유비쿼터스 컴퓨팅 환경을 위한 경량의 서비스 상호 작용 브로커 개발

문상철°, 차영록, 이경민, 이동만
한국정보통신대학교
{sangchul°, u00cha, kmlee, dlee}@icu.ac.kr

Development of a Lightweight Service Interaction Broker for Ubiquitous Computing Environments

Sangchul Moon°, Youngrock Cha, Kyungmin Lee, Dongman Lee
Information and Communications University

요 약

본 연구는 유비쿼터스 컴퓨팅 환경에 존재하는 다양한 서비스들 간의 상호 작용을 지원하는 서비스 상호 작용 브로커의 설계와 구현에 대해 소개한다. 유비쿼터스 컴퓨팅 환경을 위한 서비스 상호 작용 브로커는 (1) 소형 기기에서도 동작이 가능해야 하며, (2) 환경에서 제공되는 다양한 서비스들과 상호 작용이 가능해야 하며, (3) 확장이 용이해야 한다. 본 연구에서 개발한 서비스 상호 작용 브로커는 불필요한 기능을 제거함으로써 사이즈를 크게 줄였다. 또한 다른 웹 서비스 미들웨어들과의 상호 운용성을 갖추고 있다. 마지막으로 계층화된(layered) 구조를 채택하여 확장이 용이하도록 하였다.

1. 서론

유비쿼터스 컴퓨팅 환경에서는 컴퓨터 및 네트워크 기능을 내재된 지능객체(smart object)들이 서로 상호 작용하며 사용자에게 최적의 서비스를 제공한다[1]. 유비쿼터스 컴퓨팅 환경의 효과적인 구축을 위해서는 지능객체 간 또는 지능객체와 외부 서비스 간의 상호 작용을 지원하기 위한 서비스 상호 작용 브로커가 필요하다.

이러한 상호 작용을 지원하기 위한 분산 객체 미들웨어로는 가장 대표적으로 CORBA와 DCOM을 비롯해서 자바의 RMI, 그리고 웹서비스 미들웨어로써 가장 널리 사용되는 AXIS가 있다. 하지만 이들 미들웨어들은 대부분 데스크탑 이상의 서버 규모의 환경을 목표로 구현되어 있다. 따라서 필요한 컴퓨터 성능이나 메모리 크기등과 관련해 미들웨어 작동 환경에 대한 요구조건이 유비쿼터스 환경에서 사용되는 소형 기기들과는 다르다는 문제점이 있다. 따라서 이들 미들웨어들을 유비쿼터스 컴퓨팅 환경의 소형 기기들을 위해 사용하기에는 무리가 따른다.

본 논문은 Service Interaction Broker (이하 SIB)라는 기존 분산 객체 미들웨어의 주요 기능을 계승하고 불필요한 기능을 제거하여 유비쿼터스 환경에 맞는 소형 기기들에 적합한 새로운 상호작용브로커를 개발하였다. SIB는 계층화(layered) 구조로 되어 있어 미들웨어의 주요 기능을 다른 기능들에 영향을 미치지 않고 교체할 수 있으며, 소형 기기에 탑재할 수 있을 만큼 경량화 되어 있고 주요 웹서비스 미들웨어들과 상호 운용성을 고려하여 설계되었다. 이러한 SIB는 유비쿼터스 컴퓨팅 환경을 가정 내에 구현하는 프로젝트인 Active Surroundings에서 주요 서비스들과 응용 애플리케이션을 서로 상호 통신하게 하는 기반으로써 사용되고 있다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 본 연구에 관련된 기술과 연구들에 대해 알아보고 3장에서는 다자인시 고려사항에 대해, 4장에서는 구현에 대해 기술한다. 5장에서는 구현된 SIB를 어떻게 테스트했는지 기술하였으며, 마지막으로 6장에서는 결론 및 향후 연구 과제에 대해 기술한다.

