이동성 관리를 지원하는 경량 홈 네트워크 미들웨어 보안 기술

고 광 만[†] · 현 호 재[††] · 홍 주 희[†††] · 한 선 영[††††]

요 약

다양한 종류의 임베디드 시스템이 폭넓게 사용됨에 따라, 임베디드 정보 가전에 접근하고 제어할 수 있는 홈 네트워크 미들웨어에 관한 연구가 활발히 진행 중이다. 그러나 제한된 저장 공간과 낮은 컴퓨팅 능력은 홈 네트워크 기술을 임베디드 시스템에 적용했을 때 심각한 문제를 발생시킨다. 본 논문에서는 홈 네트워크 미들웨어의 실제 성능을 강화하기 위해 이동성 관리를 지원하는 경량 미들웨어를 제시한다. 이동성 관리는 애니캐스트 기술을 적용하여 구현하였으며, IP 기반의 홈 네트워크는 서비스(디바이스)가 노출되어 있기 때문에 사용자의 신원을 확인하는 인증과 접근에 관한 보안 기술을 제안한다.

키워드 : 경량 홈 네트워크 미들웨어, 정보 가전, 임베디드 시스템, 애니캐스트, 이동성

Lightweight Home Network Middleware Security Mechanism supporting Mobility Management

Kwangman Koh[†] · Hojae Hyun[††] · Juhee Hong[†††] · Sunyoung Han[††††]

ABSTRACT

As various kinds of embedded systems (or devices) become widely available, research on home network middleware which can access and control embedded home appliances are actively being progressed. However, there is a significant problem in applying the home network technology to embedded systems because of their limited storage space and low computing power. In this paper, we present a lightweight middleware for home network on embedded systems. Also, we propose a mechanism for mobility management which adopts the anycast technology.

Key Words : Lightweight Home Network Middleware, Home Appliance, Embedded System, Anycast, Mobility

1. 서 론

디지털 정보가전의 중요성이 대두되며 가정 내 네트워크 상의 모든 기기들에 접근하여 제어할 수 있는 홈 네트워킹 솔루션에 대한 연구가 활발히 진행되고 있다. 그러나 각각 다른 미들웨어와 프로토콜을 사용하며 추가적인 저장용량을 필요로 하기 때문에 자원이 제한적인 장치에 적용하는데 큰 문제점을 갖고 있다.

임베디드 시스템을 이용하는 디지털 가전이 중요하게 됨에 따라, 모든 임베디드 시스템을 관리하기 위한 홈 네트워크 미들웨어에 관한 연구가 활발하게 진행되고 있다[1]. 홈 네트워크 미들웨어의 성능을 강화하기 위해서, 다양한 기술적인 요구는 요즘 이동성과 경량으로 강조 되었다. 이것은 그 실무성능을 증가시키기 위해 고려되어야 할 요소들이다. 그러나 현재 홈 네트워크 미들웨어에서 이동성과 경량성이 고려될 때 문제가 있다.

첫째, 홈 네트워킹 환경 내의 클라이언트 (또는 서비스제공자)는 주로 자원제약을 가지고 있는 내장된 장치이다. 홈 네트워크를 위한 현재의 미들웨어가 특정한 플랫폼과 개발 도구를 필요로 하기 때문에, 임베디드 시스템에 적용하기에는 충분한 자원을 가지고 있지 않는 문제점들을 지니고 있다. 임베디드 리눅스 기반의 홈 네트워크를 관리하고 제어하기 위해, 리눅스의 기본 기능만을 사용하는 경량 미들웨어를 사용할 필요가 절대적이다.

둘째, 홈 네트워크 보안을 위하여 현재 해결해야 할 가장 큰 문제는 홈 네트워크 보안을 위한 기본 프레임워크 문제, 인증 및 접근 문제, 디바이스 인증서 필요성 및 프로파일 문제, 프라이버시 보호 문제 이다. 현재 홈 네트워크에서 대두되고 있는 보안문제는 서비스를 요구하는 홈 디바이스의 신원과 이들 디바이스에게 허용 가능한 서비스의 유형을 어떻게 확인하느냐이다. 이 같은 문제를 해결하기 위해 홈 네

※ 이 논문 또는 저서는 2005년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(KRF-2005-041-D00768).
† 준 회 원 : 건국대학교 대학원 컴퓨터공학과 박사과정
†† 준 회 원 : 한국정보보호진흥원 정보보호기술단 위촉연구원
††† 정 회 원 : 건국대학교 CST 선임컨설턴트
†††† 정 회 원 : 건국대학교 정보통신대학원 원장
논문접수 : 2005년 7월 22일, 심사완료 : 2006년 3월 31일

트워크에서 오디오 및 비디오 서비스를 제공하기 위한 서비스 시나리오가 필요하며, 각 시나리오에서 나타날 수 있는 보안 위협들을 도출하고, 이 같은 위협을 효과적으로 대처하기 위한 다양한 보안 서비스들을 찾아내 이를 근거로 홈 네트워크 보안을 위한 기본 프레임워크를 만들어야 한다.

위의 문제들을 극복하기 위해, 본 논문에서는 임베디드 리눅스의 제한된 자원을 가진 최적화된 LHNM(Lightweight Home Network Middleware)의 구조를 설계하고 구현했다. 이 시스템은 IPv6 스펙에 처음 제시된 애니캐스트 기술을 적용한 이동성 관리 메커니즘을 제안한다[2-4]. 그리고 클라이언트는 가장 가까운 룩업 서버를 선택하고 접속하기 위해 잘 알려진 애니캐스트 주소를 이용한다. 효율적인 룩업 서버발견은 애니캐스트 리졸버에 의해 수행된다. 각 가정의 디바이스는 사용자와 서비스 간의 ACL(Access Control List)을 통해서 인증받고, 룩업 서버는 ACL 자원을 통해서 모든 서비스들에 대한 접근을 제어한다.

