2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

LonWorks 네트워크를 이용한 원격 홈오토메이션 시스템

손영성, 박준희, 이창은, 문경덕, 김채규

한국전자통신연구원 인터넷정보가전연구부

e-mail : {ysson, juni, celee, kdmoon, kyu}@ etri.re.kr

Remote Home Automation System using LonWorks Networks

Young-Sung Son, Jun-Hee Park, Chang-Eun Lee and Kyung-Doek Moon
Internet Appliance Technical Department, ETRI.

요 약

본 논문은 LonWorks 네트워크를 이용한 원격 홈오토메이션 시스템에 대해서 설명한다. 최근 홈네트워킹 개념의 확산에 따라 다양한 정보가전 기술의 발전이 가속화 되고 있다. 또한, 다양한 정보 가전 기기를 연동하기 위한 다양한 매체(*IEEE1394, USB, PLC, TP, IRDA, BLUETOOTH* 등)가 제시되고 있고 이를 제어하기 위한 미들웨어(Jini, Havi, LonWorks 등)가 거론되고 있다. 이러한 환경에서 가능한 서비스를 미리 살펴보고 이를 구성하기 위한 기본적인 요소 기술의 필요성을 살펴본다.
본 논문에서 설계한 원격 홈오토메이션 시스템은 제어 미들웨어에서 표준이 되고 있는 LonWorks를 이용하여 인터넷에서 홈내부의 디바이스를 모니터링 가능하도록 구축하였다.

1. 서론

인터넷의 빠른 확산과 정보의 디지털화에 힘입어 1990 년대 후반에 들어오면서 가정에 PC, 프린터 등의 정보기기, 캠코더, DVD, 비디오, 오디오 등의 AV 기기 및 냉장고, 세탁기와 같은 백색가전기기를 홈 네트워크에 연결하여 원격감시, 원격 검침, 원격 제어등의 홈 오토메이션 서비스로부터 정보기기 사이의 데이터 공유와 인터넷 공유 및 홈씨어터 서비스를 제공하려는 연구가 진행되고 있다. 이러한 연구는 주로 가정내의 기기들을 연결하기 위해 전화선, 무선, IEEE 1394, 전력선 등 홈 네트워크 통신 방법에 대한 연구, 홈 네트워크에 연결되는 기기들간에 상호 운영성을 보장하는 미들웨어 기술, 홈 네트워크와 인터넷을 연동시켜주는 게이트웨이 기술과 홈 네트워크에 연결되는 단말기술에 집중되고 있다.

최근 OSGi [3]등 홈 제어 네트워크의 표준을 만들려는 시도가 이루어지고 있으나 구체적인 결과가 구현될 것은 없다. 각기 단말기기의 특성에 맞춰 Jini[4], Havi[5], LonWorks[2] 등 제어 미들웨어의 스펙이 제시되고 각자 구현중이다.

본 논문에서는 제어 네트워크 분야에서 산업 표준이 되고 있는 LonWorks 를 이용하여 인터넷에서 원격 제어가 가능한 홈오토메이션 시스템을 설계, 개발한 연구 결과를 기술한다.

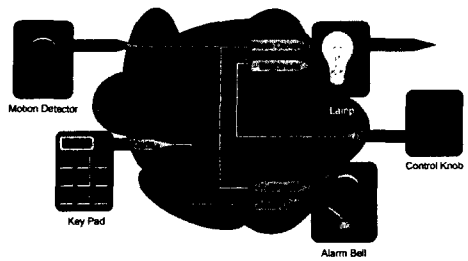
2. LonWorks 네트워크

LonWorks 네트워크는 전송할 정보량이 적고 발생빈도가 규칙적이며 정보 전송에 상대적으로 신뢰성이 있는 제어네트워크에서의 산업 표준 (de facto standard)이다.

기존의 제어네트워크에서 클라이언트/서버 아키텍쳐 또는 중앙집중적인 메인프레임/더미 터미널 아키텍쳐에서 분산제어방식(DDC, PLC)로 Peer-to-Peer 방식의 네트워크 구조를 사용한 네트워크이다.

