, Ajou University

요 약

유비쿼터스 컴퓨팅 환경에서 공통적으로 필요로 하는 요구는 사물의 이동을 자동으로 인식하고 위치를 파악하는 것이고 이를 위해서 현재 RFID를 통한 사물의 인식이 각광을 받고 있다. 하지만 RFID에 기반한 어플리케이션은 일반 어플리케이션과는 달리 태그 인식에 의한 실시간 이벤트의 처리가 가장 중요하며, 이로 인해서 RFID 어플리케이션에는 실시간 이벤트 모니터링과 실시간 이벤트 처리, 비동기 이벤트 처리, 멀티 쓰레딩, 분산 처리 등이 반드시 필요하다. 이러한 RFID 어플리케이션의 특성 때문에 RFID 어플리케이션은 일반 어플리케이션과는 다른 구조가 요구된다. 이에 본 논문에서는 RFID 어플리케이션에서 공통적으로 요구되는 컴포넌트들을 추출하고 이를 효과적으로 결합하기 위한 프레임웍을 설계, 구현하였다.

1. 서 론

유비쿼터스 컴퓨팅 환경은 언제 어디서나 사용자가 자신에게 필요한 컴퓨팅 서비스를 받을 수 있는 환경을 의미한다 이러한 유비쿼터스 컴퓨팅 환경이 실현되기 위해서는 기본적으로 사용자 및 사용자의 현재 환경을 컴퓨터가 인식할 수 있어야 한다. 최근 이러한 연구가 많이 진행되고 있으며, RFID를 통한 사용자의 인식이 각광을 받고 있다. RFID(Radio Frequency IDentification)는 초소형 반도체에 식별 정보를 넣고 무선 주파수를 이용해 상품이나 동물, 사람 등을 판독, 추적, 관리할 수 있는 기술이다. 하지만 RFID에 기반한 어플리케이션은 기존의 일반 어플리케이션과는 다른 특성들을 갖는다. 일반 어플리케이션은 프로그램 실행의 흐름이 사용자의 요구에 의해서 결정되지만, RFID에 기반한 어플리케이션은 사용자의 요구가 프로그램의 흐름을 결정하는 것뿐만 아니라 태그의 인식이 프로그램의 흐름을 결정 할 수도 있다. 즉 리더에서의 태그의 인식이 실시간으로 이벤트를 발생시키며 이러한 이벤트에 의해 프로그램이 동작된다. 이러한 이유로 RFID에 기반한 어플리케이션에는 실시간 이벤트 처리, 실시간 이벤트 모니터링, 분산처리, 멀티쓰레딩의 지원이 반드시 필요하다. 이러한 기능들은 모든 RFID 어플리케이션에 공통적으로 필요한 요소이므로, 이를 프레임웍으로 제공한다면 개발자는 GUI로직과 비즈니스 로직만을 구현할 수 있고, 따라서 단기간 내에, 견고하고 에러가 적은 RFID 어플리케이션을 개발할 수 있다.

* 본 연구는 유비쿼터스 컴퓨팅 및 네트워크 원천기반 기술개발사업인 21세기 프론티어 사업단에 의해 지원되었음.

본 논문에서는 이러한 RFID 어플리케이션이 갖는 고유한 특성들을 분석하여 공통적인 컴포넌트를 추출하고 이를 위한 프레임웍을 설계, 구현함으로서 RFID관련 어플리케이션 개발 시에 코드의 재사용성을 늘리고 안정된 어플리케이션을 구현하는 방법에 대해 논의하고자 한다. 2장에서는 RFID 시스템 구조에 대해서 설명하고 3장에서는 본 연구가 제안하는 어플리케이션의 프레임웍의 구조를 소개한다. 4장에서는 결론 및 향후 연구 방향을 제시한다.