불법 디바이스의 사용을 방지하기 위해서는 홈 네트워크의 구성요소인 디바이스 자체에 대한 인증과정이 필요하다. 디바이스 마다 부여된 Security ID로 디바이스 인증이 이루어지고 있다. 홈 네트워크에서는 디바이스 인증 외에 디바이스를 사용하는 사람의 신원확인을 위한 사용자 인증 기능도 반드시 필요하다. 또한, 기기간의 인증이 필요하다. 원활한 홈 서비스 제공을 위해서는 기본적으로 홈 네트워크 구성요소 간의 자원공유를 위한 신뢰가 확보되어야 한다[5]. 이를 위해서는 구성요소간의 기기간 상호인증이 필요하다. ACL를 이용하여 클라이언트가 서비스를 선택하여 이용할 때 접근 권한자원을 받기 위해 사용되고 있다. 이렇게 기기간의 상호 인증을 받을 때 ACL를 이용한다.

본 논문의 구성은 다음과 같다. 2장은 RPC 대해서 설명한다. 3장에서는 LHNM의 구조와 ACL 구조를 설명하고, 4장은 LHNM의 동작 개요와 ACL를 사용한 기기간의 인증 및 접근에 대해 설명한다. 5장은 구현 결과와 홈 네트워크 미들웨어와 LHNM비교 설명하고, 6장에서는 본 논문의 결론을 맺는다.

2. 관련 연구

2.1 RPC(Remote Procedure Call)

분산 처리 환경에서는 서버 응용 프로그램과 각 서버에 접근하여 작업을 요청하는 클라이언트 프로그램을 필요로 한다. 클라이언트/서버 모델에서 소켓 프로그래밍을 이용하여 프로그램을 개발 할 수도 있지만, 이와 같은 경우에 개발자는 통신 기능, 프로토콜, 그리고 다양한 작업들을 직접 작성해야 하는 부담이 있으며 프로그램의 이식성에도 한계가 있다. 이러한 문제점을 해결하기 위하여 더욱 용이한 클라이언트/서버 모델 프로그램을 개발할 수 있도록 지원해 주는 것이 RPC (Remote Procedure Call)다[6].

3. 시스템 구조

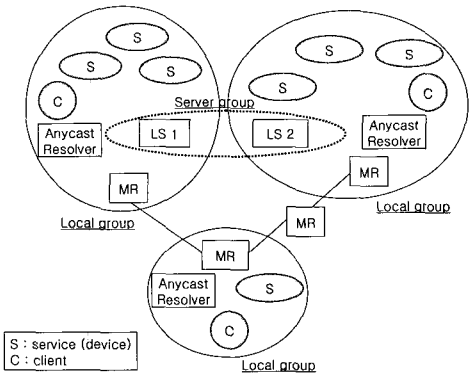
본 논문에서 제안한 LHNM 시스템에서 사용되는 용어는 <표 1>과 같이 표기한다.

〈표 1〉 LHNM 시스템 용어

LS: 룩업서버 G: 멀티캐스트 그룹/주소 Ga: 애니캐스트 그룹/모든 LS들에 의해 등록된 주소 MR: 멀티캐스트 라우터

3.1 LHNM의 구조

(그림 1)은 LHNM의 시스템 구조이다. 시스템은 LS, 클라이언트, 서비스(디바이스), 애니캐스트 리졸버와 멀티캐스트 라우터로 구성된다.



(그림 1) 전체 LHNM 시스템 구조

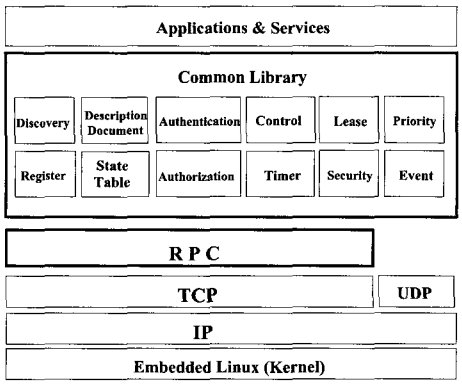
서버그룹을 형성하기 위해 멀티캐스트 주소 와 애니캐스트 주소가 LS 그룹에 주어진다. 서버 그룹의 모든 LS는 애니캐스트 그룹에 참여해야 한다. 그룹 Ga에 참여한 LS는 클라이언트 또는 서버 제공자의 요청을 수락하기 위해 인터페이스들 중 한가지를 구성할 수 있다. 본 논문에서는 클라이언트가 룩업 서비스를 제공하는 LS를 위해 Ga의 애니캐스트 주소를 안다고 가정한다[7]. 애니캐스트 리졸버는 클라이언트에게 가장 가까운 LS의 유니캐스트 주소를 보내준다. 또한 모든 LS들은 멀티캐스트 그룹에 반드시 가입을 해야 한다. 이는 각 LS에 의해 유지되는 서비스 항목을 업데이트 하는데 사용된다. 서버 그룹의 각 LS는 그 그룹의 다른 LS와 항목을 공유하기 위해 그 그룹의 업데이트 된 서비스 항목을 멀티캐스트 한다. 제안된 구조는 이질적인 플랫폼에서 적용가능하고 LS 탐색 시간을 줄일 수 있으며 전체 시스템 성능을 향상시킬 수 있다.