2. 관련 기술

CORBA (Common Object Request Broker Architecture)는 200여개의 연구기관, 기업, 학교가 객체 기술 연구를 위해 만든 OMG (Object Management Group)에서 표준을 제정, 관리하는 분산 객체 미들웨어 기술이다[5].

DCOM (Distributed Component Object Model)은 Microsoft사의 분산 컴포넌트 모델 구현이다. 이 모델은 오직 Microsoft사의 운영체제에서만 지원되기 때문에 다양한 환경에서 작동해야하는 유비쿼터스 환경의 기기들을 지원할 수 없다.

AXIS(Apache Extensible Interaction System)[2]는 Apache SOAP 프로젝트의 차세대 형태로서, 자바 언어에 기반을 둔 SOAP 의 오픈 소스 프로젝트며, 가장 널리 사용되고 있는 웹서비스 개발 미들웨어이다. 하지만 AXIS는 Java 2 Enterprise Edition (J2EE) 라이브러리를 비롯한 수많은 자바 라이브러리들을 사용하고 있기 때문에 그 크기가 상대적으로 매우 크다. 따라서 소형기기에 탑재 할 수

없다는 제약 조건을 가지고 있다.

WSOAP[3]과 KSOAP[4]은 웹서비스 클라이언트 모듈로써, AXIS의 SOAP 클라이언트 모듈들에 비해 크기가 매우 작고 J2ME 환경을 목표로 개발되었기 때문에 소형기기에 탑재될 애플리케이션 개발에 적합하다. 하지만 이들 라이브러리들은 오직 웹서비스 클라이언트 모듈만을 지원하기 때문에 서버와 클라이언트 기능을 모두 필요로 하는 SIB를 구현하기에는 많은 제약 조건을 가지고 있다.

따라서 우리는 AXIS와 WSOAP이 가지고 있는 이러한 단점들을 고려하여 Web Services 클라이언트 및 서버 모듈을 모두 지원하는 경량화된 SIB를 구현하였다.

3. 디자인 고려사항

SIB는 주요 기능들을 다른 기능들에 영향을 미치지 않고 교체할 수 있는 계층화(layered) 구조, 소형 기기들에 탑재될 애플리케이션 개발을 위한 경량화, 그리고 타 웹서비스 미들웨어들과 상호운용성을 부여하는데 초점이 맞춰져 있다.

다음은 본 연구를 통해 새롭게 디자인한 SIB의 전체 구조를 나타낸 그림이다.

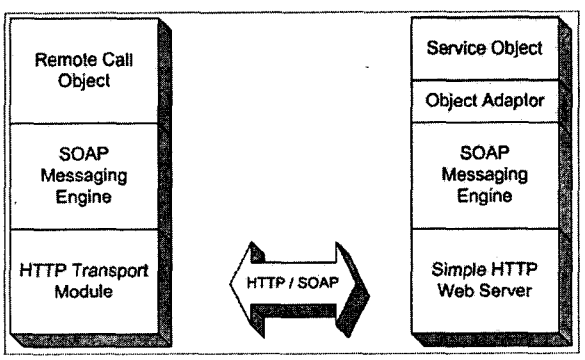

그림 1. SIB 구조

위에서 보이는 바와 같이 SIB는 크게 세 개의 계층으로 구성되어 있다. 가장 상위 계층은 원격 메소드 호출을 담당하는 Remote Call Object와 실제 웹을 통해 서비스할 로직을 담고 있는 서비스 객체 및 서비스 객체를 관리하는 Object Adaptor가 위치한다. 중간 계층은 객체 호출 및 인자들을 표준 규격에 맞춰 SOAP 메시지로 마샬, 언마샬 하며, 하위 계층은 이러한 SOAP 메시지를 HTTP 프로토콜을 이용해 실제 통신을 수행한다.

이 아키텍처는 다른 계층들에 영향을 주지 않고 세부 구현이 투명하게 교체될 수 있도록 계층들간의 인터페이스가 디자인되어 있다. 가령 SOAP 메시지 전송수단을 HTTP가 아닌 다른 프로토콜을 사용하고자 할 경우, 다른 프로토콜을 지원하는 통신 구현이 계층들간의 인터페이스를 구현하고 있다면 다른 계층들에 전혀 영향을 미치지 않고 간단히 교체할 수 있다.