LonWorks 의 통신 방식은 흔히 정보 기반 통신

(Information –based communication) 으로 기존의 명령어 기반 통신(command-based communication)과 구분된다.

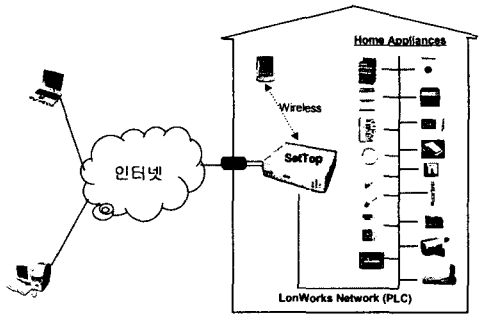


[그림 1] LonWorks 네트워크 구조

각 디바이스는 네트워크 변수 (Network Variable : NV) 라는 특별한 형태의 입출력 오브젝트를 이용하여 디바이스 간의 바인딩 정보만을 바꿈으로서 통신을 재설정할 수 있다. 이러한 네트워크 변수는 LonMark 협회[1]에서 표준적으로 정하여 표준 네트워크 변수 (Standard Network Variable Type : SNVT)를 미리 정의함으로써 여러 밴더의 디바이스가 함께 운용될 수 있는 상호개방성(Interoperability)를 제공한다[2].
모든 디바이스는 Neuron Chip 을 탑재하고 있으며 이 칩은 LonWorks 디바이스의 핵심이라 할 수 있는 LonTalk 프로토콜을 가지고 있어 LonWorks 통신을 가능하게 해준다. 각 디바이스 개발자는 해당 디바이스의 동작을 기술하기 위해서 Neuron-C 를 이용해서 동작을 구현한다. Neuron-C 에서 NV 의 동작 및 이벤트를 구체화하고 이에 따른 전체 네트워크 동작은 NV 의 바인딩에 의해서 결정된다.

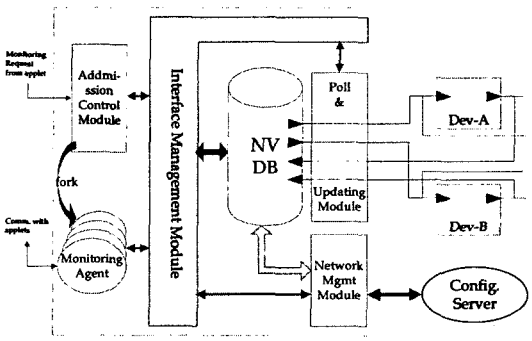
3. 원격홈오토메이션시스템

[그림 2]은 본 논문에서 정의하는 홈네트워크 서비스의 기본적인 물리적 환경을 보여주고 있다. 그림에서 클라이언트 단말은 인터넷으로 연결되어 있는 노트북과 데스크 탑 컴퓨터, 그리고 집안에서는 무선으로 연결된 PDA 단말이 된다. 일반적으로 홈 가전(Appliance) 이라고 하는 다양한 제품군들이 있다. 센서, 전구 등과 같이 단순한 제품에서부터 냉장고, 세탁기와 같은 약간의 지능형 디바이스, TV, VTR, Audio 들과 같은 고 지능성 장비에 이르기까지 폭 넓은 분야에 수만 가지의 디바이스가 존재하고 있다. 이러한 장비들이 인터넷 저편의 PC 나 노트북, 혹은 집안의 임의의 위치에 존재할 수 있는 이동형 단말에서 제어될 수 있기 위해서는 어떤 형태로든 네트워크를 구성하고 있어야 한다. 특히, 최근에 LonWorks 에서 사용하고 있는 다양한 데이터 송수신 Transceiver 중에서 전력선 통신(PLC) 방법이 네트워크 환경으로 각광을 받고 있다. 또한, 그림에 있는 모든 가전 디바이스들에는 LonWorks 통신모듈 (Neuron-chip embedded controller)을 가지고 있다. 외부 인터넷에 접속되어 있는 유선 단말이나, 집안 내부의 무선 통신 단말에서 가전 제품을 제어 및 모니터 하기 위해서는 기본적으로 Settop Box 라고 하는 홈 장비를 경유해야 한다. Settop 에는 Web Server 와 HMS(Home Management System)이 존재하게 된다.



