U-Healthcare 서비스를 위한 홈게이트웨이 표준안 모델 연구

유진근○ 서대영
한국산업기술대학교 컴퓨터공학과
yjk760○@kpu.ac.kr, seody@kpu.ac.kr

A Study on Standardization of Home Gateway for U-Healthcare Services

Jinkeun Yu○ Daeyoung Seo
Dept. of Computer Engineering, Korea Polytechnic Univ.

요 약

본 연구에서는 가정내 환자가 Zigbee 등의 센서로 측정한 생체정보들을 통하여 U-Healthcare 서비스를 하려고 할 때 고려하여야 하는 홈네트워크의 표준화를 위하여 홈네트워크 미들웨어들의 동향을 간단히 알아보고 그에 준하여 홈게이트웨이의 표준을 연구하고 최적 모델을 도출하여 그에 따라 프로토타입을 설계 후 구현하였다.

1. 서 론

출산율 저하와 평균수명 연장으로 인하여 고령화 사회로 가는 세계적 추세에서 특히 우리나라는 그 속도가 빨라 대책이 시급하다. 많은 노인성 질병환자가 늘어날 것인데 비하여 의료인의 숫자는 그만큼 늘어나지 않을 것이고 그로 인하여 지속적인 관리를 해주기 위한 방법 중 하나로 u-Health 서비스가 각광 받을 전망이다. 자가 관리가 어려워 전문 메디컬 센터의 전문 의료인의 도움을 받아 관리를 하기 위한 모델이 필요한데, 관리 서비스를 위한 표준화된 정보교환 모델이 필요하다. 따라서 환자가 병원에 내원하는 빈도수를 줄이면서 가정에서 벌어지고 있는 여러 요소들을 평가하고 대응할 수 있는 시스템이 절실한 실정이며, 유무선 통신 기술을 이용한 U-Healthcare, 특히 Home Healthcare의 개념이 대두되고 있다.

본 연구에서는 U-Healthcare 서비스의 목적으로 ZigBee [1] 센서들을 통하여 수집되는 환자의 혈당, 심전도, 체성분등 정보들을 게이트웨이나 홈서버를 통하여 메디컬 정보 센터에 보내고 그곳에서는 환자의 상태를 관리하고 전문 의료인의 관리 메시지를 저장하고 환자 및 간병인에게 그 정보를 보내주는 서비스에서 데이터 정보들의 효율적인 전송과 포맷을 위한 표준화된 모델을 연구한다. 갖가지 테스트와 기존의 표준화된 방법을 통하여 최적에 가까운 모델을 도출하고 그에 따른 프로토타입을 구현하도록 한다.

본 논문은 2장에서 홈네트워크에 대한 표준화 모델의 동향을 분석하고, 3장에는 표준화 모델 연구를 통하여 홈게이트웨이의 구현에 대해 설명하였다. 마지막으로 4장에서는 결론을 맺는다.

2. 표준화 동향

이 절에서는 홈네트워크 미들웨어들, 홈게이트웨이에 대한 표준화 동향에 대해 알아본다. 홈게이트웨이의 사실상 표준화(De Facto Standard)를 위한 단체인 OSGi Alliance가 있고 [2], 국제 표준으로서는 ISO/IEC JTC1 SC25 WG1 이 있다. 이들에 대한 간단한 기술적인 특징과 표준화 동향에 대해 알아본다.

일반 가정에 홈네트워크가 구축이 되면서 많은 사용자들이 기기들을 홈네트워크에 연결하고 기기들을 제어하고 싶어 한다. 정보가전 미들웨어 기술은 이러한 정보가전 기기들 간을 연결하여 통신이 가능하게 하기 위한 플랫폼과 통신 프로토콜이라고 할 수 있다. 대표적인 것으로 HAVi(Home Audio/Video Interoperability) [3], UPnP (Universal Plug and Play) [4], Jini [5,6,7] 등이 있으나 현재는 UPnP 기술이 가장 대중적이라고 할 수 있다. 이 절에서는 대표적인 홈네트워크 미들웨어들에 대한 간략한 소개와 동향에 대해 알아본다.