3.2 LHNM 프로토콜 스택

(그림 2)는 LHNM 프로토콜 스택과 라이브러리 모듈을 나타낸 것이다.

클라이언트가 상태 테이블의 등록과 업데이트를 LS에 서비스 요청할 수 있도록 여러 기능들이 LHNM 공통 라이브러리에 제공된다. 이로써, 그 클라이언트는 RPC를 통해 불러올 수 있다. 이 기능들은 다음과 같다:

- Discovery 모듈 : 클라이언트는 ERS(Expanding Ring Search) 기술과 응용 계층 애니캐스트를 통해 가장 가까



(그림 2) LHNM 프로토콜 스택

운 LS를 찾을 수 있다[8]. LS는 또한 Discovery_LS요구에 개별 IP주소로 응답할 것이다.

Register 모듈 : LS가 발견된 후, 서비스 공급자는 LS에 등록할 때 사용 되는 모듈이다.

Description document 모듈 : 서비스의 종류 및 기능 등의 정보를 표시하기 위한 모듈이다.

State table 모듈 : 서비스의 상태정보를 저장하기 위한 모듈이다.

Authentication 모듈 : 사용자의 신원을 확인하는 모듈이다.

Authorization 모듈 : 인증이 완료된 후 사용자에게 어떤 서비스가 제공가능하고 어떤 서비스에 대한 접근이 가능한지를 판단하는 모듈이다.

Control 모듈 : 각종 기능을 제어하는 모듈인데, Lease, Security, Priority, Event등을 제어하며 관리 한다.

Timer 모듈 : Active alarm timer 는 LS가 움직임과 결함을 찾아낸다.

Lease 모듈 : LHNM시스템 환경에 있는 많은 서비스에 접근하는 것은 lease에 기반한다. Lease는 보증된 시간 이상 접근을 허가한다. 각 lease는 클라이언트와 서비스 사이에서 협상한다. 만일 자원이 더 이상 필요치 않거나 그 클라이언트(또는 네트워크)가 실패한다면, 갱신 요청이 받아들여지지 않을 것이다. 그 lease기한이 끝난다면, 그 서비스는 더 접근할 수 있지 않다. 그리고, LS는 자동적으로 그 서비스를 재등록 하게 된다.

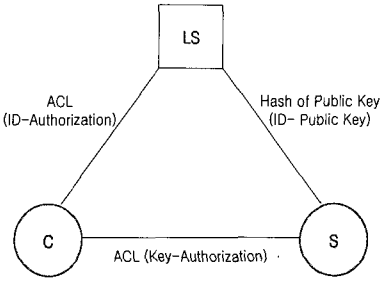
Security 모듈 : 클라이언트의 요구를 받자마자, LS는 ACL를 검사하고 그 요구를 수용할지 거부할지 결정한다.

Priority 모듈 : 작업 우선권을 제어한다. 임무 기한에 맞추고, 실시간 서비스 예측가능성을 높이기 위해 LHNM은 그 우선권에 따라 모든 작업일정을 만들다. 또한, 낮은 우선 임무의 선점은 더 높은 우선권 작업의 데드라인 요구를 충족시키기 위해 필요하다.

Event 모듈 : Event의 전송과 처리는 서비스 공급자에 의해 사용된다. 서비스 공급자는 클라이언트가 서비스 이벤트 등록을 허락할 수 있고 그런 이벤트의 발생의 통지를 받을 수 있다.

3.3 ACL구조

(그림 3)은 서비스를 LS에 등록 하면서 ID-Public key를 LS에서 받는다. 클라이언트는 LS에 사용자 증명을 위해 ID와 암호를 보내고 클라이언트는 LS로부터 등록되어 있는 서비스 리스트를 받는다. 클라이언트는 서비스 리스트를 보고 자신이 원하는 서비스를 선택 하면 LS는 ACL에서 ID-Authorization를 준다. 클라이언트는 LS로부터 받은 ID-Authorization를 가지고 선택한 서비스의 Key-Authorization값과 비교해서 권한을 부여 할지를 결정한다.



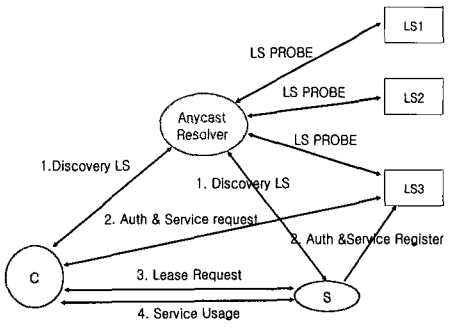
(그림 3) ACL 동작구조

ACL에 의해 제공되는 중요 기능은 다음과 같다.

- 발급자(Issuer) : 인증서 공표와 서명의 주체를 의미하고 발급자의 공개키 혹은 키의 해쉬값이 될 수 있다.
- 주체(Subject) : 주어진 인증서에 의해 권한을 얻은 개체이다. 주체필드에는 공개키나 키의 해쉬값, 또는 이름을 기록한다.
- 위임 (Delegation) : 발급자가 주체에게 재위임 권한을 부여했을 때 지정한다. 부울 값으로 표현하고 위임 허용시 "true"로 설정한다.
- 권한(Authorization) : 접근권한을 의미하며, 권한은 인증서 발급자에 의해 자유롭게 정의된다.
- 유효기간(Validity Dates) : 발급자에 의한 인증서의 유효 시점을 말한다.