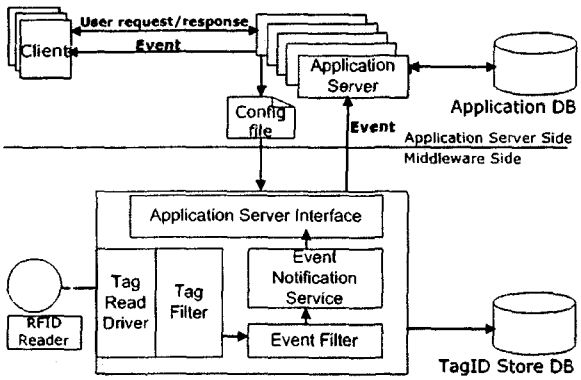
2. RFID 시스템 구조

RFID기반 어플리케이션은 사용자의 요구에 의해서 프로그램이 수행되는 것뿐만 아니라, 태그 인식에 의한 이벤트에 의해서도 프로그램이 수행되도록 설계하여야 한다. [그림 1]과 같이 어플리케이션 서버는 미들웨어 위에서 동작한다[1]. 어플리케이션에서 미들웨어 측으로 자신이 관심 있는 이벤트의 조건들을 전송하면 미들웨어 측에서는 어플리케이션 측에서 받은 이벤트 발생 조건들을 검사하여 조건에 맞을 때만 그에 맞는 이벤트를 발생시키게 된다.

예를 들어 어떤 위치에서 임의의 RFID를 가진 태그가 리더에 읽히게 된다든지, 임의의 위치에서 2개의 다른 태그가 리더로부터 동시에 읽히게 되면 이벤트를 발생시키라는 등의 조건이다. 따라서 미들웨어는 어플리케이션 측으로부터 받은 이벤트 조건들을 검사하고 그에 따른 이벤트를 발생시켜 어플리케이션 서버 측으로 전송함으로서 어플리케이션 서버가 동작하도록 한다.

또한 어플리케이션 서버 내에서도 역시 클라이언트 측으로 이벤트를 발생시키게 되는데 이러한 이벤트는 미들웨어로부터 읽은 태그 데이터를 어플리케이션이 분석하

여 클라이언트 측으로 전송하는 이벤트이다. 미들웨어는 실시간으로 어플리케이션 측에 이벤트를 전송하는 것 이외에도 리더로부터 읽힌 태그 데이터를 미들웨어 측에서 관리하는 데이터베이스에 저장하게 된다. 이는 어플리케이션 측에서 요구하는 이벤트 기반의 실시간 태그 데이터의 처리와는 별도로 RFID의 컨텍스트 추적, 히스토리 추적에 필요하며 어플리케이션 측의 요구로 미들웨어의 어플리케이션 서버 인터페이스를 통해 데이터를 제공하게 된다.



[그림 1] RFID 시스템 구조

3. RFID 어플리케이션 프레임웍

3.1 어플리케이션 프레임웍의 요구사항

RFID 어플리케이션은 기본적으로 분산 환경에서 작동하며, 태그에 의한 이벤트 발생에 의해 프로그램의 로직 수행이 가능해야 하며, 태그 데이터의 실시간 감시 및 비동기 이벤트 처리도 가능해야 한다. 프레임웍은 이러한 기능을 지원하도록 해야 한다. 다음은 어플리케이션 프레임웍이 가져야 할 기능적 요구사항이다.

3.1.1 클라이언트와 어플리케이션 서버의 분리

클라이언트 측은 GUI 같은 클라이언트의 편의를 위한 로직만이 위치하게 되고 실제적인 작업은 어플리케이션 서버 측에서 수행하게 함으로서, 어플리케이션의 수정 시에 개발자는 클라이언트 프로그램의 재배포 없이 어플리케이션 서버 측의 로직만을 수정해도 되는 효과를 볼 수 있다. 현재 RFID 어플리케이션은 어플리케이션 서버와 클라이언트 사이를 이벤트에 기반 하여 통신하므로, 어플리케이션 서버 측의 컴포넌트들과 클라이언트 측의 어플리케이션 이벤트 리스너 컴포넌트를 컴포지션 하기 위한 프레임웍의 지원이 필요하다.

3.1.2 이벤트 처리

어플리케이션 서버에서 처리하는 이벤트는 크게 2가지 타입이 있다. 첫째는 미들웨어로부터 전송되는 미들웨어 이벤트, 둘째는 어플리케이션 서버 자체 내에서 생성하는 어플리케이션 이벤트이다.

-미들웨어 이벤트

미들웨어 측에서 전송되는 이벤트 중에서 실시간으로 클라이언트 측으로 전송되는 이벤트이다. 이러한 이벤트는 어플리케이션 서버 측에서 정의한 이벤트를 미들웨어 측으로 전송하면 미들웨어의 이벤트 필터 루틴을 통해 조건을 검사하여 조건에 맞으면 어플리케이션 서버를 통해 클라이언트 측으로 전송하는 이벤트이다.