2.1 홈네트워크 미들웨어들

2.1.1 Jini

Jini는 1998년 SUN 사에서 발표한 분산 환경의 홈네트워크 자원공유 시스템이다. Jini는 하부구조로 Java RMI(Remote Method Invocation) 와 RMI 에 통합된 분산 환경에 대한 Java 플랫폼 보안모델을 갖는다. 상위 계층으로 서비스 발견 및 통지를 하는 디스커버리(Discovery), 서비스를 검색하는 룩업(Lookup), 갱신 가능한 지속 기반 모델을 사용하여 자원 할당을 보장하는

리싱(Leasing), 분산 환경에 대한 Java Beans 이벤트 모델을 확장한 이벤트, 그룹에 대한 모든 변경을 두 단계 커밋(two-phase commit) 프로토콜을 처리하는 트랜잭션과 단순한 통신과 Java 객체의 관련 그룹 저장을 위해 사용되는 JavaSpace가 있다. [5]

Jini 시스템은 Java 어플리케이션 환경을 1대의 VM(Virtual Machine)으로부터 네트워크에 접속된 여러 대의 기기에 적용 가능하도록 확장한 것이다. Java 어플리케이션 환경은 분산 컴퓨터에 있어서 코드 및 데이터가 기기간의 이동하는 것이 가능하게 해준다. 또 별도의 기기로부터 내려 받은 코드를 의심하지 않고 실행할 수 있도록 보안 기능을 준비하고 있다. 룩업 서버가 네트워크를 이탈하지만 않는다면, 그리고 룩업 서버로 너무 많은 요청이 몰리지 않는다면 기술적으로는 미들웨어들 중 가장 우수하며, 뒤에 소개할 OSGi의 기술적 근간을 이루고 있다. 하지만 곧이어 나온 MS사의 UPnP 에 밀려 향후 홈네트워크의 미들웨어가 되기에는 힘들며 SUN 사에서도 그 사실을 인지하고 있다. OSGi에서도 새로 나온 4.0 규격에서는 Jini에 대한 지원을 하지 않는다. 나중에 유비쿼터스 환경으로 갈 때를 위하여 선행연구에는 Jini가 바람직하지만 지금 현 단계로서는 Jini를 굳이 고집할 이유는 없어 보인다. 유비쿼터스 환경에서는 Jini의 개념을 차용한 보안 모델이 기술적으로 각광을 받을 수도 있지만 반드시 쓰인다는 보장은 없다.

2.1.2 HAVi

HAVi는 1998년 소니, 톰슨, 필립스, 도시바, 샤프, 히다치 등 8개 가전업체가 참여하여 홈 엔터테인먼트 서비스를 위한 미들웨어 업체 표준을 정의하는 그룹이다. 한때 20여개 업체가 참여했고 비비드로직(Vivid Logic) 사 등이 실제 HAVi를 구현하여 제품으로 내어놓기도 했다. 오디오 비디오 컨텐츠를 IEEE 1394 선 위에 스트림으로 전달하는데 문제가 없도록 연동시키기 위하여 표준 API를 만드는데 목적이 있었다. HAVi 기술은 IEEE 1394 홈네트워크에 가장 적합하도록 구성되어 있고 일부 기술이 DVM/MHP 에 포함되었으며 Open Cable에서 IEEE 1394를 채택하긴 했지만 주도를 하였던 소니(Sony)사마저 HAVi를 포기하였고, UPnP AV 와 DLNA 등을 통하여 대체가 가능하게 됨으로써 입지가 넓어지는 일은 없을 것이라 판단된다. 현재 노트북에 대부분 IEEE1394 포트가 있음에도 불구하고 실제 IEEE 1394 제품들이 거의 없어 아예 쓰지 않는 사용자들이 대부분이며 그런 사용자들이 늘어나는 것엔 다소 시간이 걸리리라 판단된다 [5].

