유비쿼터스 디지털 시스템 및 서비스의 구축 사례

*송교선, 박겨레, 최유락, 이세상평화, 김민영, 김윤삼, 조은선
충북대학교 전자전기컴퓨터공학부

Implementation of Ubiquitous Digital Systems and Services

*Kyo-Sun Song, Gyeo-Re Park, Yu-Raak Choi, SeSangPyoungHwa Lee,
Min-Young Kim, Yun-Sam Kim, Eun-Sun Cho
School of Electrical & Computer Engineering
Chungbuk University

Abstract

In this paper, we propose a simple middleware for ubiquitous computing systems. Developed with UPnP in Java, it efficiently organizes runtime interactions of devices. To show the feasibility of this system, we implemented a set of customized services based on this middleware. This prototype allows users to enjoy services at various spatial points even while moving around.

I. 서론

'유비쿼터스 컴퓨팅'이란 모든 사물이 사용자 서비스를 위해 상호 연결되고 지능화된 공간을 구성하기 위한 컴퓨팅 환경이다. 또한 컴퓨터는 눈에 보이지 않게 물리적 공간에 내재되어 제공되며 사용자의 상황에 따라 다양한 서비스가 제공되어야한다. 이러한 관련 연구들이 그간 적지 않게 있어왔지만 실제로 이러한 환경을 구축, 실험한 사례는 국내에 그리 많지 않다.

본 논문에서는 간단한 유비쿼터스 미들웨어를 제안하고 이를 이용한 서비스 구축 사례를 소개한다. 제안하는 유비쿼터스 미들웨어는 주어진 사용자를 위하여 서로 다른 장소에 있는 기기들이 연동되어 서비스 할 수 있도록 Java 와 uPnP를 사용하여 구성하였다. 그리고 이를 이용하여 사용자가 장소를 이동하였을 때에 이동전의 공간에 내재된 기기에서 받았던 자신이 좋아하는 영화나 음악 등의 서비스를 새로운 공간의 또 다른 장비에서도 끊어짐 없이 지속적으로 받을 수 있도록 하는 구현된 시스템인 1Inch 시스템 을 소개한다. 실험환경의 각 공간에는 소프트웨어 로봇 및 사용자 편의 서비스가 가능한 디스플레이 장비들이 각각 내재되도록 하였으며, 사용자 이동에 대한 감지는 카메라와 RFID를 통해 이루어졌다.

II. 미들웨어

2.1 동작 모델

미들웨어의 기본적인 동작 모델은 모든 Device에게서 발생한 event를 Event Deliver에 의해서 EventManager에게 전달하면 그에 상응하는 처리를 Ctrl을 통해서 Action을 취해주어 Device가 동작하게 해주는 구조이다. 그림 1은 이러한 구조를 보여준다.

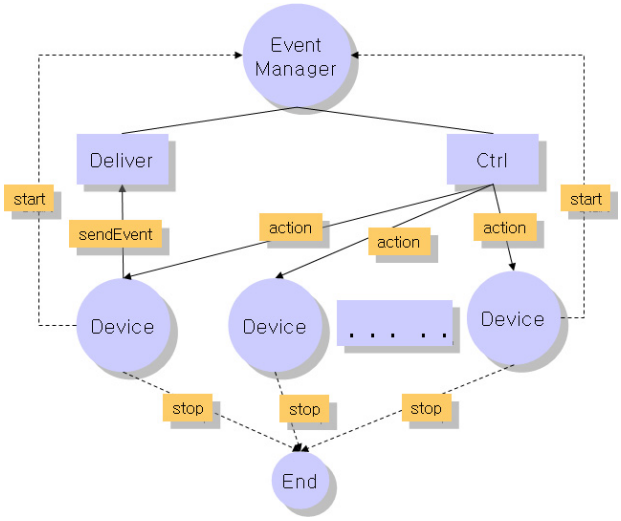

그림. 1 동작 모델

2.2 uPnP 구현

Universal Plug and Play의 약자로 연결하면 인식하는 Plug and Play의 개념을 네트워크까지 확장한 개념으로 로컬 네트워크에서 장치의 상태를 보거나 제어가 가능하다. 장치 제어용으로 CP(Control Point) 사용하고 있다.

UPnP protocol은 많은 표준들을 기반으로 하고 있다.(GENA, SSDP, SOAP, HTTPU, HTTP) 따라서 UPnP device를 만들기 위해서는 이런 protocol들을 이해하고 구현할 필요가 있다. CyberLink에서는 UPnP 개발을 위해서 자바 패키지를 제공하고 있다. uPnP를 이용하여 위의 모델을 구현했을 경우 몇 가지 단점이 존재한다.