4. LHNM 동작 메커니즘

(그림 4)는 LHNM의 동작과정을 나타낸 것으로 룩업 서버, 클라이언트, 서비스의 동작 절차를 통해 서버 선택 알고리즘과 이동성 관리를 보여 준다.



(그림 4) LHNM 동작

4.1 Discovery LS

클라이언트는 ERS 기술을 이용하여 로컬 LS를 찾을 수 있다. 그 클라이언트는 TTL(time-to-live)값을 가지고 그룹 주소로 FIND_localLS 패킷을 다중 전송한다. 그 다음, 클라이언트가 해당 REPLY_localLS 패킷을 받으면, 그 패킷에서 해당 로컬 LS의 IP주소를 얻는다. 만약 응답이 없으면 다음과 같이 애니캐스트 주소를 명시하여 다중 LS로부터 선택된 LS를 가지고 통신할 수 있다.

1. 클라이언트는 애니캐스트 리졸버에 애니캐스트 주소와 함께 LS를 발견하기 위한 요청을 보낸다.
2. 리졸버는 4.6항목에 나와있는 서버 선택 알고리즘을 이용하여 어떤 LS가 그 순간 가장 적합한지 결정한다.
3. 클라이언트와 선택된 서버 사이의 연결이 새로 만들어 진다.

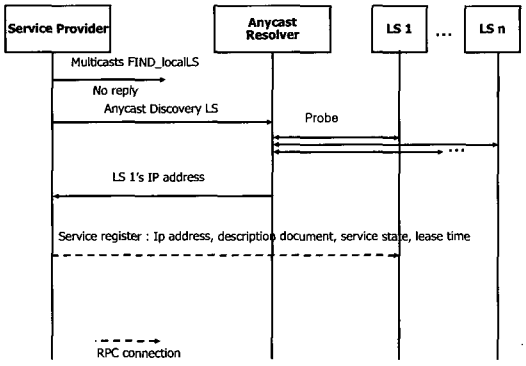
4.2 룩업 서버

서비스와 클라이언트를 찾거나 등록 할 때 RPC를 이용한다. LS 처리과정은 RPC 서비스를 실행하는 주 스레드와 클라이언트의 Discovery_LS를 이용해서 LS를 발견하고 또는 애니캐스트 리졸버의 요청을 받는 스레드로 구성된다. Discovery_LS 모듈을 사용함으로써 적절한 LS를 찾고 Register모듈을 사용함으로써 LS에 서비스 리스트를 등록한다. LS에 의해 제공되는 중요한 기능은 다음과 같다.

- Add device: 리스트에 대한 서비스를 등록
- Remove device: 리스트에서 서비스를 제거.
- Remove all device: 리스트에서 모든 서비스를 제거.
- Update SST: 서비스 상태 테이블을 갱신
- GetDeviceInfo: 서비스의 정보 되돌려주기
- ConfirmAuth: 인증의 결과
- Alivealarm: 클라이언트에 살아있음을 통지
- ReplicatesLS: 속하는 서버 그룹에 갱신된 서비스의 리스트를 멀티캐스트
- ReplylocalLs: LS IP 주소로 응답

4.3 서비스

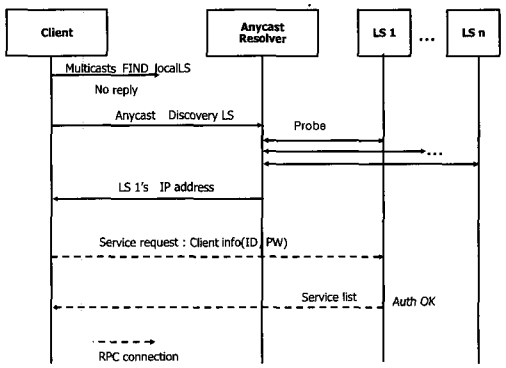
LHNM 구조 개념 중 가장 중요한 것은 서비스이다. 서비스는 다른 서비스를 이용할 수 있다. LHNM 시스템은 서비스가 더하여지거나 또는 언제든지 LS에서 빠질 수 있다. 서비스가 작동 중이라면 Discovery_LS 모듈 (그림 5)에서 DiscoveryLookup 함수를 이용해 LS를 발견하고 등록한다.



(그림 5) 서비스 등록

4.4 클라이언트

클라이언트가 Discovery_LS 모듈을 발견한 이후 애니캐스트 리졸버로부터 LS의 IP 주소를 받는다. 클라이언트는 LS에 사용자 증명을 위해 ID와 암호를 보내고 LS으로부터 서비스 리스트를 받는다. 클라이언트는 서비스 리스트 (그림 6)에서의 요청된 서비스를 선택한다. 서비스에 대해 lease 시간을 협상한 뒤에 클라이언트는 RPC를 통해 해당 서비스를 요청한다.



(그림 6) 클라이언트의 서비스 검색

4.5 이동성 관리 기능

애니캐스트 주소는 인터넷 안에 분산된 서버 그룹을 나타내는 데 이용된다. 중복된 서버는 개별 IP 주소 대신 애니캐스트 주소로 광고한다[9, 10].