2.1.3 UPnP

UPnP는 1999년 6월 MS, Intel, Compaq(현 HP와 합병하기 전), Mitsubish, Philips, Sony 등 150여 업체가 정의한 PC 중심의 가전기기 제어 미들웨어 단체 표준 기술이다. UPnP는 IP를 기반으로 하여 각종 가전기기와 PC등의 정보 네트워크(information network)가 투명하게 연동할 수 있는 개방형 표준을 정의하고 있다. 기기가 인터넷에 연결이 되지 않았을 경우에는 자동 IP(Auto IP) 주소 할당 및 관리를 해주고, 홈네트워크 상에서 기기나 서비스를 찾는 디스커버리 구동기, DNS 서버가 없는 환경에서 DNS 서비스를 제공하는 명칭 해결(Name Resolution) 서비스나 기기를 상세히 기술하는 데이터 구조인 기기 명세 스키마(Description Schema), 서비스 및 기기 규정 API 등이 요소 기술이다. UPnP는 XML로 메시지들을 전달하게 된다. 2003년 멀티미디어 파일들의 호환을 위해 창설된 DHWG(Digital Home Working Group)이 그 이듬해에 이름을 DLNA(Digital Living Network Alliance) 로 바꾸었는데, DLNA 에서 UPnP를 기본 아키텍쳐로 채택하여 UPnP는 그 입지가 넓어졌다. 하지만 UPnP는 그 크기가 커서 모든 홈네트워크 기기에 들어갈 수가 없는 구조이다. 가령 예를 들어 UPnP 용 전등이 따로 만들어진다던지 하는 일은 거의 일어나기 힘든 일이다. UPnP의 네트워크 컴포넌트는 다음과 같다.

- 기기(Device)
- 서비스(Services)
- 제어 포인트(Control Points)

UPnP 는 다음 다섯 개 단계를 거치게 된다. [5]

- 주소(Addressing) : UPnP 는 TCP/IP 프로토콜에 기반을 두고 있으며 핵심은 주소에 있다. 각각의 기기들은 DHCP 클라이언트를 가지며, 기기가 처음 네트워크에 접속될 때 DHCP 서버를 찾게 된다. 서버가 있으면 주소를 할당 받지만 없을 경우는 자동 IP를 사용한다.
- 발견(Discovery) : 기기가 주소획득 후 발견단계가 되는데 SSDP(Simple Service Discovery Protocol)를 사용한다. SSDP는 기기가 서비스를 네트워크 내의 다른 기기에 전달하도록 도와준다. 또 제어 포인트가 새로 추가되면 관심 있는 기기를 찾도록 해준다.
- 디스크립션(Description) : 제어포인트가 기기발견 후 제어 포인트는 기기에 관해 아주 작은 정보를 갖고 있다. 제어 포인트가 기기의 능력을 좀 더 많이 알 수 있고, 또 기기와 상호작용하기 위해, 제어 포인트는 발견 단계에서 획득한 URL 정보를 이용하여 기기 디스크립션을 획득한다. 기기는 논리적인 디바이스와 서비스를 포함하고 있다. 디바이스에 관계된 UPnP 디스크립션은 XML로 포함되어있다. 디스크립션은 모든 디바이스와 서비스 리스트, 제어, 이벤팅 그리고 프리젠테이션을 위한 URL을 포함하고 있다.
- 제어(Control) : 제어 포인트는 위의 단계들을 통해 관련된 중요 정보들을 확보하게 된다. 서비스에 관련된 정보를 더 확보하기 위해, 각각의 서비스와 관련된 UPnP 디스크립션을 찾아야 한다. 서비스와 관련된 디스크립션은 XML로 표현되어있으며 명령, 액션, 서비스 응답, 매개변수의 리스트를 포함하고 있다. 이 변수들은 실행 시 서비스 상태를 모니터링 하고 변수들의 데이터 타입, 범위, 그리고 이벤트 속성을 기술한다. 기기를 제어하기 위해 제어 포인트는 액션 요구를 기기 서비스에게 요구한다. 이 때 제어 포인트는 적절한 제어 메시지를 이 서비스와 연관된 제어 URL 에 보낸다. 제어 메시지는 SOAP(Simple Object Access Protocol)을 이용하여 XML로 표현된다.

- 프리젠테이션(Presentation) : 만약 기기가 프리젠테이션을 위한 URL을 갖고 있으면 제어 포인트는 URL 을 이용하여 이 페이지를 로딩하고, 사용자가 기기를 제어할 수 있도록 한다. 웹을 이용한 기기 제어는 기기 생산자에 의해 제어 레벨이 정해지게 된다.

2.2 OSGi Alliance

OSGi Alliance는 1999년 서비스 게이트웨이의 표준화된 API를 제공해주기 위하여 결성된 사실상 표준단체(De Facto Standard)이다. SUN의 초창기의 JES(Java Embedded Server)로부터 영향을 받아 15여개 회원사가 발족을 하였고 지금은 40개 가량의 회원사가 있다. 게이트웨이를 위한 프레임워크용 표준 API를 정의하였으며 버전이 올라갈 때마다 새로운 번들을 지원하는 API를 내놓고 있다.