첫째로 EventManager와 Device는 양방향 통신을 해야 하지만 UPnP에서는 불가능하다. 그 이유는 CP(Control Point)에서 Device로의 제어만이 가능하고 Device에서 CP로는 아무것도 할 수 없다. Device는 CP에 Action과 Attribute Description을 제공하고, CP는 제공된 Attribute Description을 바탕으로 Device를 제어하기 때문이다.

따라서 Device에서 발생한 이벤트를 EventManager에 전달하기 위한 별도의 방안 필요하고 2가지 정도로 해결방법이 있다.

2.2.1 체킹에 의한 방법

Device는 발생한 이벤트를 기억할 임의의 Event Attribute를 가지고, CP는 일정 시간마다 각 Device의 Event Attribute를 체크하여, 실행할 이벤트가 있을 경우 적절한 조치를 취한 후, Event Attribute를 초기화하는 것이다.

하지만, UPnP는 네트워크 통신에 의한 딜레이가 비교적 심하기 때문에 CP가 Event Attribute를 확인하기 전에 새로운 Event가 발생할 경우, 이전 Event는 처리되지 않고 사라지게 될 수가 있다. 이 경우 Event Attribute를 여러 개 둔다고 해도 근본적인 해결은 되지 않는다.

2.2.2 CP를 이용한 방법

EventManager에 이벤트를 전달하기 위해 Device에서 접근 가능한 특별한 CP 제공하는 것이다. 즉 Deliver는 EventManager에 이벤트 발생을 알리는 단일 메소드 sendEvent()만을 제공하는 작은 CP역할을 하는 것이다.

이렇게 할 경우 체킹에 의한 방법과 같은 문제는 발생하지 않지만, Device 하나당 하나의 CP가 만들어 지기 때문에 제공하는 메소드가 하나라 하여도, Device를 찾거나 리스트를 관리하는 것은 모두 동일하기 때문에 쓸데없이 많은 자원을 소요할 수 있다. 또, 하나의 장치에 둘 이상의 Device가 존재할 경우가 있는데 CP도 Device의 개수만큼 존재하기 때문에 A Device에 대한 A CP와 B Device에 대한 B CP가 있다고 할 때, B CP에서 Device들을 찾을 때, 그 결과를 A CP가 받는 경우 발생할 수도 있어 B CP는 온전한 Device list를 가지지 못하여, 통신이 제대로 이루어지지 않을 수 있다.

2.3 RMI 구현

RMI(Remote Method Invocation)는 자바가 제공하는 분산 객체들 간의 메서드의 호출을 말한다.
RMI를 사용할 때 Server는 rmiregistry를 띄워야만 하고 Server의 주소를 Client가 알고 있어야 한다. 또, 원격으로 사용할 객체가 수정되었을 경우, 해당 stub 및 skeleton 파일을 Client에 다시 적용하여야 하는 번거로움 있다.

UPnP에 비해서 보다 유연하게 기능들을 구현하는 것이 가능 하다. 일단 객체 통신이 가능하기 때문에 보다 다양한 이벤트를 처리할 수 있고 보다 나은 통신 속도를 보여준다. 하지만 네트워크 연결을 위해서 주소와 이름을 알고 있어야 하기 때문에 새로운 Device 추가 시에 재 컴파일이 필요하다.

위의 두가지 방법(UPnP, RMI)으로 구현을 하여 동작 테스트를 거쳐보았다. UPnP는 범용 프로토콜과 추후 확장이 쉽다는 장점이 있었지만 테스트에서는 그다지

만족스러운 성능을 보여주지 못해서 현재는 RMI로 미들웨어를 구현하였다.

III. 서비스 구현

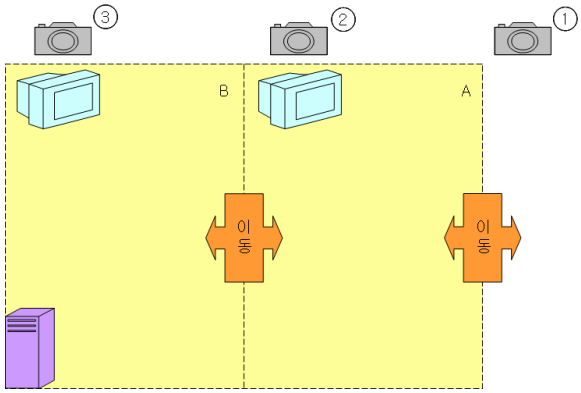

그림. 2 1 Inch Service Room

1 Inch 서비스는 그림 2와 같이 구성된 공간에서 제안한 미들웨어에 의해 동작하는 유비쿼터스 서비스이다. ① web cam은 방안에 사람이 없을시 방문하는 사람들을 캡쳐해서 저장해 놓는 역할을 하고 ②, ③ web cam은 A방이나 B방에 사용자의 움직임을 파악하여 사용자의 위치를 확인한다. 각 방에 설치되어 있는 화면에는 1 Inch 서비스 화면이 나온다. 만약 A방에서 1 Inch 서비스 중 하나인 영화나 음악 감상을 하다가 B방으로 이동하면 A방에서 보고 있던 영화나 듣고 있던 음악이 B방에서 이어서 서비스되어진다.