SIB는 현재 SOAP 1.1의 표준을 지키고 있으며 상호호환성 테스트를 위해 아파치의 AXIS및 Wingfoot사의 WSOAP을 기준으로 상호 통신이 가능하도록 디자인 되었다. 현재 지원 가능한 타입은 Integer, String과 같은 기본 타입들과 기본 타입의 배열, 개발자가 선언한 구조체 및 구조체의 배열, 중첩된 구조체와 그 배열들이다.

Remote Call 및 Object Adaptor 계층은 애플리케이션 개발자가 쉽고 간편하게 미들웨어의 기능을 이용할 수 있도록 *inout, out* 파라메터나 서비스 생명주기 관리 정책과 같이 자주 사용되지 않는 기능을 과감히 제외시켜 단순한 인터페이스의 형태로 디자인되었다. 그리고 애플리케이션 로직에 따라 Callback 메커니즘을 사용해야 할 경우 이를 편리하게 지원할 수 있도록 Object Reference 개념을 도입하여 개발자가 원격 객체의 레퍼런스를 원격 메소드 호출의 인자로 손쉽게 전달할 수 있게 제공하고 있다.

4. 구현

개발은 Linux Redhat 8에서 자바 언어로 이루어졌다. 개발툴은 Sun의 Java Development Kit 1.5을 사용했으며 소스 및 바이너리 코드는 자바 버전 1.2를 기준으로 제작되었다. SOAP과 WSDL 메시지 제작 및 해석을 위한 XML 파싱은 SAX 파서를 이용했으며, 파서 구현은 아파치에서 제공하는 xerces를 이용하였다.

SIB에서 HTTP 메시지를 처리하는 서버는 I/O 모델 측면에서 두 가지 버전으로 제작되어 있다. 하나는 클라이언트의 Request당 하나의 쓰레드를 할당하여 처리하는 방식으로, 이는 크기가 작고 구현이 간단하며 개발자가 쓰레드에 대해 고민할 필요 없이 서비스 객체를 개발할 수 있다. 하지만 Request의 수에 따라 서버 쓰레드의 수가 비례해서 증가하기 때문에 리소스를 많이 사용하며 멀티 쓰레드들간의 컨텍스트 스위칭으로 인해 성능이 저하된다는 문제가 있다.

다른 하나는 Java 1.4의 NIO 기능을 이용하여 단일 쓰레드에서 Non-Blocking 소켓과 I/O 멀티플렉서를 이용하여 구현하였다. 이 방법은 상대적으로 구현이 복잡하고 최신의 자바 실행환경이 필요하며, 서비스 객체 개발자가 필요에 따라 서비스 객체 내에서 직접 쓰레드를 생성해서 구현해야 한다는 단점이 있다. 하지만 서버 로직을 처리하기 위

해 쓰레드를 단 한 개만 사용하기 때문에 리소스 사용이 적고, 쓰레드간 컨텍스트 스위칭이 없기 때문에 수행속도가 빠르다. 따라서 많은 Request를 처리할 필요가 없는 작은 소형 기기는 전자의 I/O 모델을 사용하여 크기를 줄이는 것이 낫고 상대적으로 많은 Request를 처리해야 하는 경우에는 후자를 선택하는 것이 바람직하다. 물론 이러한 I/O 모델들은 위에서 설명한 것처럼 다른 코드에 영향을 미치지 않고 쉽게 바꿀 수 있다.

SOAP 메시지를 만들고 해석하는 부분을 구현하는 과정에서 우리는 JUnit 프레임워크를 이용한 유닛 테스트들을 작성하여 테스트 수행을 자동화하였다. 이러한 테스트들은 미들웨어가 SOAP 메시지 해석과 작성 기능이 정상적으로 작동하는지 검사한다. 이를 이용해 만일 개발자가 코드를 수정했을 경우, 해당 수정 부분이 기존 SOAP 메시지 작성 및 해석 기능 수행에 영향을 미쳤는지 여부를 빠르게 테스트할 수 있도록 하여 개발 능률을 크게 향상시킬 수 있었다.