초창기에는 홈네트워크 서비스를 제공해주기 위해 가정용 홈 게이트웨이를 목표로 하였으나 3.0부터는 자동차용 게이트웨이, 4.0에서는 모바일용 기기에 까지 확장을 하였고 현재 엔터프라이즈 단에서도 그 필요성이 증대하는 상황이다. Knopflerfish 나 Oscar 같은 공개 소프트웨어가 나와 있으나 안정성을 보장하지는 못한다. IBM의 자바 개발 툴인 Eclipse 3.0부터 OSGi를 기본으로 채택하였고, Apache 에서도 채택하였다. OMA(Open Mobile Alliance)의 DM(Device Management) 에서 쓰는 규격을 OSGi 의 규격으로 채택하였으며 OSGi Alliance의 멤버인 노키아와 모토롤라가 JSR(Java Specific Request) 232 규격 제정에 참여하였고 이것이 OSGi Mobile Spec 이기도 하다.

OSGi는 다른 표준과 달리 서비스를 전송해주는 입장에서 바라보고 있으며 가장 기본적인 홈네트워크의 어플리케이션이 (원격)관리라는 것이 될 것임을 간파하고 많은 전문가들이 고심하여 만든 규격이다. 사용자 한 사람만으로 봐서도 시간이 지남에 따라 관심을 갖는 서비스가 달라질 수 있으며, 수많은 사용자들을 위해 모든 가능한 서비스를 한 기계에 다 집어넣을 수 없다. 그러므로 필요한 서비스만을 다운로드하여 서비스를 해주는 개념으로 가야하고 그 때마다 서비스 기사를 파견하기 힘들 수 있으므로 동적으로 소프트웨어가 로딩되어 마치 프로세스처럼 동작하다가 필요가 없어지면 제거되는 라이프 사이클도 관리가 되어야 한다. 하나의 서비스에서 다른 서비스를 가져다 쓸 수 있는 구조를 가져 공간을 절약할 수 있으며 그러기 위해선 엄격한 번들(서비스들의 묶음인 소프트웨어로서 동적으로 로딩되는 소프트웨어 단위) 관리가 필요하다. 홈네트워크 분야의 경우 집안의 하드웨어를 필요할 때마다 교체하는 빈도는 거의 없어야 하지만 소프트웨어는 시시각각으로 변할 수 있다. 가령 패치를 해야 하는 경우 필요한 서비스를 다운로드해야 한다던지, 그 교체주기는 하드웨어 교체주기보다 상당히 빈번하게 일어나게 될 것이다. 그러기 위해서라도 동적으로 로딩이 되어야 하며 또한 보안 문제를 따지지 않을 수 없다.

OSGi는 이러한 철학에 기반을 두고 만들어진 서비스 게이트웨이용 표준이다. 유럽에는 이미 암암리에 많이 퍼져 있으며 직접 서비스 하고 있는 곳도 있다. UPnP가 홈네트워크를 위한 기기들을 연결하는 프로토콜의 성격을 갖고 있다면 OSGi는 외부에서 가정 내부로 서비스를 딜리버리 해주고 관리를 해주는 실행 환경이라고 보면 된다.

국내의 경우에도 사업을 실제 하려는 경우 OSGi의 기본 아이디어를 채택한 표준으로 제안한 경우도 있으며, 표준 제안하는 경우도 OSGi를 채택했을 경우 해결되는 문제점들이 절반이 넘게 됨을 알 수 있다. OSGi 의 장점은 언제 기능이 첨부되고 삭제되더라도 그것에 유연하게 대처할 수 있는 표준이며 미들웨어가 복수개가 채택되더라도 그것을 지원해주며 어느 것으로 대체되더라도 해당 번들 형태로 소프트웨어를 만들면 언제나 시스템 재부팅 없이 업그레이드 될 수 있다는 장점이 있다.

2.3 ISO/IEC JTC1 SC25

OSGi가 사실상 표준이었다면 ISO/IEC JTC1 SC25 WG1 에서 논의하는 것은 홈네트워크 기기간 상호연동을 위한 국가 표준을 제정하는 곳인데, 홈게이트웨이도 그 중 하나이며 해당 표준을 위한 여러 나라들의 경쟁이 치열한 곳이다. 현재 UPnP 등이 홈네트워크 미들웨어로서 강세를 띠고 있으며 OSGi에서도 세가지 기술을 SC25 에 제안을 해놓은 상태이다. 일본의 에코넷은 보안 부분에 치중되고 있다.