4.5.1 룩업 서버 이동성

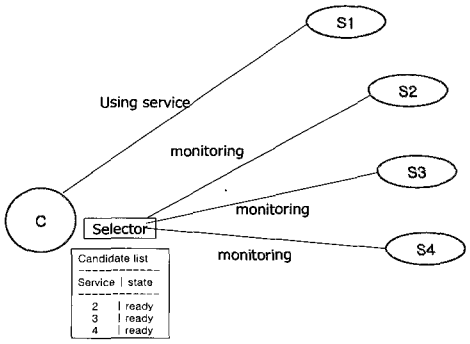
LS는 로컬 멤버에 자신이 살아있음을 알리기 위해 정기적으로 메시지를 보낸다. 클라이언트가 alive alarm timer에 의해 LS의 움직임을 탐지한다면, Discovery_LS 작동을 시작한다. 그 다음, 클라이언트는 가장 가까운 대체 룩업 서버에 연결한다.

4.5.2 서비스 이동성

서비스는 lease 기술을 이용해 자신의 존재를 알려줄 수 있다. 그러나 LS는 lease 시간 동안 정확한 서비스 정보를 가지고 있지 않을 수도 있다. 서비스는 lease 시간이 끝나기 전에 자신의 이벤트 청취자 (클라이언트 또는 기타 서비스)에게 존재를 정기적으로 알려준다. 서비스가 물러간다고 가정함에 따라, 이벤트 청취자가 지정된 시간 내에 서비스로부터 이벤트를 받지 못한다면, 청취자는 서비스의 활성화 유무를 확인하기 위해 서비스에 대한 질문 메시지를 보낸다. 서비스가 어떤 응답도 하지 않는다면, 청취자는 서비스의 부재를 인식한다. 청취자는 LS에게 이 정보를 알린다. LS가 이것을 받으면 서비스 리스트를 업데이트 하고 다른 LS에 멀티캐스트한다. 클라이언트는 최상품으로 그 기능을

제공하고 모든 리스트를 유지하는 서비스를 선택한다. 클라이언트가 가장 좋은 서비스 공급자로부터 서비스의 설비를 제공받는 동안 선택자는 다른 서비스의 상태를 모니터한다.

(그림 7)에서 연결된 서비스 S1이 물러간다면, 그 클라이언트는 더 이상 그 서비스를 사용할 수 없다. 이 경우, 클라이언트는 리스트에서의 서비스 중의 하나를 골라서 선택된 서비스를 활성화시킨다.



(그림 7) 클라이언트와 서비스 이동 선택 동작

4.5.3 클라이언트 이동성

각각의 클라이언트는 그들의 ID와 IP 주소 그리고 State 정보를 가지고 있고, Selector에 의해 다음 서비스의 리스트를 관리한다. 이동 클라이언트가 또 다른 링크로 이동한다면, 다른 LS로부터 새롭게 갱신된 서비스 리스트를 받기 위해 Discovery_LS 연산을 수행해야 한다. 그 클라이언트가 서비스 리스트를 받는다면, Selector는 그 현존하는 연결을 멈추고 더 좋은 서비스를 위해 새로운 연결을 설정한다.

4.6 서버 선택 알고리즘에 기초한 클라이언트 요구

인터넷 클라이언트가 애니캐스트로 질문을 하게 되면, 애니캐스트 리졸버는 반드시 그 클라이언트에게 그 질문에 가장 좋은 것이 애니캐스트 그룹 중 어느 LS인지 말해야 한다. 본 논문에서 이 문제를 처리하기 위해 서버 선택 알고리즘에 기초한 클라이언트 요구라 불리는 알고리즘을 제시한다.

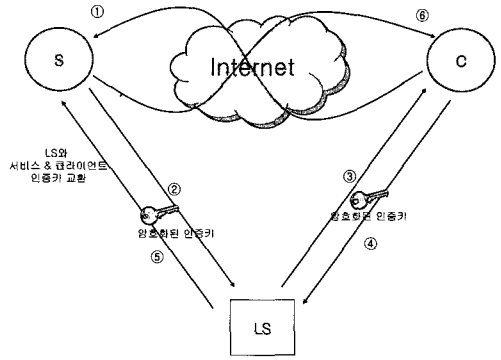
서버 선택 알고리즘에 기초한 그 클라이언트 요청 절차는 다음과 같다:

1. 클라이언트가 로컬 LS을 찾는 데 실패한다면, 애니캐스트 리졸버에 애니캐스트 주소와 관련된 애니캐스트 질문을 보낸다.
2. 그 질문은 애니캐스트 리졸버에 도달한다.
3. 애니캐스트 리졸버는 애니캐스트 그룹 내에 모든 LS에 패킷을 보낸다.
4. 애니캐스트 리졸버는 참가 LS를 나타냄으로 우선 응답하고, 클라이언트 라우팅 경로 및 참가한 LS의 유니캐스트 IP주소를 보낸다. 애니캐스트 리졸버는 다른 LS로부터 나머지 응답을 버린다.
5. 클라이언트는 RPC 기술을 사용함으로써 받아들여진 정보에 따라서 연결을 설정한다.

서버 선택 알고리즘을 수행한 후, 면밀히 조사된 LS는 핑과 같은 조사요구에 응답해야 한다. 서버 응답 성능이 낮다면, 응답 성능이 더 좋은 서버보다 더 늦게 될 것이다. 동시에 그 리졸버에 대한 응답은 네트워크 실행에 대한 정보를 포함한다. 이 알고리즘은 [11]에 기술된 알고리즘에 기반한다.

4.7 ACL를 사용한 기기간의 인증 메커니즘

(그림 8)은 기기간의 인증 동작 절차를 나타내고 있다.



(그림 8) 기기간 인증 메커니즘 동작 절차

ACL를 기초로 기기간 인증 메커니즘 동작 과정은 다음과 같다.

