분산형 홈 네트웍에서 동적 서비스 지원을 위한 코바 기반 네이밍 서비스

김도훈[0],박준호,오주용,강순주,문경덕
경북대학교 실시간 시스템 연구실,한국전자통신 연구원
(zamggan[0],zec,anyong)@palgong.knu.ac.kr,sjkang@ee.knu.ac.kr,kdmoon@etri.re.kr

CORBA based Naming Service for Supporting Dynamic Services in Distributed Home Network

Do-Hoon Kim[0] ,Jun-Ho Park, Ju-yong Oh , Soon-Ju Kang, Kyeong-Deok Moon
Real Time Systems Laboratory, Kyungpook National University
Electronics and Telecommunications Research Institute

요 약

분산 네트웍환경하에서 코바의 네이밍 서비스는 네임 서버에 저장한 논리적 이름을 통해서 객체의 위치 정보를 얻는다.이 방식은 서버의 물리적인 위치 변화에 상관없이 객체의 위치 투명성을 제공한다. 그러나 기존의 네이밍 서비스의 기능은 단순히 데이터베이스 역할만 할 뿐, 동적인 환경 하에서 서버와 클라이언트간의 유동적인 실효정보를 업데이트 하지못하고 네임 서버도 하나의 중앙 집중서버 구조로 되어있어서 홈 네트웍상에서 룸 단위의 디바이스 정보를 처리하는데 있어서 적합하지 않다.본 논문에서는 이러한 문제를 해결하고 분산형홈 네트웍 환경에 적합한 서비스를 제공하기 위해 OMG 네이밍 서비스를 기반으로 하여 새로운 네임 서버구조를 설계하고 디바이스 정보를 효율적으로 제공할 수 있는 모델을 제시한다.

1. 서 론

홈 네트웍 환경은 다양한 가전기기들이 연결되어, 제공되는 각각의 서비스에 대한 정보의 상호교환과 제어, 감시가 필요한 특수한 형태의 분산 컴퓨팅 환경이다.
홈 네트웍에서는 이러한 분산 컴퓨팅 환경하에서 존재하는 여러 서비스들을 효율적으로 관리하고 네트웍 하부 구조를 추상화 함으로써 사용자가 보다 편리한 홈 제어를 할 수 있도록 미들웨어를 기반으로 하는 애플리케이션이 필요하다. 본 논문에서는 홈 네트웍 환경에 적합한 분산 환경의 표준 미들웨어로써 CORBA(Common Object Request Broker Architecture)를 선택하였다.
클라이언트/서버 구조에서 코바를 기반으로 한 분산 객체 애플리케이션 개발은 플랫폼과 프로그래밍 언어에 독립적이라는 코바의 장점을 활용함에도 불구하고 분산된 객체들의 상호연동을 고려해야 하기 때문에 개발하기가 쉽지 않다. 따라서 분산 애플리케이션의 확장과 개발을 위해서는 OMG(Object Management Group) 에서 제정한 코바서비스중에 분산 환경에서 객체를 쉽게 찾을 수 있도록 제공하는 네이밍 서비스를 사용한다.
코바에서 ORB객체가 다른 객체의 메소드를 호출하려면 그 객체의 객체 참조자를 얻어야 한다. 객체 참조자를 얻는 방식에서 벤더에서 제공하는 고유의 함수를 이용하려면 객체 참조자를 얻기 위해 호스트 이름 또는 서버 이름과 같은 물리적인 정보가 필요하다. 따라서 서버가 다른 호스트로 자리를 옮길 경우, 이 서버와 통신하는 모든 클라이언트 코드내의 함수호출 부분에서 바뀐 서버 이름을 적용해줘야 한다. 네이밍 서비스는 이러한 개발상의 비효율성을 개선하고 분산된 객체를 쉽게 찾을 수 있는 구조를 제공한다. 즉, 코바 표준으로써 다른 ORB간의 호환이 되고 객체의 인터페이스, 서버 또는 호스트 이름과 무관한 사용자정의의 논리적인 이름으로 객체를 구별하므로 객체 기능 위주의 특정한 이름을 부여할 수 있는 동시에 위치 투명성을 제공하는 장점이 있다.
그러나 기존의 이러한 네이밍 서비스가 제공하는 기능은 분산형 홈 네트웍 환경에 적용하기에는 기능이 부족한 면이 많다. 만약,모든 디바이스의 정보를 저장하는 네임 서버가 다운 됐을 경우에는 중앙 집중 서버 구조로 인해 홈 내부의 모든 디바이스의 정보가 사라지게 될 것이다. 그리고 네임 서버에 객체를 등록한 뒤에 그 객체의 서비스를 요청할 경우 , 요청하기 직전에 해당 디바이스가 연결이 끊어졌다거나 홈내의 다른 곳으로 이동 했을 경우에는 네임 서버에는 기존에 등록된 불필요한 객체의 위치정보를 클라이언트측에 넘겨주게 되어 올바른 홈 내의 제어가 불가능하게 될 것이다.
본 논문에서는 이와 같은 문제점 때문에 기존의 네이밍 서비스에 홈 네트웍이라는 특수한 환경을 위한 구조를 설계,추가하고 동적인 디바이스의 실효정보를 효율적으로 업데이트하기 위한 모델을 제시하고자 한다.

