EIA 709.1 표준을 지원하는 리눅스 기반 홈 제어 네트워크 관리 플랫폼 구현

박 준 희[†] · 손 영 성[††] · 문 경 덕[†††]

요 약

본 논문에서는 홈 제어 네트워크의 대표적 표준으로 자리잡고 있는 LonWorks 시스템의 구성관리 소프트웨어의 플랫폼(LonWare)을 리눅스 기반에서 구현한 내용을 기술한다. LonWare는 NMML, LonWare API, LonWare DB의 3개의 모듈로 이루어져 있고, 응용 프로그램 개발자에게 LonWorks 시스템을 쉽게 접근 및 제어할 수 있는 인터페이스를 제공한다. 또한, 가정 내의 디바이스 네트워크 DB를 가정 내 홈 서버에서 관리하므로 보안상의 단점을 보완할 수 있다.

Implementation of Management Platform of Home Control Network based on EIA 709.1 Standard

Jun hee Park[†] · Young-Sung Son[††] · Kyeong Deok Moon[†††]

ABSTRACT

In this paper, we describe the implementation of the configuration platform (LonWare) based on Linux for LonWorks which is one of the popular standard of home control network. LonWare consists of three modules NMML, LonWare API, LonWare DB, and it provides semantically well-defined APIs for application device developers to easily access and control. And, LonWare DB is not needed to be located the outside of home, so the security-safe configurator can be made.

키워드 : EIA 709.1, 홈 제어 네트워크(Home Control Network), LonWorks, LonWare

1. 서 론

인터넷 정보가전이라고 하면 가정 내에 존재하는 디지털 TV, 인터넷 냉장고, DVD 등등, 컴퓨팅 기능과 네트워킹 기능을 구비하고 있는 백색 가전 기기를 연상하게 된다. 그러나, 마이크로 프로세서의 고속화와 경량 및 소형화에 따라 센서, 액츄에이터(Actuator)와 같은 많은 소형기기에 까지 마이크로 프로세서가 내장되면서 인터넷 정보가전의 영역은 가정 내의 모든 디바이스에까지 확장되어가고 있다. 특히, 지능형 홈 혹은 사이버 홈 등과 같은 인터넷 정보가전 산업의 대표적인 응용이 효과적으로 지원되기 위해서는 센서와 같은 소형 디바이스의 원격 및 자동 제어와 모니터 서비스가 이루어져야 한다. 센서 및 액츄에이터(Actuator)와 같은 가정 내 소형 디바이스들을 연결하여 구성된 네트워크를 홈 제어 네트워크라고 명명한다.

Echelon사에서 개발, 공급하고 있는 LonWorks는 다양한 통신 매체를 지원하지만, 특히 전력선을 통한 통신을 통해서 홈 제어 네트워크의 종합 솔루션을 제공하는 시스템이다. LonWorks 시스템은 ANSI 표준으로 등록되어 있는 EIA709.1(LonTalk) 이라고 하는 네트워크 프로토콜을 이용해서 통신을 하며, 자체적으로 개발한 LNS(LonWorks Network Service)라고 하는 비 표준화된 구성 관리 기술을 통해서 네트워크를 관리하도록 하고 있다. 또한, 다양한 소형 디바이스들간의 상호 운용성을 지원하기 위해서 LonMark 라고 하는 컨소시엄을 구성하여 디바이스들 간의 인터페이스를 표준화 하는 작업을 진행하고 있다.