1. 클라이언트는 서비스에게 해쉬함수로 JobID와 서비스주소를 키를 만들어 암호화하여 서비스를 요청한다.
2. 서비스 요청을 받은 서비스는 행동을 하기 전에 LS로 연결하고 요청한 클라이언트에 대한 서비스를 수행해도 되는지 알기 위해 생성된 JobID 확인 요청을 한다.
3. LS는 서비스가 넘겨준 암호화된 JobID를 복호화하여 보고 클라이언트에게 보낸다.
4. 클라이언트는 받은 JobID에 대한 해쉬값을 생성하여 다시 LS에게 전송한다.
5. 암호화된 키 값을 받은 LS는 자신이 가지고 있는 해쉬값을 생성하여 동일한지 비교한다. 그리고 클라이언트에게 재확인 요구를 수행하고 클라이언트로부터 재확인 정보를 서비스에게 보낸다.
6. 서비스는 받은 동일한 정보를 통해 서비스 수행 여부에 대하여 지정된 동작을 취하게 된다.

5. LHNM 구현

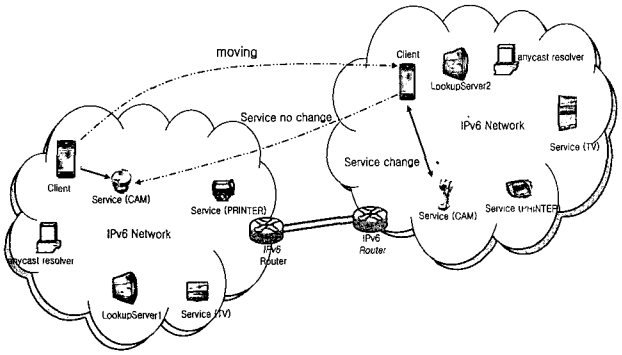
리눅스에서 기본으로 설정이 되어 있는 RPC를 이용 LHNM 기본 프로토콜 스택을 사용하였다.

5.1 구현 환경

LHNM과 이동성 및 관리를 테스트하기 위해 (그림9)과 같이 2개의 다른 네트워크를 구성하였다. 그리고 2대의 룩업 서버와 6개의 서비스 그리고 1대의 클라이언트를 두어

테스트 베드를 구성했다. 1대의 클라이언트가 Cam 서비스를 사용하다가 다른 곳으로 이동 했을 때 Cam 서비스를 계속 사용 하거나 다른 서비스의 Cam을 사용할 수 있도록 테스트 베드를 구성하여 테스트 했다. 사용한 운영체제는 임베디드 리눅스 시스템과 리눅스 시스템을 사용하였다.

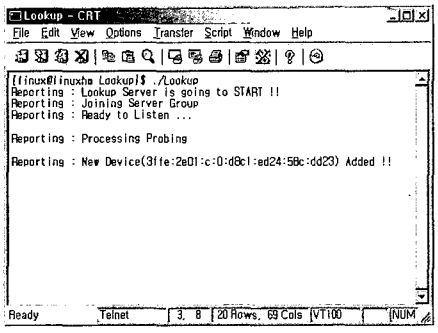
룩업 서버는 Red Hat 9.0 (kernel 2.4.20-8)를 사용 했으며, (PDA)는 임베디드 리눅스 Qt(qt-embedded2.3.1)를 이용하여 PDA에 GUI 환경(그림 11, 12)을 만들었다.



(그림 9) LHNM 테스트 환경

5.2 룩업 서버 실행

(그림 10)은 화면에 LS의 실행을 보여준다. 만일 LS가 시작되면, 이는 서버 그룹에 참여하고 Discovery 함수를 이용하여 요구를 듣는다. 새로운 서비스가 있으면 LS에 등록한다. LS에 등록 하면서 자신의 주소를 등록한다.

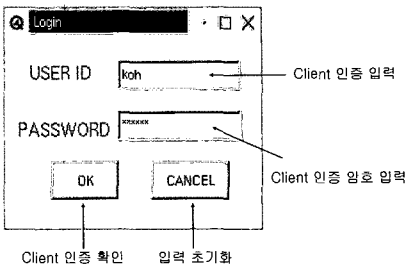


(그림 10) 룩업 서버 실행

5.3 클라이언트 실행

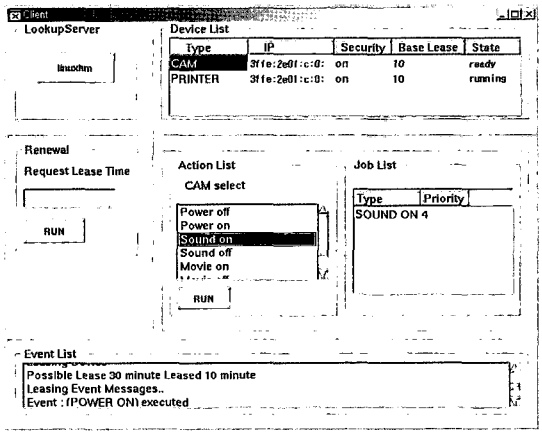
클라이언트의 GUI부분은 Qt로 쓰여져 모든 관련된 정보를 (그림 11, 12)에서 볼 수 있는 것처럼 화면에 표시한다.