여기서 사용된 사용자 인식 방법과 1 Inch 서비스에 대해서 설명한다.

4.1 Motion Detector

Motion detection을 위해 사용된 Web cam은 초당 25 frame의 영상을 전송할 수 있다. 단순한 감지를 위해선 1frame/sec으로도 충분하다. 따라서 충분한 영상을 capture 하여 메모리에 저장을 할 수 있다.

이미지간의 변화를 감지하기 위해 잠시전의 이미지와 현재 이미지의 각 픽셀을 비교하는 것을 기본으로 하고 실시간으로 영상의 변화를 감지해야 하기 때문에 영상을 효과적으로 처리하기 위해서 Binary Conversion을 사용하였다.

4.1.1 Binary Conversion Algorithm

영상은 영상정보를 지니고 있는 독립적인 작은 이미지 픽셀(pixel)들로 이루어져 있으며 각각의 이미지 픽셀은 Index, Red, Green, Blue 이렇게 4가지의 정보로 구성된다. Index는 픽셀의 위치를 담은 정보이며 Red, Green, Blue 이렇게 기본적인 3원색의 정보를 규합하여 자신만의 독특한 색상을 만들어 낸다.

이러한 픽셀들이 모여서 하나의 '의미있는' 영상을 구성하게 된다. 따라서 각각의 픽셀은 특별한 '값'을 지니고 있으며 우리는 그 '값'을 통해서 이미지를 재구성할 수 있다.

Binary Conversion의 경우 각각의 값에 대한 정보를 일관적인 Integer 값으로 읽어 들인 후 그 값에 대한 히스토그램을 완성하고 각각의 값에 대한 평균값을 산출한다. 그 평균값 이상의 값을 가지고 있는 픽셀만 1로 표시하고, 그 이하의 값은 모두 0으로 처리하면 완벽히 이진 화된 영상을 추출해 낼 수 있다.

4.1.2 Image comparison

이미지는 web cam의 해상도 320x240에 의해 76800(320x240)픽셀이 존재하고 각 픽셀은 0과 1의 값을 가지게 된다. 각각의 픽셀 값을 비교해서 1초전 이미지와 현재 이미지의 차이를 수치화할 수 있다.
320x240의 해상도에서 사람의 눈으로 변화 없어 보이는 이미지가 수치화된 값에선 100~9000 사이에 나타나고 약간의 변화에도 15000 이상으로 값이 변하게 된다. 이를 기반으로 움직임을 감지할 수 있게 된다.

그림. 3 방 B에서 방 A로 이동

4.2 1 Inch Service

1 Inch 서비스는 전체화면으로 화면에 보여주며 기본적인 4가지 서비스인 영화보기, 음악듣기, 날씨정보, 방문자 사진을 보여주는 서비스와 간단한 동작을 하는 캐릭터로 이루어져 있다. 각 서비스에 필요한 리소스(영화, 음악, 사진)는 1 Inch Service Room을 구성하는 네트워크 내의 폴더 공유를 통해 이루어지는 구조를 가지고 있다.

영화와 음악을 자바에서 재생하기 위해서는 두가지 방법이 있는데 한 가지는 직접 미디어 파일을 다루어서 모든 처리를 사용자가 해주는 것과 JMF라고 하는 자바 미디어 프레임워크를 사용해서 미디어 파일을 다루는 방법이 있다. 현재 JMF 2.1.1e까지 나온 JMF는 직접 미디어 파일을 다루는 방법보다는 훨씬 나은 라

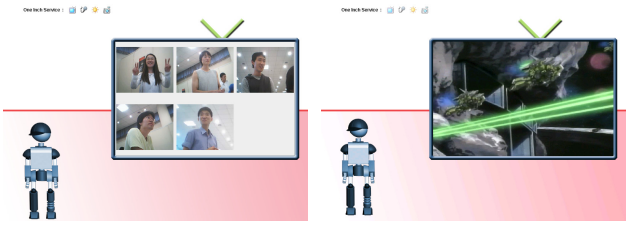

그림 4 방문자 사진보기와 영화보기 화면

이브러리를 제공해주고 있다. 음악의 경우는 화면에 출력하는 것이 없기 때문에 지정된 사진들을 슬라이드 쇼로 보여주게 되어있다.