Echelon에서는 LonWorks 시스템의 구성관리 도구로서 Windows 운영체제 상에서 동작하는 LonMaker라는 소프트웨어를 제공하고 있다. LonMaker는 상기한 LNS를 인프라로 활용하는 도구로서, LNS는 빌딩 혹은 가정 내에 LonWorks 네트워크로 연결되어 있는 디바이스들을 찾고, 정보

† 정 회 원 : 한국전자통신연구원 정보가전연구부 선임연구원
†† 정 회 원 : ETRI 근무
††† 정 회 원 : 한국전자통신연구원 정보가전 제어 S/W 연구팀장
논문접수 : 2002년 3월 8일, 심사완료 : 2002년 4월 24일

를 DB화하여 구축하며, 각 디바이스의 주요 자료구조를 수정하여 디바이스들간에 연동이 가능하도록 지원한다. 그러나, LNS는 상술한 바와같이 Windows 운영체제 상에서 개발되었으므로, 오직 Windows를 탑재한 호스트에서만 동작할 수 있다. 또한, LNS의 내부 명세와 DB 구조등에 대해서는 전혀 개방하고 있지 않기 때문에, 현재는 Echelon 사에서 공급하는 소프트웨어들을 그대로 활용하는 방법이외에 대안이 없는 실정이다.

ETRI에서는 LonWorks 네트워크에 연결되어 있는 디바이스를 구성, 관리, 제어, 모니터가 가능한 구성관리 도구를 리눅스 기반에서 개발할 수 있는 하부구조를 구현하였다. 즉, 리눅스 상에서 상술한 LNS와 같은 기능을 수행하는 소프트웨어 계층을 개발하였다. 본 논문에서는 LonWare ver 1.0으로 명명된 리눅스용 LonWorks 구성관리 하부구조의 구현 내용을 기술한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구로서 홈 제어 네트워크의 기술로 소개된 기술들을 설명하고, 3장에서는 Echelon사의 제품군을 중심으로 LonWorks 시스템의 구성과 서비스, 관리 개념과 전략 및 방법에 대해서 간략히 소개한다. 4장에서는 본 논문의 핵심 내용인 LonWare의 API에 대해서 구현 내용을 기술하며, 5장에서 향후 ETRI에서 계속 연구할 방향을 간략히 설명하고 맺음을 한다.

2. 관련 연구

2.1 X-10

X-10은 1976년부터 스코틀랜드의 Pico Electronics라는 회사에서 수행된 프로젝트의 이름이다. 이 프로젝트의 결과, 전력선을 이용하여 조명이나 단순 가전기기들을 제어할 수 있는 모듈이 개발되었으며, 1978년에서 1979년 사이에 상용화가 이루어졌다.

X-10은 전력선을 통신 매체로 사용하는 프로토콜로서, 256개의 주소할당이 가능하고, 6개의 간단한 함수와 명령어 -ON, OFF, DIM, RIGHT, ALL LIGHT ON, ALL UNITS ON -를 갖는다. 디바이스가 X-10 네트워크에 연결되기 위해서는 X-10 수신 모듈을 전원 단자와 디바이스 사이에 부착해야 하며, X-10 디바이스들은 상기한 X-10 주소와 명령어를 이해해야 한다.

X-10은 전력선 이외의 통신 매체를 수용하지 못하고, 256개의 주소와 6개의 함수만으로 구성된 프로토콜은 많은 가전 디바이스를 효과적으로 제어하기 힘들다. 또한, 대부분의 X-10 기기들은 단순한 기기들로서 단방향 통신만을 제공한다. 즉, 신호를 받아서 그 해당 동작은 수행하지만 다른 기기에 신호를 보내지 못하거나, 그 반대의 경우에 속하는 디바이스가 많다.

2.2 CEBUS

CEBUS는 1992년에 기존의 X-10이 가지고 있는 단점을 극복하고자 오랜 연구 및 토의 후 제정된 새로운 표준이다. CEBUS는 개방형 구조를 가지고 있으며, 전력선, TP(Twisted-Pair), 동축 케이블, 적외선, RF 등등 다양한 매체를 통해서 통신이 가능하다. 또한 모든 전송 매체에 동일한 7.5 kbps의 동일한 전송속도를 제공한다.

CEBUS는 OSI 7 계층 모델에서 4,5,6 계층이 없는 계층적 구조를 갖는다. 하부의 2 계층은 각 통신 매체별로 필요한 약속을 정의하고 있으며, 3계층에서는 라우팅과 네트워크 구성을 위한 프로토콜을 정의하고 있다. 가장 상위 계층인 7계층(CAL)은 CEBUS의 디바이스들이 통신할 때 사용되는 언어를 정의하고 있다. CAL에서는 자원의 할당과 제어의 두 가지 주요한 기능을 수행한다.