[그림 2] 홈네트워크 서비스 환경

[그림 3]는 본 논문에서 기술하고 있는 원격 오토메이션 시스템을 위한 S/W 블록 구조도를 보여주고 있다. 본 시스템에서 제어되는 가전기기들의 약속된 통신을 위해서 LonTalk 프로토콜과 전력선 통신용 전송장비가 사용된다. 또한, 홈 브라우징을 지원하기 위해서 홈서버는 서블릿이 지원되는 웹 서버를 가지고 있어 원격 오토메이션 서비스를 받기 위해서는 클라이언트들이 웹 브라우저를 내장하고 있다. 이러한 환경에서 클라이언트와 홈서버 사이에는 TCP/IP 를 사용해서 홈브라우징을 지원한다.



[그림 3] LonWorks 소프트웨어 블록 구조도

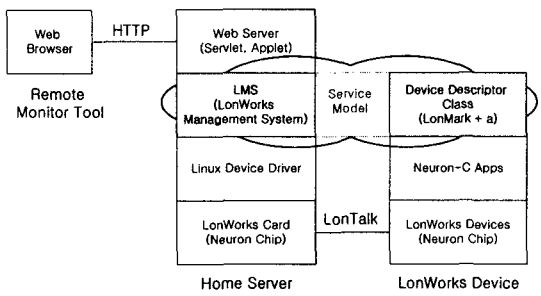
클라이언트에서 실제적으로 서비스를 수행하는 모듈은 애플릿(Applet)이다. 애플릿은 가전기기, 즉 디바이스 마다 하나씩 존재한다. 애플릿은 사용자가 선택한 디바이스를 제어하고, 모니터 하므로 가전기기 개발자가 공급한다.
애플릿은 모니터링 서블릿(Monitoring servlet)과 1:1 통

신을 한다. 서블릿은 애플릿의 요청에 따라 웹 서버에 의해서 생성되는 Java 로 프로그램된 Thread 모듈로서, 애플릿이 제어하고자 하는 가전기기에 대한 정보를 가지고 있다. 서블릿은 애플릿에서 전달되는 사용자 요구(모니터, 제어)를 HMS 로 전달하는 기능을 수행한다.

뉴론 응용(Neuron Application)은 뉴론 칩(Neuron-Chip)의 응용 프로세서에서 수행되는 응용 프로그램으로서, 일반적으로 가전기기 개발자가 자신의 기기를 효과적으로 제어하기 위해서 제작한 디바이스용 응용이다. 뉴론 응용은 Neuron C 프로그램 언어를 이용해서 구현된다.

4. 서비스 구성 및 구현

기본적인 홈오토메이션을 위한 홈서버는 Linux 로 구현된 Settop Box 이며, 여기에는 Web Server 및 Servlet 을 구동할 수 있는 환경이 구축되었다. 기본적인 홈 네트워크 내의 네트워크 구성(Network Configuration)은 이 Settop box 에 저장되어 있고 각 디바이스를 웹 브라우저에 표시하기 위한 Icon 과 Applet 을 저장하고 있다.



그림[4] 홈오토메이션 서비스 구성도

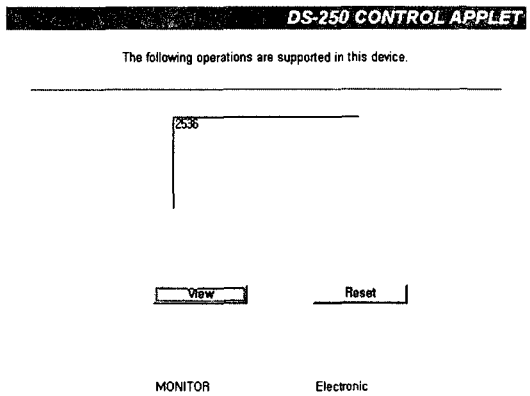
4.1 서비스 시나리오

본 장에서는 원격지에서 클라이언트가 홈 브라우징 서비스를 받기 위해서 서버와 통신하는 절차에 대해서 기술한다.