한국에서 제안한 CCP (Common Communication Protocol)가 NP(New Proposal) 단계를 넘어섰다. ETRI (한국전자통신연구원)에서는 UMB(Universal Middleware Bridge) 라는 이름으로 일부를 제안하려는 움직임이 있다(2007년 하반기 제안 예정). UMB는 UPnP, PLC (LnCP 와 LonWorks 둘다), HAVi, Jini 등을 다 지원해줄 수 있도록 만든 통합미들웨어 구현을 위한 한 방법이다. 홈게이트웨이의 기능을 지능형이 되는 한도내에서 최소한으로 단순화 시키고 홈서버와 차별성을 두고 있다.

2.4 U-Healthcare 서비스를 위한 표준 모델

이상 살펴본 바에 의하면 홈게이트웨이는 홈서버와 별도로 되도록 권장하고 있으며 홈게이트웨이는 24시간 구동되어야 한다. 그런 이유로 냉장고가 되어야 한다는 주장도 있었으나 현재는 그럴 가능성은 적어 보인다. 고급 디스플레이가 장착된 스마트 폰 제조사들도 홈게이트웨이가 될 수 있음을 주장하고 있다. 세탑박스 제조사들이나 RG 제조사들이 각기 홈게이트웨이가 되어야 함을 주장한다. 어떤 형태가 되든 24시간 구동되면서도 동적 로딩이 가능하여 달라지는 서비스에 대처 가능하여야 한다. 이를 위해 홈게이트웨이는 리눅스 기반 위에 JVM으로서 IBM J9을 선택하였고 안정적인 ProSyst mBedded 5.2 OSGi 프레임워크를 올렸다.

3. 본론

홈게이트웨이의 하드웨어 장비는 Intel PXA255/400MHz RISC Processor를 사용하였고 128MB의 SDRAM 메모리와

32 FLASH 메모리를 가지고 있다. OS는 Linux이고 IBM J9 자바 VMware위에 ProSyst mBedded 5.2인 OSGi 프레임워크를 올렸다. 그리고 홈서버와 헬스케어센터와의 통신을 위하여 Spheon JSOAP (SOAP 1.1)[8]을 사용하였다.

홈게이트웨이는 댁내에서 생체측정장비로부터 들어온 데이터를 헬스케어센터나 홈서버로 보내고 헬스케어센터및 관리 데몬으로 전송하는 부분과 헬스케어센터나 홈서버에서 및 관리 데몬에서 받은 데이터를 처리하여 Zigbee로 보는 부분으로 나뉘게 된다.

홈게이트웨이는 생체측정장비와 통신을 Zigbee를 이용하여 통신을 하게 되며, Zigbee와 홈게이트웨이는 시리얼 통신을 하게 된다. Zigbee와 통신은 기본적으로 RFC1055[9]의 규약으로 통신을 하게 되며 별도의 프로토콜을 만들어 통신에 있어서 데이터 손실을 최소화 하도록 하였다. 그리고 Zigbee를 관리하기 위해서 별도의 Manager Daemon과 통신을 하게 된다.

다음의 표 1은 생체 측정 단말기로부터 들어온 데이터를 홈서버나 헬스케어센터로 분기하여 전달하는 데이터 흐름도이다. 혈당기와 맥파기는 홈서버및 홈서버로 전달하여 응급상태를 사용자에게 알려주거나 측정된 데이터를 가공하여 헬스케어센터로 전달하게 된다. 홈게이트웨이에서 헬스케어센터로 전달되는 데이터로서는 맥파기를 제외한 모든 데이터가 전달된다.