CEBUS는 EIA 표준으로서 디바이스간의 호환성을 보장한다. 또한, 하드웨어로 구현되어 있지 않은 표준으로서 제품 개발자가 하드웨어부터 모든 것들을 선택할 수 있으므로, 가격의 최적화가 가능하다. 또한, 개방된 표준화 미팅이 있어서 개발자가 표준화 작업에 참여할 수 있다.

상기한 바와 같이 CEBUS는 여러 기관과 기업이 모여서 만들어진 표준이다. 반면, LonWorks는 Echelon이라고 하는 한 회사가 만들어낸 표준이다. 이러한 배경의 결과 CEBUS는 LonWorks에 비해서 개발도구와 소프트웨어의 지원이 열악하다. 이것은 디바이스 개발자에게 있어서 치명적인 요소가 될 수 있다.

2.3 LonWorks

LonWorks는 1991년에 미국의 Echelon 사에 의해서 제안된 홈 제어 네트워크 기술이다. Echelon은 Neuron Chip에 네트워크 프로토콜과 기본적인 응용 환경을 하드웨어 혹은 훰웨어(Firmware)로 구현하였으며, 디바이스 개발자에게 다양한 개발도구를 제공하였다.

LonWorks는 OSI 7 계층을 모두 수용하는 네트워크 프로토콜과 응용 수준에서 네트워크를 구성, 관리, 제어할 수 있는 관리/진단 메시지와 자료구조를 정의하고 있다. 또한, 강력한 구성 관리 툴에 의한 강력한 망의 설정(Configuration) 기능을 갖으며, 설정 완료 이후에는 디바이스들 끼리 완벽한 분산 연동 시스템을 구성할 수 있다.

그러나, 지원하는 제품군이 Microsoft Windows에 지향적이고, 네트워크를 구성하고 설정하는데 필수적인 소프트웨어를 고가에 제공하고 있다.

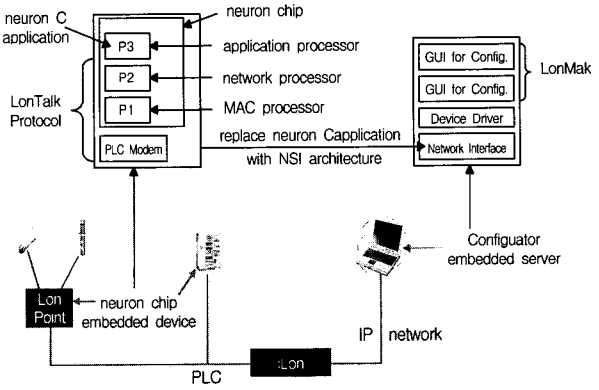
3. LonWorks 시스템

3.1 시스템 구성

3.1.1 Neuron Chip

LonWorks 시스템은 NC(Neuron Chip)가 내장된 디바이

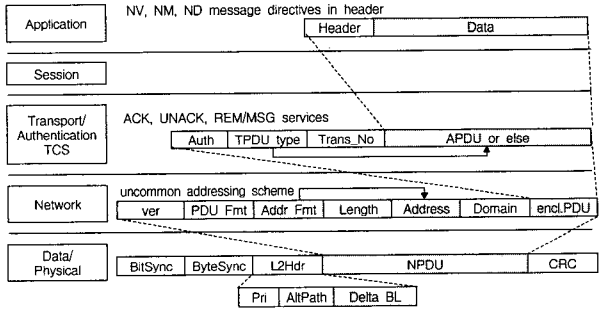
스들로 구성된다. NC는 LonWorks통신을 위한 MAC 프로세서와 네트워크 프로세서, 그리고 디바이스 응용 프로그램을 수행하는 응용 프로세서 등, 3개의 프로세서로 구성된 8비트 마이크로 프로세서이다. LonWorks의 통신 미디어는 앞장에서 기술한 바와 같이 전력선(PLC), TP(Twisted-Pair), RF 등 다양하며, 이들은 트랜시버(Transceiver)라는 ASIC으로서 NC의 MAC 프로세서와 연결된다. NC는 전등, 센서 등의 실제 전기 디바이스와 인터페이스 할 수 있는 입출력(IO) 핀을 가지고 있다. 디바이스 개발자는 NC의 입출력 핀에 자신들의 디바이스를 물리적으로 연결하여 LonWorks 디바이스를 디자인한다.