2. 요구 분석 및 전체 클래스 다이어그램

분산형 홈 네트웍 환경에서는 서비스를 제공하는 디바이스에 대한 유동성을 고려해야 한다. 그래서, 홈 내에 연결되어 있는 유동적인 디바이스들에 대한 원격제어와 모니터링을 위해서는 정확하고 신뢰성 있는 정보로의 업데이트가 필요하다. 현재의 네이밍 서비스 같은 역할 만으로는 각 디바이스에 대한 실효정보를 기대하기 어렵기

때문에 디바이스 실효정보의 이동과 변화를 체크해서 적절하 게 업데이트를 해줄 확장된 네임서버의 역할이 요구된다. 확장된 네임 서버는 디바이스 정보에 대한 신뢰성과 디바이스 위치에 관련없이 제어,감시가 가능한 위치 투명성을 유지해야 하므로 IOR과 디바이스 네임을 기준으로 디바이스 정보를 저장하였다. 그러나 기존의 방법대로 해쉬테이블만 써서 정보를 저장하면 디바이스의 위치와 서비스종류만으로 검색을 하고자 할 경우에는 적합하지 않기 때문에 검색을 위한 저장공간을 따로 추가할 필요가 있다. 그리고 전체적으로는 물리적 단위를 기반으로 한 계층적 구조를 위해 홈 서버는 룸서버를 관리하고 룸서버는 각 디바이스를 관리하도록 하였다. 여기에 일시적으로 룸 서버가 그 기능을 상실할 경우에 다른 서버에 영향을 주지 않고 디바이스 정보를 간직할 수 있는 결함허용성을 보장할 수 있어야 한다. 그래서 이를 위한 타 룸서버로의 디바이스 정보를 전달하는 기능이 필요하다. 그리고 서비스를 사용할 때 서비스 duration을 추가함으로써 동적인 환경에서 그 duration에 맞추어 실효정보를 업데이트 할 수 있는 사용자 기반의 제어기능도 추가할 필요가 있다. 다음 그림은 위에서 설명한 몇가지 기능들을 기반으로 네이밍 서비스를 구성하는 전체적인 클래스 다이어그램을 나타낸 것이다.

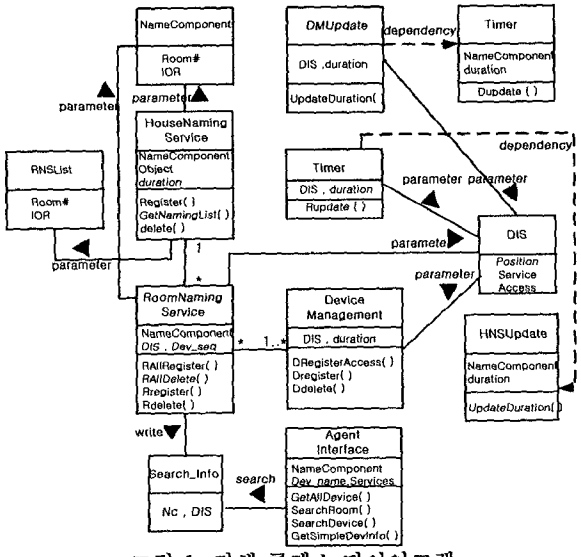

그림 1. 전체 클래스 다이어그램

3. 제안하는 네임서버 구조

3.1 하드웨어 구성

본 논문에서 바탕이 되는 하드웨어 구성은 다음과 같다. 한 가정에 하나의 홈 서버가 존재하고 각 방마다 룸 브리지[3]라는 각 방의 디바이스 관리 서버가 존재한다. 룸 브리지는 방이라는 물리적 단위로 구분되어지고 각 방의 디바이스에 대한 정보 뿐만 아니라, 홈서버내의 모든 룸 브리지가 가지고 있는 각 디바이스 정보를 공유하고 관리한다.