[1] 클라이언트는 웹 브라우저를 통해서 홈 브라우징 웹 서버에 접속한다.
[2] 홈 브라우징 웹 서버와 클라이언트의 웹 브라우저는 인증과 수락 제어 (Admission Control) 과정을 수행한다.
[3] 연결이 수락되면 웹 서버는 클라이언트 웹 브라우저에게 홈 그래픽 페이지를 공급하고, 클라이언트는 이 페이지를 사용자에게 보여준다. 이때, 초기화면에서 보여주는 그림은 홈 가전들의 현재 상태를 약식(ex. Icon)으로 보여준다.
[4] 사용자는 브라우저를 통해서 서비스 받고자 하는 가전기기를 선택한다.
[5] 웹 서버는 사용자가 선택한 가전기기의 애플릿(Applet)을 클라이언트 웹 브라우저에 다운로드 한다.
[6] 다운로드된 애플릿은 웹 서버에게 에이전트 서블릿(Servlet)의 생성(fork)을 요구한다.
[7] 웹 서버는 애플릿이 요구한 서블릿을 생성하고, 클라이언트에 의해 선택된 디바이스 정보를 서블릿에게 전달한다.
[8] 서블릿은 애플릿과 새로운 (TCP)연결을 설정한다.
[9] 사용자는 애플릿에 의해서 보여지는 디바이스 인터페이스를 통해서 선택된 디바이스를 모니터 혹은 제어하며, 사용자 입력 내용은 애플릿에 의해서 패킷으로 생성되어 서블릿에게 전달된다.
[10] 애플릿의 요구 패킷을 받은 서블릿은 요구된 내용에 선택된 디바이스의 정보를 추가하여 데이터 패킷으로 HMS 에게 전달한다. 서블릿과 HMS 간의 통신에 사용되는 메커니즘은 IPC 가 될 수도 있고, 실제 네트워크 통신이 될 수도 있다. 이를 동시에 만족하기 위하여, TCP/IP Socket 을 이용한다.
[11] HMS 는 LonWorks 디바이스를 제어하여 사용자가 원하는 결과를 약속된 형태로 리턴한다.
[12] 서블릿은 리턴된 결과를 애플릿에게 전달한다.
[13] 애플릿은 리턴된 결과를 사용자에게 그래픽 형태로 보여준다.
[14] 위의 [9]~[13]의 과정을 반복한다.
[15] 사용자가 애플릿을 Kill 시키면, 애플릿은 서블릿에게 연결 종료를 지시하고 프로세스를 마친다.
[16] 서블릿은 애플릿에게서 연결종료 신호가 오면 프로세스를 마친다.

4.2 시스템 구현

본 장에서는 앞에서 기술한 서비스 시나리오를 수행하는 S/W 모듈들을 정의한다. 본 서비스를 제공하기 위해서 필요한 S/W 모듈은 3 가지이다.



[그림 6] 전력계 Applet

첫째, Web Browser 에서 구동될 HTML 문서와 각 디

바이스의 GUI 를 나타낼 Applet 이다. HTTP Request 및 Response 의 메시징 방법을 따르는 일반적인 웹브라우저이며 기본적으로 Java Applet 을 구동시킬 수 있어야 한다. Java Applet 은 parameter 로 LonWorks 네트워크 내에서 전력계를 나태내는 Device ID 를 받는다.

둘째, 웹서버에서 구동되는 Servlet 이다. 웹 서버가 서블릿을 생성한 후 서블릿에게 정보를 공급하는 과정으로서, IPC 메커니즘을 이용한다. 애플릿과 서블릿 사이의 통신은 TCP/IP 를 통해서 이루어진다. 웹 브라우저에서 애플릿을 생성하면, 애플릿은 서블릿의 생성을 웹 서버에게 요구한다(시나리오[6]). 웹 서버에 의해서 생성된 서블릿은 애플릿과 새로운 연결을 설정한다(시나리오[7]). 서블릿과의 연결설정 작업이 마무리 되면, 비로소 사용자에게 홈 브라우징 서비스를 제공할 수 있는 준비가 된다. 서비스를 위한 모든 준비가 마무리되면, 애플릿은 사용자 입력을 기다린다. 사용자 입력이 발생하면, 애플릿은 원하는 서비스를 서블릿에게 요구한다. 애플릿이 서블릿에게 요구하는 서비스는 UPDATE 와 MONITOR 의 두 가지이다. 그러므로, 애플릿이 서블릿에게 전달하는 패킷의 내용에 포함될 내용은 다음과 같다.