날씨 정보는 기상 정보를 제공하는 kweather에서 데이터를 받는다. 받은 데이터는 텍스트 파일이기 때문에 간단한 파싱을 통해 화면에 표시를 해주게 된다.

방문자 사진을 보여주는 것은 사용자가 나가있던 사이 방문한 사람들의 사진을 보고 누가 왔다 갔는지 알 수 있게 해준다.

Ⅳ. 관련 연구

4.1 미들웨어 관련

Project Aura는 카네기 대학에서 연구한 유비쿼터스 아키텍쳐이다. Aura는 시스템 OS에서부터 하드웨어까지 모든 부분에 대해 연구를 하고 Aura 아키텍쳐를 만들어서 이것에 예로 Portable Help Desk 어플리케이션을 만들었다. PHD(Portable Help Desk)는 시간과 공간의 정보를 다루는데 즉, 사람에 대한 관련 정보, 위치, 방향, 스케줄 시간 등 데이터를 다룬다. Aura의 경우 많은 컨텍스트 정보들을 관리하는 서버들이 효율적으로 서로 처리하고 교환하는가에 중점을 두었다.

이와 다르게 PICO에서는 Camileun이라는 하드웨어 디바이스들과 Delelgent라는 Autonomous 소프트웨어 entity로 구성되어 필요 할 때마다 Community라는 것을 구성하여 정보들을 처리하게 한다. Camileun이라는 디바이스들로부터 들어오는 정보들을 Delegent에 의해 처리되어 진다. PICO는 소프트웨어 Agent에 의해서 자동으로 필요한 Community를 구성하여 정보전달을 할 것인지에 초점을 두었다.

우리는 각 디바이스에 의한 처리보다는 Aura와 같이 많은 컨텍스트 정보를 어떻게 서버에서 효율적으로 관리할 것인가에 대해 향후 연구 할 것이다.

4.2 서비스 구현 관련

MIT에서 Smart Room 프로젝트는 실제로는 보이지 않는 애완동물이 존재해서 사용자의 명령을 따르거나 도움을 준다. Smart Room은 vision-based tracking을 통해서 실제로 없는 애완동물과 사용자를 연결해주고 프로젝션을 통해 화면에 보여주게 꾸며져 있다.

UICU의 GAIA 시스템의 example로 Active space라는 것이 있다. Active space와 Smart Room은 우리와 같이 공간적인 서비스라는 측면은 비슷하다. 즉, 서비스 가능한 공간 내에서 사용자의 입력에 따라 서비스를 해준다.

1Inch 서비스 역시 서비스 가능한 공간 내에서 사용자의 입력에 따라 서비스를 해주는 것은 위의 두 시스템과 같다. 하지만 두 시스템은 서비스 공간 내의 사용자 이동은 고려하였지만, 1Inch 서비스처럼 서비스 공간을 이동하는 사용자에 대해서는 고려하지 않았다.

Ⅴ. 결론

본 논문에서는 유비쿼터스 미들웨어와 서비스를 제안하였다. 미들웨어는 자바의 RMI와 UPnP로 구현을 하였으며, 이러한 환경 하에 동작하는 유비쿼터스 서비스인 1 Inch 서비스를 통해서 사용자가 다른 곳으로 이동을 하더라도 이전 장소에서 받고 있던 영화나 음악 서비스를 이어서 받을 수 있는 것을 보였다. 향후에는 지금보다 다양한 이벤트를 처리할 수 있게 현재의 미들웨어를 개선하고 사용자에 보다 적극적인 서비스를 할 수 있는 지능화된 로봇과 네트워크 내에 리소스를 공유하여 영화나 음악을 보여주는 것이 아닌 VOD나 MOD와 같은 서비스를 제공하는 방식으로 구현되어야 할 것이다.

참고문헌

[1] David Garlan, Daniel P. Siewiorek,Asim Smailagic, and Peter Steenkiste Aura: Toward Distraction-Free Pervasive Computing

[2] Kumar, M. Shirazi, B.A. Das, S.K. Sung, B.Y. Levine, D. Singhal, M PICO: A Middleware Framework for Pervasive Copumting

[3] M. Roman, C.Hess, R. Cerqueira, A. Ranganathan, RH Campbell, K. Nahrstedt. Gaia: A Middleware Infrastructure for Active Spaces. IEEE Pervasive Computing Vol. 1, No. 4, p. 74-83. October - December, 2002