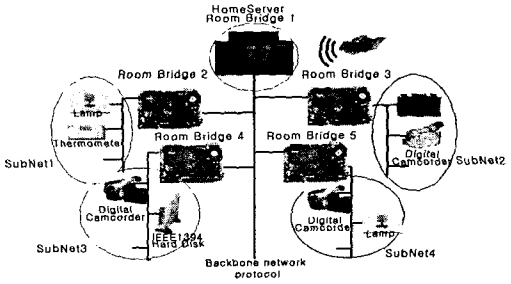

그림 2. 홈 네트웍 하드웨어 구조

3.2 소프트웨어 구성

제안하는 소프트웨어는 전체적으로 홈 서버와 룸서버 그리고 룸서버에 연결되는 각종 디바이스의 등록과 제거를 책임지는 디바이스 관리자가 계층적으로 존재한다.

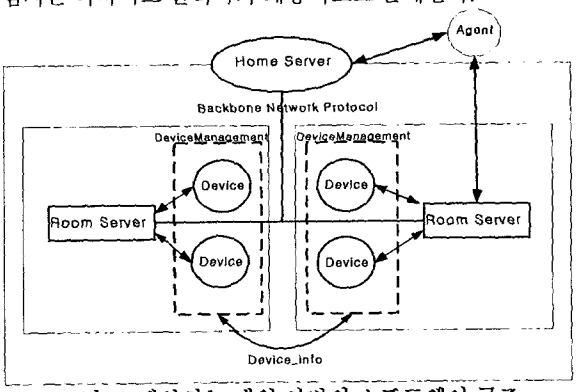

그림 3. 제안하는 네임 서버의 소프트웨어 구조

한 가정에는 하나의 홈서버가 있고 각 방에는 룸브리지에 룸 서버가 하나씩 올라가게 된다. 홈 서버는 홈 내에 존재하는 모든룸 서버들의 위치정보를 저장한다. 각 룸 넘버와 이에 연결되는 IOR을 묶어서 저장함으로써 현재 홈 서버에 몇 개의 룸 서버가 연결되어 있는지를 보여준다. 룸 서버는 활성화 되면서 홈 서버의 위치를 찾아서 자신의 룸넘버와 IOR을 넘겨준 다음에 자신에게 연결되는 디바이스의 정보를 해쉬 테이블을 이용해서 저장하고 관리한다. 하나의 룸 서버가 가지고 있는 디바이스 정보는 자신에게 연결되어 있는 디바이스 정보 뿐만 아니라, 이웃 룸 서버에 연결되어 있는 디바이스 정보도 타이머를 이용한 업데이트를 통해 전달 된다. 따라서 각방에 있는 룸 서버는 홈 서버 내에 연결된 모든 룸 서버에 연결되어 있는 디바이스들의 정보를 모두 가지게 된다. 이렇게 함으로써, 어느 하나의 룸 서버가 다운 됐을 경우라 할 지라도 다른 룸 서버를 통해 홈의 디바이스들을 관리하고 제어할 수 있다.

3.3 추가되는 IDL 정의와 데이터 저장 구조

위에서 언급한 소프트웨어 구조를 바탕으로 코바 애플리케이션을 개발하기 위해서는 우선 IDL을 정의해야 한다. 본 논문에서는 제안한 소프트웨어를 구현하는데 있어서 기존 네이밍 서비스의 NameContext 인터페이스에 정의되어 있는 id와 kind를 이용해 객체를 구분하는 방식을 유지하였으며, 이를 사용함과 동시에 새로운 모듈과 인터페이스를 다음과 같이 정의하였 다.

```
Module HomeServer {
        struct Binding {
                unsigned short RoomNum ;
                string  DeviceName ;    };
        struct Position {
                unsigned short  RoomNum ;
                string   DeviceName ;
                string   IOR ;
                string   IDL ;    } ;
        typedef sequence<Position>position ;
        struct Services {
                sequence<string>  Service ;
                sequence<string>  Status ; } ;
        struct Access {
                Binding binding[2] ;
                unsigned short Priority ; } ;
        struct DIS {
                Position  position ;
                Services service ;
                Access   access ;        } ;
        typedef sequence<DIS> Dev_seq ;
        interface HouseNamingService { };
        interface RoomNamingService { } ;
        interface DeviceManagement { } ;
        interface AgentInterface { } ;
        } ;
  } ;
```