(그림 11)은 클라이언트가 LS에 등록을 할 때 클라이언트는 인증을 받기 위해 ID와 암호를 입력하여 LS에 등록을 한다. 클라이언트가 보낸 ID와 암호로 로그인을 요청하면 Authentication는 DB에 저장된 ACL리스트에서 일치하는 ID와 암호가 있는지 검색한다. 로그인 과정이 성공적으로 수행되면 서비스 리스트를 볼 수 있는 권한이 있다. 그리고 ACL를 기초로 기기간 인증 메커니즘을 이용하여 클라이언트는 서비스에게 해쉬함수로 JobID와 서비스주소를 키를 만들어 암호화하여 서비스를 요청한다.



(그림 11) 클라이언트 인증 실행

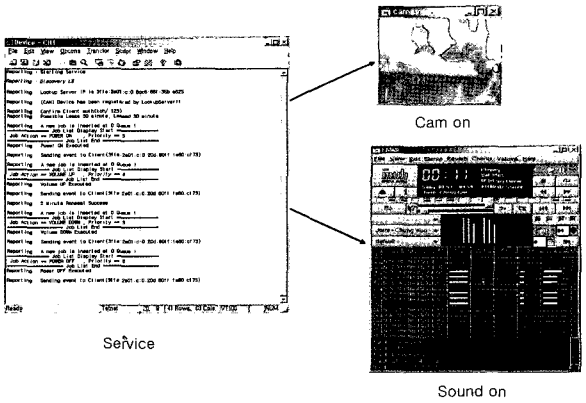
클라이언트 ID와 암호가 ACL리스트와 일치하여 로그인 과정이 성공하면 (그림 12)을 볼 수 있다. LS(linuxhm) 버튼을 누르면 룩업 서버에 등록되어 있는 서비스의 정보를 Device List화면에 보여 준다. CAM이라는 서비스를 선택하면 기본 lease 값을 협상한다. 협상 되었으면 서비스 작업 리스트 중 원하는 작업을 선택하여 실행시킨다.



(그림 12) 클라이언트 동작

5.4 서비스 실행

가장 가까운 LS에 RPC 기능을 사용하여 정보를 등록 한 후 discovery_LS 스레드와 RPC 서비스를 실행하는 주 스레드로 구성된다. 주 스레드는 클라이언트에게 DoAction RPC 인터페이스를 제공한다.



(그림 13) 서비스 실행

클라이언트 ID와 암호가 ACL리스트와 일치하여 로그인 과정이 성공하였다고 서비스에게 내용을 알려 준다. 선택된 서비스는 클라이언트가 "Cam on" 그리고 "Sound on" 명령을 (그림 13)와 같이 보낼 때 서비스 JobID와 클라이언트가 가지고 있는 JobID를 비교하고 맞으면 작동한다. 이 작동은 인증과 lease 시간 체크 과정을 통한 후 수행된다.

5.5 홈 네트워크 미들웨어 솔루션 과 LHNM 비교

현재 다양한 홈 네트워크 미들웨어 중에서 세인의 관심을 가장 많이 받고 있는 솔루션으로는 Jini, UPnP 및 HAVi를 꼽을 수 있다. 이 중에서 Jini는 Sun Microsystems에서 제안한 것으로서, IP 네트워크망과 Java 2의 RMI 기술을 기반으로 개발되었다[12]. 현재 표준 스펙은 2.1까지 개발되어 있으며 Home Networking Gateway의 표준인 OSGi(Open System Gateway Initiative)의 근간을 이루고 있다. UPnP는 Microsoft 사에서 제안한 것으로서 IP 네트워크망과 XML, Soap 기술을 기반으로 개발된 Windows Platform용 Home Networking Middleware Solution 이다[13]. 다음으로 HAVi는 Home Audio Video Interoperability의 약자로서, 소니, 필립스, 도시바, 톰슨 등의 AV업계가 디지털 AV기기간의 정보 교환을 목적으로 제안한 Home Networking Middleware Solution 이다[14].

대표적인 홈 네트워크 미들웨어와 LHNM 비교 정리하면 다음 <표 2>과 같다.

〈표 2〉 홈네트워크 미들웨어와 LHNM 비교

	Jini	UPnP	HAVi	LHNM
Network	IP Network	IP Network	IEEE 1394	IP Network
S/W	Java 2, RMI	HTTP, HTML, XML	Object-Oriented	RPC
운영방식	C/S	C/S	P2P	C/S
취약분야	Stream, 인증	Stream 처리	IP Network 기반 접속	없음
Standard	Jini 2.1	UPnP 1.0	HAVi 1.1	LHNM 1.0

현재 가장 각광 받고 있는 Jini는 Java 2 Runtime Environment에 기반하고 있으며, UPnP는 Windows CE 이상에 기반하기 때문에 실제 미들웨어가 제대로 동작하기 위해서는 최소 20Mbyte 이상의 저장용량이 요구된다. 따라서, 제한적인 자원을 갖고 있는 임베디드 장치에 실제 적용하기에는 많은 문제점을 갖고 있다. 또한, 실시간 임베디드 리눅스가 다양한 디지털 정보가전기기를 목적으로 하고 있음에도 불구하고, 현재 이러한 환경에서 작동하는 홈 네트워킹용 미들웨어 솔루션은 전무한 상황이다. 본 논문에서는 임베디드 리눅스에서 제공하는 RPC를 기반으로 하여 미들웨어의 경량화를 최우선적으로 고려하여 설계하였다. 우선 이동성과 사용의 편의성을 고려하여 서비스 리스트를 LS에서 관리하도록 하였다.

6. 결론 및 향후 연구

현재 홈 네트워크 미들웨어가 특정한 플랫폼과 개발 툴을 요구하기 때문에 충분한 자원을 가지고 있지 않은 임베디드 시스템에 적용될 때 여러 가지 문제를 가지고 있다. 임베디드 리눅스의 홈 네트워크를 유지하고 제어하기 위해 리눅스의 기본 기능만을 사용하는 경량의 미들웨어를 사용할 필요가 있다.