표 1 데이터 흐름도

생체 센서 종류	홈게이트웨이 -> 홈서버	홈게이트웨이 -> 헬스케어센터
혈당기	O	O
심전도기	X	O
체성분 분석기	X	O
호흡/ 심박 단말기	X	O
체온기	X	O
맥파기	O	X

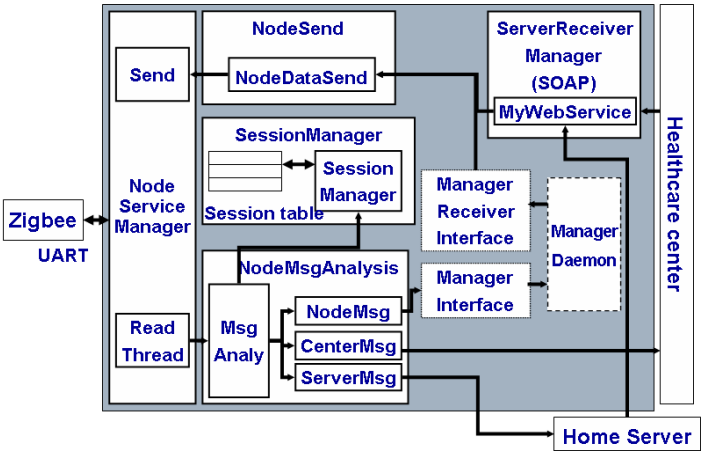

그림 1 게이트웨이 설계

홈게이트웨이는 그림 1과 같이 크게 7개의 모듈로 나뉘게 되는데 Zigbee와 통신을 위한 NodeServiceManager와 홈서버나 헬스케어센터로 데이터를 보내기 위한 NodeMsgAnalysis, 세션을 관리하기 위한 SessionManager, 홈서버와 헬스케어센터로부터 데이터를 받고 분석을 위한 ServerReceiverManager와 NodeSend가 있다. 그리고 Zigbee 관리를 위하여 ManagerReceiverInterface와 ManagerInterface가 있다.

3.1 Zigbee로부터 받은 데이터의 분석 및 처리 과정

NodeServiceManager는 Zigbee와의 통신을 담당하고 있다. 통신 속도는 19200bps이며 데이터 비트는 8bit, 정지 비트는 2bit로 통신을 하고 있다. 시리얼 통신 규약인 RFC 1055의 표준으로 데이터를 송수신 하도록 구현이 되어 있다. 통신하는 데이터 구조는 Header, Payload, Footer 필드로 나뉘며 Header는 데이터의 종류와 명령어, 채널정보 등이 포함되어 있다. 그리고 Data는 실제 데이터와 부가적인 정보들이 포함하고 있으며 Footer는 데이터의 유효성을 검증하기 위하여 checksum 비트를 두었다. 그래서 시리얼로 들어온 데이터를 RFC 1055규약으로 디코딩을 하여 Header 분석과 데이터가 유효성 검증을 위해 Footer 필드를 이용하여 검사한다.

NodeMsgAnlysis는 실제 Zigbee로부터 들어온 데이터를 처리하는 부분으로써 데이터의 헤더와 바디부분을 분석하고 데이터를 홈서버, 헬스케어센터 및 관리 데몬으로 보내기 위해 결정 하는 부분이다. MsgAnaly에서 데이터를 분석하여 전송될 목적지에 따라서 NodeMsg, CenterMsg, ServerMsg에 보내 전송하도록 구현하였다.

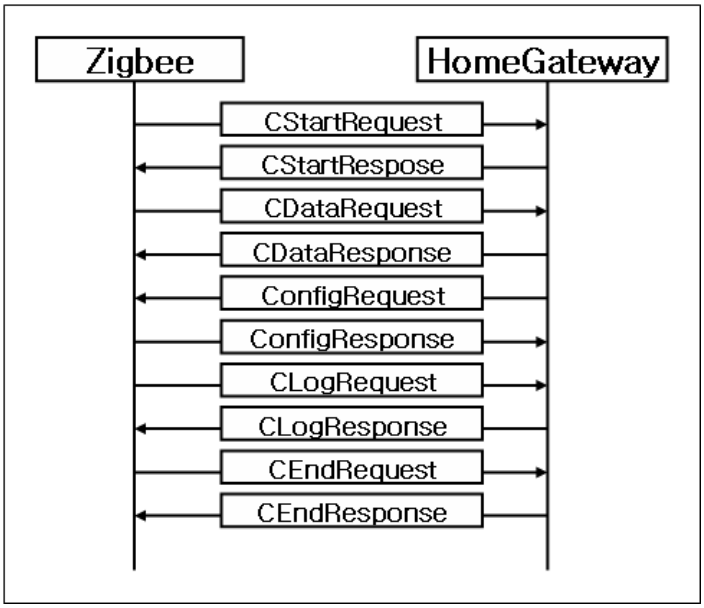

그림 2 측정 데이터 프로세서

전송된 데이터 형식은 위의 표 3과 같은 명령어 형식으로 데이터를 받게 된다. 명령어 필드는 Header에 포함 되어 있다.