-어플리케이션 이벤트

미들웨어에서 읽은 태그 데이터를 어플리케이션 서버 내에서 분석하여 클라이언트 측으로 발생하는 이벤트이다. 이러한 이벤트는 어플리케이션 서버 내에서 설정한 의미 있는 조건을 만족하면 클라이언트 측으로 전송하는 이벤트이다.

위와 같이 미들웨어로부터 전송되는 이벤트를 처리할 수 있는 기능과 어플리케이션 서버로부터 클라이언트로 전송되는 이벤트를 처리하는 기능은 컴포넌트로 구현하고 프레임웍은 컴포지션 할 수 있어야 한다.

3.1.3 실시간 모니터링

RFID 태그의 인식은 실시간으로 계속 모니터링이 가능해야 한다. 즉 RFID 태그의 이동 감지 및 히스토리 추적, 컨텍스트 정보 등을 실시간으로 감시해야 하는 기능이 컴포넌트로 구현되고 프레임웍은 이를 지원해야 한다.

3.1.4 비동기 이벤트 처리

RFID 어플리케이션은 비동기 이벤트를 처리 할 수 있는 기능이 컴포넌트로 제공되고, 프레임웍은 이를 컴포지션 할 수 있어야 한다.

3.1.5 사용자의 요구 처리

RFID 어플리케이션이 이벤트 기반의 어플리케이션이기는 하지만 또한 사용자의 요구에 의한 프로그램 데이터의 처리도 지원해야 한다. 이러한 사용자의 요구를 받아들이는 컴포넌트가 필요하며, 프레임웍은 이러한 컴포넌트를 컴포지션 할 수 있어야 한다.

3.1.6 분산처리 및 멀티 쓰레딩

RFID 시스템은 미들웨어, 어플리케이션 서버, 클라이언트가 서로 네트워크를 통해서 통신하므로 분산처리에 대한 기능이 필요하고, 또한 어플리케이션 서버에는 다수의 클라이언트가 붙을 수 있는 구조이므로, 멀티 쓰레딩에 대한 기능도 필요하다. 이는 컴포넌트로 구현되고 프레임웍은 이를 지원해야 한다.

3.2 RFID 어플리케이션 컴포넌트

위의 RFID 어플리케이션 프레임웍의 요구사항으로부터 RFID 기반 어플리케이션 프레임웍에는 다음과 같은 기능을 하는 컴포넌트가 있어야 한다. 다음은 어플리케이션 프레임웍을 구성하기 위한 각각의 컴포넌트에 대한 설명이다.

a. 미들웨어 인터페이스 컴포넌트

미들웨어와의 통신 링크를 설정하고 미들웨어로부터 발생하는 이벤트를 받아들이기 위해 대기하는 리스너

인터페이스 컴포넌트이다.

b. 클라이언트 매니지먼트 컴포넌트

어플리케이션 서버에서 발생하는 이벤트를 클라이언트에 전송하기 위해 이벤트를 관리하고 현재 어플리케이션 서버에 등록된 클라이언트 중에서 적절한 클라이언트를 검색하고 조건에 맞는 클라이언트에게 이벤트를 전송하기 위한 컴포넌트이다.

c. 데이터베이스 인터페이스 컴포넌트

어플리케이션 서버 내에서 외부의 데이터베이스에 접속하기 위한 루틴을 구현한 인터페이스 컴포넌트이다

d. 클라이언트 인터페이스 컴포넌트

클라이언트 측의 요구를 받아들이기 위해 대기하는 리스너 인터페이스 컴포넌트이다.

e. 실시간 이벤트 모니터링 컴포넌트

태그의 인식 결과를 실시간으로 검색해서 이벤트를 발생시키는 컴포넌트이다.

f. 비즈니스 로직 컴포넌트

어플리케이션 서버에서 수행할 비즈니스 로직이 들어간 컴포넌트이다.

g. 어플리케이션 서버 관리 컴포넌트

어플리케이션을 미들웨어에 등록, 등록해제 시키며, 등록된 어플리케이션이라면 시작이나 종료를 시킬 수 있게 하는 컴포넌트이다.