(그림 1) LonWorks 시스템의 구성도

3.1.2 LonTalk 프로토콜

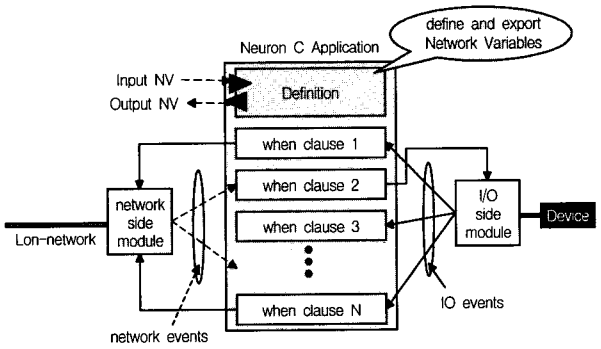
NC를 구성하는 3개의 프로세서중 2개는 네트워크 관련 프로세서이다. 이 두 개의 프로세서에는 EIA 709.1 이라는 이름으로 미국 ANSI 표준으로 등록된 네트워크 프로토콜 이하 LonTalk 이 하드웨어와 훰 웨어로 구현되어 있다. 앞장에서 기술한 바와 같이 LonTalk은 OSI 7 계층을 수용하는 네트워크 프로토콜로서 신뢰성을 보장한다. (그림 2)는 LonTalk 네트워크 프로토콜의 각 계층별 패킷 포맷을 보여주고 있다.



(그림 2) Packet format of LonTalk protocol

3.1.3 디바이스 응용 프로그램

NC의 나머지 한 프로세서는 실제 디바이스를 외부에서 제어하고 모니터 하기 위한 상위 인터페이스를 제공하는 디바이스 응용 프로그램을 수행하기 위한 프로세서이다. Echelon에서는 Neuron C라고 하는 크로스 컴파일러를 디바이스 응용 프로그램 개발을 위해 제공하고 있다. 이와 같이 디바이스 응용 프로그램은 NC의 응용 프로세서에서 동작할 수도 있지만(Neuron chip-hosted node), 복잡한 디바이스의 경우에는 NC에서 제공하는 응용 프로세서보다 강력한 외부 프로세서를 이용해서 자체적인 프로그램 언어를 이용해서 개발할 수가 있다(Host-based node). 후자의 경우에는 NC와 외부 프로세서간의 통신 방법이 필요하고 이를 위해서 MIP(Microprocessor Interface Program)라는 인터페이스가 정의되어 있다.



(그림 3) Neuron chip-hosted node의 응용 프로그램

(그림 3)은 Neuron chip-hosted node의 구성에서 디바이스 응용 프로그램을 자세히 보여준다. 모든 디바이스 응용 프로그램들은 하나 이상의 NV(Network Variable)와 하나 이상의 When 절을 갖는 것을 원칙으로 한다. NV는 디바이스의 정보를 외부로 제공하거나(Output NV), 외부 정보를 받아서 (Input NV) 자신의 디바이스 상태를 변경하는데 사용된다. When 절은 이벤트 처리를 수행하는 루틴으로서 이벤트에는 네트워크 이벤트와 디바이스 이벤트가 있다.