<Network Variable> :: <UPDATE/MONITOR> :: < DATA >

이 내용을 포함한 패킷의 구조와 구조체 선언은 다음과 같다.

```
typedef   struct {
          short     totalLen;
          short     dataLen;
          short     seqNum;
          char      cmd;
          char      data[1];
} Applet_Svl_Msg;
```

[그림 7] 애플릿-서블릿간의 패킷 포맷

상기한 자료구조는 서블릿이 요구된 서비스 결과를 애플릿에게 전송할 경우에도 같은 포맷으로 사용된다.
Total Length 는 패킷의 전체 크기를 나타낸다.
Data Length 는 제일 마지막 Data 필드의 길이를 나타낸다. 이 패킷에서 가변길이를 갖는 필드는 NV type 필드와 Data 필드가 있다. NV type 필드는 LonMark Guideline 에서 정의하고 있는 SNVT(Standard Network Variable Type)이나 UNVT(User-defined Network Variable Type)이 들어가게 된다. Data 필드는 Update 혹은 Monitor 되는 변수의 실제 값을 갖게된다.
Sequence No.는 Query 와 Reply 의 일치성을 보장하기 위해서 사용되는 일련 번호이다. 애플릿이 시작되면 애플릿은 0 부터 하나씩 숫자를 증가한다.

```
public void service(ServletRequest request,
ServletResponse response) {
try{
    br = new BufferedReader(
    new InputStreamReader(request.getInputStream()));
    String str = br.readLine();

    s = new Socket("129.254.74.49",6000);
    out = new BufferedWriter(
        new OutputStreamWriter(s.getOutputStream()));
    out.write(send);
    out.newLine();
    out.flush();

    in = new BufferedReader(
    new InputStreamReader(s.getInputStream()));
    String temp = in.readLine();

    pw = new PrintWriter(response.getOutputStream());

    pw.println(temp);
    pw.flush();
    pw.close();
    br.close();
    in.close();
    s.close();

}//catch(Exception)
}//service(request, response)
```

마지막으로 서블릿과 HMS 이다. 서블릿은 애플릿과 HMS 의 인터페이스 역할을 한다. 애플릿과 서블릿의 인터페이스는 앞에서 정의한 프로토콜을 따르고, 서블릿과 HMS 와의 인터페이스는 IPC 메커니즘을 이용한다. 홈에서 웹 서버의 위치와 HMS 의 위치는 물리적으로 한 곳에 존재한다는 보장을 할 수 없다. 그러므로, 서블릿과 HMS 의 통신을 위해서는 Socket IPC 를 이용한다. 시나리오 설명에서 잠시 언급하였듯이 기기 개발자가 사용자 정의 서블릿을 사용할 경우에는 서블릿과 HMS 의 프로토콜을 이해해야 한다.
서블릿은 애플릿으로부터 [그림 7]의 패킷을 받은 후 자신이 보유하고 있는 디바이스 정보(Neuron ID)를 추가하여 HMS 에게 Query 를 보내게 된다.

5. 결론 및 추후과제

본 논문에서는 최근 관심이 높아지고 있는 홈네트워킹 서비스에 대한 원격 홈 오토메이션 시스템을 소개하고 전력계를 이용한 원격감시시스템에 대해서 설명하였다.

추후과제로 최근 홈네트워크의 서비스 모델로 많이 소개되고 있는 OSGi 의 구성에 맞춰 시스템을 최적화하는 작업을 수행 예정이다.

참고문헌

[1] LonMark, http://www.lonmark.org
[2] 백우진,"LonWorks 기초", http://www.echelon.co.kr
[3] OSGi, http://www.osgi.org
[4] Jini, http://www.jini.org
[5] Havi, http://www.havi.org