표 1.추가 IDL 정의

위의 추가된 IDL에서는 각 디바이스의 위치,서비스등의 정보와 룸 서버의 위치 정보를 저장하기 위한 구조체를 정의하였고 이들을 사용하여 등록과 제거, 업데이트를 하기위한 오퍼레이션을 각 인터페이스 내부에 정의하였다. 업데이트 오퍼레이션의 경우에는 디바이스 등록시에 파라미터로 사용자가 duration정보를 주도록 하였다. 등록된 객체의 레퍼런스는 유지할 필요가 있지만 서비스와 관련해 유효하지 않은 객체의 정보는 적절하게 update되어야 한다. 따라서 서비스의 유효시간을 업데이트 오퍼레이션의 파라미터로 정의해서 짧은 시간의 서비스를 요하는 서비스는 굳이 삭제 오퍼레이션을 사용하지 않고도 자체적으로 처리할 수 있다. 각 디바이스의 정보를 좀 더 자세하게 정의하면 다음과 같다. 하나의 디바이스는 속해있는 룸 서버의 룸넘버와 디바이스 이름, 그리고 IOR,IDL정보를 저장하고 있는 Position구조체와 서비스에 따른 상태-TV를 예로 들면 power라는 서비스에는 on/off가 있고 volume 이라는 서비스에는 up/down가 있듯이- 를 나타내는 Services 구조체와 현재 서비스중인 디바이스 상태와 그 서비스를 요구한쪽의 두 곳의 정보를 저장하고 그 우선순위에 따라 서비스를 제공하는 Access라는 구조체를 가지고 있다.디바이스 데이터 저장 구조를 간략하게 나타내면 다음표와 같다.

< Device Information Structure Definition >
Position: [[Room # , Device Name] [IOR , IDL]]
Service: [[Service , Status] [... , ...]]
Access: [binding[2] , Access Level or Priority] :

< Device Store Data Structure Definition >

Room #	Device Name	Duration	DIS

표2. 디바이스 데이터 저장 구조

그리고 아래의 시퀀스 다이어 그램은 두개의 룸 서버1,2가 홈 서버에 먼저 홈 서버에 등록한뒤 룸 서버1 에 디바이스가 등록하여 룸 서버1의 자료를 업데이트 한 다음에 홈 서버에서 얻은 연결된 이웃 룸 서버2를 찾아 등록된 디바이스 정보를 넘겨주는 과정을 나타낸 것이다

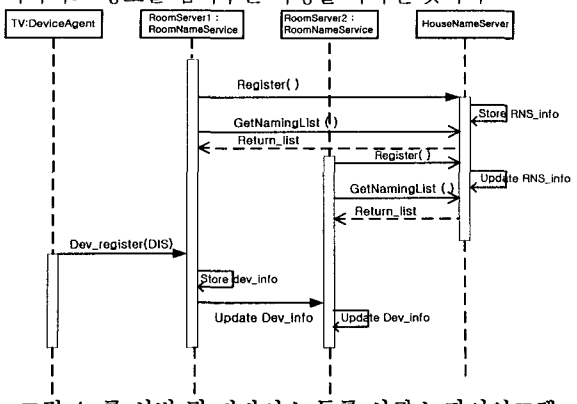

그림 4. 룸 서버 및 디바이스 등록 시퀀스 다이어그램

4. 결론 및 향후 연구 과제

코바를 미들웨어로 한 홈 네트웍환경하에서의 분산 애플리케이션의 확장과 개발을 위해 코바의 서비스 기능은 분산 객체의 동적인 환경에 적합하도록 설계되어야 한다. 본 논문에서는 이를 룸 단위의 계층적 구조를 가지는 홈 내의 각종 디바이스의 상태와 정보의 업데이트를 효율적으로 개선하고 각 디바이스에 대한 구체적 정보를 바탕으로 사용자 정의의 기능을 추가함으로써 홈 내의 제어를 보다 동적인 환경에서 유용하도록 하였다. 제안된 모델은 디바이스 상태 변화시에 각 룸 서버의 update time을 체크하여 테스트 할 수 있도록 자바GUI 환경을 조성중이다.향후의 연구로써 홈 단위를 벗어나 중앙 집중 서버 구조하에서 기업 단위의 분산형 작업을 제어하기 위해 제안한 모델을 적용한 실질적인 네이밍 서비스에 대한 연구가 필요하다.

[참고 문헌]

[1]CORBA Specification v2.3 OMG Document Oct,1999

[2]CORBAservices : Common Object Services Specification, OMG Document 98-12-09

[3] Soon-ju Kang " ROOM-BRIDGE : A Vertically Configurable Network Architecture and Real-Time Middleware for Interoperability between Ubiquitous Consumer Devices in Home" ,Lecture Notes in Computer Science 2218(p.232-251)

[4] Jun-Ho Park " Multi-Agent based Home Network Management System Using Extended Tuple Space Model" , SAM' 02 SCOPE

[5]William Adjie-Winoto " The design and implementation of an intentional naming system" ,Dec SOSP ' 99

[6] Henning Vinoski, Advanced CORBA Programming with C++

[7] Douglas C.Schmidt and Fred Kuhns," An Overview of the Real-time CORBA Specification" ,IEEE Computer special issue on Object Oriented Real-time Distributed Computing , June 2000

[8] Ndds, http://www.rti.com/ , Draft RTPS(Real-Time Publish-Subscribe) Specfication