5. 테스트

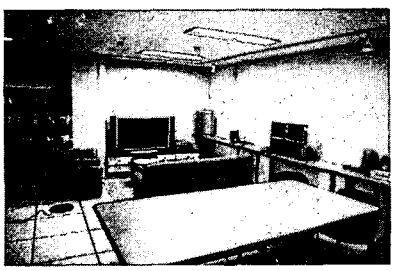

그림 2. Active Surroundings 데모룸

SIB가 소형 기기에서 성공적으로 작동하는지 확인하기 위해 Active Surroundings환경에서 사용되는 서비스들과 연계하여 간단한 데모 애플리케이션을 작성하였다. 소형 기기는 HP사의 iPaq RX 3700 PDA를 이용하였다. 데모 애플리케이션은 Active Surroundings의 디스커버리 서비스를 통해 현재 데모룸 내에 사용 가능한 서비스가 무엇이 있는지 그 목록을 전달 받아 화면에 표시한다. 만일 데모룸 내에서 새로운 서비스가 발견되어 서비스 목록에 변경이 생겼을 경우, 디스커버리 서비스는 원격 메소드 호출을 통해 새로운 목록을 PDA로 전송한다. 이 데모 애플리케이션 시나리오를 통하여 우리는 다음의 사항을 테스트 할 수 있다.

(1) PDA에 사용될 애플리케이션을 작성할 수 있을 만큼 미들웨어의 크기가 작은가.

(2) PDA에 탑재된 미들웨어가 서비스 객체를 등록하여 외부 메소드 호출을 처리할 수 있는가.

(3) 콜백 메커니즘이 동작하는가.

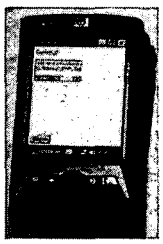

그림 3. PDA 탑재

테스트 결과, 우리의 새 구현은 위의 세 가지 요구 사항을 모두 만족한다는 사실을 확인하였다. 미들웨어 크기는 현재 800KB 정도로 기존 구현에 비해 약 5분의 1 수준으로 줄어들어 PDA 수준의 소형 기기에 탑재하는데 무리가 없었다. 또한, 작성된 데모 애플리케이션을 통해 PDA에서 데모룸의 사용 가능한 서비스 목록을 화면에 실시간으로 표시하였으며, PDA는 디스커버리 서비스에 자신이 관리하는 서비스 객체의 레퍼런스를 전달하여 상황이 변화했을 경우 통보 받을 수 있었다.

6. 결론 및 향후 과제

본 연구를 통해 우리는 유비쿼터스 컴퓨팅환경에서 소형 기기들이 외부 서비스들과 상호 통신하는 애플리케이션을 쉽게 작성할 수 있도록 지원하는 미들웨어를 개발하였고 가정 내 유비쿼터스 컴퓨팅환경을 구현하기 위한 Active Surroundings 프로젝트 환경에서 성공적으로 테스트를 수행하였다.

차후 본 구현은 좀 더 다양한 애플리케이션 구현을 통해 기능을 보완, 발전시킬 것이며, PDA보다 더 작은 소형 기기들을 지원할 수 있도록 J2ME의 CLDC 환경을 지원할 예정이다.

7. 참고 문헌

[1] D. Saha, A. Mukherjee, Ubiquitous Computing: *A Paradigm for the 21st Century*, IEEE Computer, IEEE Computer Society Press, pp. 25-31, March 2003.

[2] Apache AXIS Project : http://ws.apache.org/axis/

[3] Wingfoot SOAP : http://www.wingfoot.com/

[4] KSOAP : http://kobjects.sourceforge.net/

[5] CORBA : http://www.omg.org