CStartRequest와 CStartResponse는 데이터를 전송을 시작하겠다는 수신, 응답 명령어로써 생체측정장비의 식별번호와 다른 기타 정보들이 내포하고 있으며 CDataRequest와 CDataResponse는 실제 데이터를 전송과 전송을 받았다는

응답 명령어다. 데이터의 사이즈가 클 경우에는 여러번의 CDataRequest로 나누어 데이터를 보내게 된다. CConfigRequest와 CConfigResponse는 생체측정장비의 설정값이나 사용자 정보가 변경될 경우 전달되는 명령어이다. CLogRequest와 CLogResponse는 생체측정장비에서 쌓은 로그를 전송할 때 사용하는 명령어이고 CEndRequest와 CEndResponse는 메시지를 전송을 완료하겠다는 명령어다. CAdvDataRequest와 CAdvDataResponse는 생체측정장비가 시간이나 기타정보를 요청할 경우 사용되는 명령어이고 보통 생체측정장비가 데이터를 측정하기 전에 시간보정이나 다른 기타 정보들이 맞는지 확인할 때 사용한다. 그리고 명령어에서 CAdvDataRequest, CAdvDataResponse를 제외한 모든 명령어는 그림 2와 같은 처리 순서를 가지게 된다.

표 3 Healthcare Information Sensor

Command Name	Direction	Node Requirement	Gateway Requirement
CACK	any	M	M
CStartRequest	Node to gateway	M	-
CStartResponse	Gateway to node	-	M
CDataRequest	Node to gateway	M	-
CDataResponse	Gateway to node	-	M
CConfigRequest	Gateway to node	-	M
CConfigResponse	Node to gateway	M	-
CLogRequest	Node to gateway	M	-
CLogResponse	Gateway to node	-	M
CEndRequest	Any	M	M
CEndResponse	any	M	M
CAdvDataRequest	Node to gateway	M	-
CAdvDataResponse	Gateway to node	-	M

NodeMsgAnlysis에서 세션을 관리를 위해 SessionManager를 사용하게 되는데, 세션을 관리를 위해서는 CStartRequest와 CEndRequest를 이용하였다. CStartRequest는 Header의 채널별로 Payload의 생체측정장비의 식별번호를 SessionManager로 보내어 세션을 유지하도록 하고 생체측정장비의 식별번호에 따라서 홈서버 및 헬스케어센터로 데이터를 보낸다. CStartRequest 이외에 데이터 패킷들은 생체측정장비의 식별번호 정보가 없기 때문에 SessionManager에서 해당 채널의 생체측정장비의 식별번호를 가져와서 홈서버나 헬스케어센터로 보낼지 결정하게 된다. 그리고 CEndRequest에서 데이터가 올 경우 SessionManager의 해당 채널에 대한 센서타입을 삭제하도록 구현 되어 있다.

그리고 생체측장비로부터 들어온 실제 데이터를 전송하기 위해서 데이터를 적재를 해서 전송할 필요가 있는데 그 이유는 보통 체온계와 같이 측정 데이터의 사이즈가 작을 경우 거의 한 패킷에 데이터를 받게 되지만 심전도기와 같은 센서들은 측정된 데이터의 사이즈가 크고 연속으로 받게 되고 측정을 하지 않을 때까지 계속적으로 송신되기 때문에 SOAP 웹서비스의 부하를 주게 되며 그로 인하여 전송 결과 메시지를 늦게 주거나 웹서비스의 세션이 많이 생기는 문제가 발생하여 이를 해결하기 위해 데이터를 일정 사이즈로 적재하여 웹의 부하를 줄일 수가 있었다. 다른 이외에 명령어들은 한 패킷으로 이루어져 있기 때문에 적재를 시킬 필요가 없이 바로 전송이 가능하다. 그래서 이를 해결하기 위해서 CDataRequest 메시지의 Payload의 부분에서 측정 데이터가 마지막 패킷이라는 필드를 두고 그 필드가 셋이 될 때까지 데이터를 300개 단위로 묶어서 데이터를 전송하도록 구현했다.