본 논문에서는 홈 네트워크를 위한 LHNM을 설계하고 구현했다. 이 미들웨어는 소켓 API 및 RPC 기술을 이용하여 구현되었는데 이로써 제한된 자원을 가진 특별한 플랫폼이나 개발 도구가 필요하지 않은 임베디드 리눅스에서 운영될 수 있다.

또한, 홈 네트워크에 있는 이동성이 홈 네트워크 미들웨어에 몇 가지 문제점을 야기했고 따라서 우리는 이동 네트워크 내의 LHNM의 이동성 관리 기술을 제안한다. 이동성 관리를 지원하기 위해 응용 계층 애니캐스트 기술이 적용된다. 그 응용 계층 애니캐스트 접근법은 서버 이동성, 측정가능성 그리고 오류 허용을 설명하는 클라이언트의 요구를 토대로 한 서버 선택 솔루션을 제공한다. 이 미들웨어는 디지털 가전에 효율적인 이동성 관리 기술을 제공한다. 그리고 각 가정의 디바이스가 불법적으로 사용되는 것을 방지하기 위해, 사용자의 신원확인을 위한 사용자 인증기능 및 서비스의 ID를 LS에서 ACL자원을 통해서 서비스에게 접근제어할 수 있는 솔루션을 제공한다.

본 논문의 향후 연구가 네트워크 이동성을 지원하기 위한 배경 인식과 이동 기술에까지 서비스 선택 기술을 확장할 것으로 기대한다[15, 16].

참 고 문 헌

[1] E.A. Lee, "What's Ahead for Embedded Soft-ware?," IEEE Computer, pp.18~26, Sept., 2000.

[2] S. Deering and R. Hinden, "Internet Protocol Ver-sion 6 (IPv6) Specification," RFC 1883, December 1995.

[3] S. Weber and L. Cheng, "A Survey of Anycast in IPv6 Networks," IEEE Communications Magazine, Vol.42, No.1, pp.127~132, Jan., 2004.

[4] R. Engel, V. Peris, and D. Saha., "Using IP Anycast for Load Distribution and Server Location," Proc. Third Global Internet Conference, 1998.

[5] 한종욱, 김도우, 주홍일, 이윤경, 남택용, 장종수, "안전한 홈네트워크 구축을 위한 보안요구사항", 정보처리학회지, 제11권 제3호, pp.38~44, 2004.

[6] Sun Microsystems, "RPC: Remote Procedure Call Protocol Specification," RFC 1050, April 1988.

[7] K. Arnold, B. O'Sullivan, R. Scheifler, J. Waldo, and A. Wollrath, The Jini Specification, Addison-Wesley, 1999.

[8] D. Boggs, "Internet Broadcasting," XEROX Palo Alto Research Center, Technical Report CSL-83-3, 1983.
[9] M. Yamamoto, et al., "Server Load Balancing with Network Support: Active Anycast," Proc. the Second International Working Conference on Active Networks (IWAN), 2000, Tokyo, Japan, pp.371~384.
[10] S. Deering, "Host Extensions for IP Multicast-ing," RFC 1112, August 1989.
[11] W. Jia and W. Zhou, "Efficient Algorithm for Mobile Multicast using Anycast Group," IEEE Proc.-Commun., Vol.148, No.1, February 2001.
[12] http://www.jini.org/
[13] http://www.upnp.org/
[14] http://www.havi.org/
[15] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research," Technical Report, TR2000-381, Department of Computer Science, 2000.
[16] T. Ernst, "Network Mobility support Goals and Requirements," Internet Draft, 2003.

고 광 만

e-mail : koh@cclab.konkuk.ac.kr
2001년 배재대학교 컴퓨터공학과(공학학사)
2003년 건국대학교 컴퓨터공학과(공학석사)
2003년~현재 건국대학교 컴퓨터공학과 박사과정
관심분야 : 홈 네트워크 미들웨어, 멀티캐스트 보안, overlay 멀티캐스트 등

현 호 재

e-mail : hjhyun@cclab.konkuk.ac.kr
1999년 건국대학교 컴퓨터공학과(공학석사)
2005년 건국대학교 컴퓨터공학과(공학박사)
2005년~현재 한국정보보호진흥원 정보보호 기술단 위촉연구원
관심분야 : 멀티캐스트 보안, IPv6 보안, 홈 네트워크 미들웨어, QoS 등

홍 주 희

e-mail : hongjuhee@paran.com
1996년 강원대학교 전자계산학과(이학석사)
2003년 건국대학교 컴퓨터공학과(공학박사)
2004년~2005년 건국대학교 인터넷미디어학부 강의교수
관심분야 : 응용레벨의 QoS, 차세대 네트워크, overlay 멀티캐스트 등

한 선 영

e-mail : syhan@cclab.konkuk.ac.kr
1977년 서울대학교 계산통계학과(학사)
1979년 한국과학기술원 전산학(석사)
1988년 한국과학기술원 전산학(박사)
1981년~현재 건국대학교 컴퓨터정보통신학과 교수
1998년~1999년 미국 Maryland 컴퓨터과학과 객원교수
2004년~현재 건국대학교 정보통신대학 학장
2005년~현재 건국대학교 정보통신대학원 원장
관심분야 : overlay 멀티캐스트, 멀티캐스트 보안, QoS, 차세대 네트워크 등