h. 어플리케이션 서버 설정 컴포넌트

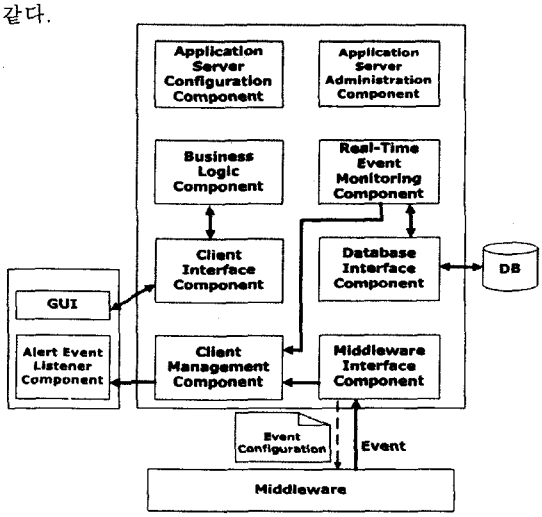
서버 리소스의 접근허가 등의 보안관리, 허용되는 사용자관리 등의 설정을 할 수 있는 컴포넌트이다.

i. 경고 이벤트 리스너 컴포넌트

어플리케이션 서버 측에서 클라이언트로 이벤트를 전송할 때 클라이언트 측에서 이벤트를 감시하고 있는 컴포넌트 이다.

3.3 RFID 어플리케이션 프레임웍의 구조

어플리케이션 프레임웍의 전체적인 구조는 [그림 2]와 같다.



[그림 2] RFID 어플리케이션 프레임웍 구조

[그림 2]에서 보는 것과 같이 클라이언트는 어플리케이션 서버로부터 2가지 종류의 이벤트를 받게 된다. 첫 번째는 미들웨어 측에서부터 받는 이벤트로서, 어플리케이션 서버의 미들웨어 인터페이스 컴포넌트가 미들웨어 측에서 전송되는 이벤트를 받게 되고 이는 클라이언트 매니지먼트 컴포넌트에 등록된 클라이언트들에게 클라이언트 경고 이벤트 리스너 컴포넌트를 통해서 전송된다. 또 한 가지는 어플리케이션 서버 자체내에서 발생하는 이벤트로서 리얼타임 이벤트 모니터링 컴포넌트가 일정 시간간격으로 데이터베이스를 검색하여 변경사항을 클라이언트 측으로 이벤트로 전송하는 이벤트이다.

개발자는 비즈니스 로직, GUI, 미들웨어 인터페이스 컴포넌트와 클라이언트의 경고 이벤트 리스너 컴포넌트의 이벤트를 처리하는 이벤트 핸들러를 재정의 하면 된다. 이에 프레임웍은 프로그램이 동작하기 위한 각각의 컴포넌트들 간의 링크를 설정하고, 네트워크를 통한 통신, 쓰레드, 이벤트 처리 구조를 지원하게 된다[2][3][4].

4. 결론 및 향후 연구 방안

본 논문에서는 RFID 어플리케이션에서 공통적으로 요구되는 컴포넌트들을 추출하고 이를 효과적으로 결합하기 위한 프레임웍을 설계, 구현함으로서 RFID관련 어플리케이션 구축 시에 코드의 재사용성을 늘리고 안정된 어플리케이션을 구현하는 방법에 대해 논의하였다. 현재 본 논문에서 제안된 프레임웍을 통해 실제 여러 어플리케이션을 적용하여 보았고 구조적이고 안정된 프로그램이 생성됨을 볼 수 있었다.

현재 구현된 시스템은 기본적으로 화이트박스(White box) 프레임웍이다[5].따라서 향후 연구 과제로는 이러한 프레임웍을 블랙박스(black box) 프레임웍으로 설계, 구현하는 것에 대한 연구가 필요하고 좀 더 많은 RFID 어플리케이션에 대한 사례연구가 필요하다.

참고 문헌

[1] Sun microsystems, "Sun's Auto-ID Architecture", June 2003

[2]R.E. Johnson, Frameworks = (Components + Patterns), Communication of ACM, Vol.40.No 10, October 1997.

[3]David Parsons, et al., A "Framework" for Object Oriented Frameworks Design,

[4]Erich Gamma et al, Design Patterns, Addison-Wesley,1995

[5]Mohamed E.Fayad, Douglas C.Schmidt. Object-Oriented Application Frameworks, Communication of the ACM, Vol.40,No 10, October 1997.