ManagerInterface는 데이터를 관리용 데몬에게 데이터를 보내는 역할을 한는데, 별도의 통신 프로토콜을 사용하였다. 여기서 관리용 데몬으로 데이터를 보내기 위해 RFC 1055 규약으로 데이터를 인코딩하여 데이터를 전송하도록 하였다.

3.2 외부서버로부터 받은 데이터를 Zigbee로 전송하는 과정

ServerReceiverManager는 홈서버나 헬스케어센터로부터 들어온 데이터를 수신하게 되며 SOAP을 Sheop JSOAP로 구현하였다. Sheop JSOAP은 SOAP 1.1 버전으로 구현 되어 있다. 웹 서비스를 통해서 데이터 통신이 가능하게 구현 되어 있다. 여기서 받은 데이터를 NodeSend로 보내어 데이터를 분석하고 NodeServiceManager로 보내지게 된다. ManagerReceiverInterface도 마찬가지로 관리용 데몬으로부터 데이터를 받는 역할을 하는데 RFC 1055 규약으로 데이터를 디코딩하여 NodeSend에게 데이터를 보내어 처리하도록 하였다.

NodeSend에서는 데이터를 Zigbee로 전송하기 위해 데이터를 NodeServiceManager의 Send로 보내어 데이터의 유효성을 위해 Footer에 checksum을 만들어 넣은 다음 RFC 1055의 프로토콜의 정의를 통해서 인코딩을 한 후 Zigbee로 전송한다.

NodeSend는 현재 역할이 미비하나 향후 보안에 디코딩 등을 확장하기 위해서 만들어 놓은 번들이며 보안을 적용할 경우 많은 역할을 하게 될 것이다.

4. 결론

이상으로 U-Healthcare 서비스를 위한 표준화 홈게이트웨이 및 홈서버 표준화 모델에 대해 알아보았고 그에 따라 프로토타입을 설계하여 구현해 보았다. 국가 표준은 표준화가 완성이 되지 않고 진행중인만큼 전반적으로 표준의 큰 틀을 거스르지 않으면서도 앞으로 가장 가능성이 크며 추후의 변화에도 언제든 적응할 수 있는 시스템을 구현하였다. 홈게이트웨이와 홈서버는 U-Healthcare 서비스뿐만 아니라 홈네트워크에도 쓰이는 방향으로 하였으며 댁내의 가전기기 제어는 PLC 또는 Zigbee 어느 것이든 원하는 대로 번들을 통하여 쓸 수 있는 틀을 마련하였다. 홈게이트웨이와 홈서버는 표준화 동향대로 분리되는 방향으로 하였다. 홈게이트웨이의 하드웨어 사양은 현재에도 무리가 없고 점차로 고급

화, 지능화 되는 추세이므로 현재의 모델 및 앞으로 나올 새로운 서비스에도 쉽게 적응할 수 있으리라 기대한다. 보안 측면에서 보자면, 자바 모델의 Java 2 Security를 통하여 버퍼 오버플로 공격을 원천적으로 차단하고 데이터 통신은 SOAP을 사용하여 보안을 강화하였다. 변화에 대한 적응의 측면에서 보자면(flexibility), Zigbee 기기가 추가될 때마다 원격에서 업그레이드가 가능한 모델이며 나중에 다른 형태의 표준으로 진화하더라도 수용할 수 있는 모델이다. 이것은 Zigbee가 표준에서 사라지고 그것을 대체하는 다른 표준이 나와도 그에 적응을 할 수 있는 구조이다.

5. 참고문헌

[1] Zigbee Alliance site http://www.zigbee.org
[2] OSGi Alliance site http://www.osgi.org
[3] HAVi site http://www.havi.org
[4] UPnP Forum site http://www.upnp.org
[5] 한치문, 박광로, 디지털 홈네트워크 기술표준 개론, TTA, 2004
[6] 양재수, 전호인, 유비쿼터스 홈네트워킹 서비스, 전자신문사, 2004
[7] Gerado O'Driscol, The Essential Guide to Home Networking Technologiesl, Prentice Hall, 2001
[8] Florian Muller, Spheon JSOAP An implementation of SOAP 1.1 for Java, http://soap.fmui.de, 2003
[9] J.Romkey, Request for Comments: 1055, Network Working Group, 1988