3.1.4 iLon

iLon은 인터넷 상에서 가정 내 디바이스의 상태를 모니터 하고 제어 할 수 있도록 하기 위해서, 또 인터넷 상에 있는 구성 관리 툴이 가정 내의 디바이스를 구성 관리 할 수 있도록 지원하기 위해서 설계된 게이트웨이이다. 후자를 위해서는 LonWorks의 관리 메시지를 인터넷 패킷에 캡슐화 하여 구성 관리 툴에게 전달하는 기능을 수행하며, 전자를 위해서는 자체에 웹서버를 두어 가정 내 디바이스의 상태를 보고자 하는 인터넷 상의 웹 브라우저에게 그 정보 제공을 한다.

3.1.5 Configurator

앞 장에서 기술한 LonMaker는 LonWorks 구성 관리 툴의 한 예이다. 최근에 발행된 LonMaker는 앞 단원에서 기술한 iLon을 이용하여 인터넷 상에서 가정 내 LonWorks 시스템을 구성 관리 할 수 있는 기능이 추가되었다.

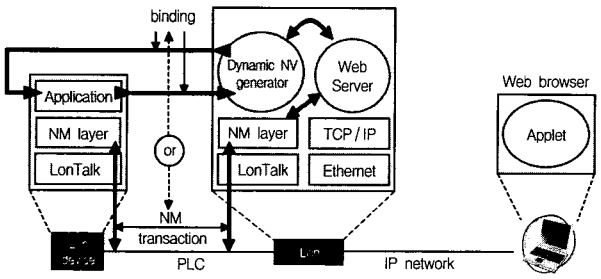
구성 관리 툴은 네트워크 상의 모든 디바이스들을 찾고, 이들의 구성 관리 정보를 모두 자신의 DB에 보관하며, 디바이스가 가지고 있는 중요 자료구조에 접근하여 그 내용을 변경 함으로서, 디바이스의 성질을 바꾸고, 각종 서비스를 생성/소멸하는 기능을 수행 한다.

3.2 서비스

센서 및 Actuator, 가전 등의 서비스는 디바이스 자체로 제공된다. 즉, LonWorks 네트워크에 연결이 되지 않더라도 디바이스 자체가 제공할 수 있는 서비스는 기본적으로 가지고 있다. 이러한 전기 디바이스들이 LonWorks 네트워크에 연결되면서 발생하는 중요한 서비스는 제어, 모니터, 그리고 연동 서비스라고 할 수 있다. 즉, 단순한 전기 디바이스들을 네트워크로 연결하면서 원격지에서 디바이스의 상태를 모니터하고, 제어할 수 있으며, 디바이스들 간의 인터페이스를 통해서 한 디바이스 상태의 변화에 따라 다른 디바이스가 특정 동작을 수행하게 하는 연동 서비스가 가능해 진다.

3.2.1 제어 및 모니터 서비스

이 서비스는 인터넷 상에서 가정 내 디바이스의 상태를 체크하고 Actuator 류의 디바이스에 대해 동작을 명령하는 서비스를 말한다. 이 서비스를 제공하기 위해서는 크게 두 가지 방법이 활용될 수 있다. 첫째, 가정 내에서 인터넷과 LonWorks간의 게이트웨이 역할을 수행하는 장치(iLon)에, 동적으로 모니터 및 제어하고자 하는 디바이스의 NV를 선언하고 구성 관리 툴을 이용하여 게이트웨이 NV와 다바이스 NV를 연결(Binding)하는 방법을 활용할 수 있다. 이를 위해서는 게이트웨이 내에 동적 변수 생성 기능이 포함되어 있어야 한다. 둘째, 게이트웨이 역할을 하는 장치가 직접 LonWorks 네트워크 관리 메시지를 통하여 디바이스의 특정 NV값을 읽어오고, 변경시키는 것이다. 이를 위해서는 게이트웨이에 이를 위한 모듈이 내장되어야 한다. (그림 4)는 모니터 및 제어 서비스의 방법을 그림으로 표현하고 있다.



(그림 4) 원격 모니터 및 제어 서비스

3.2.2 연동 서비스

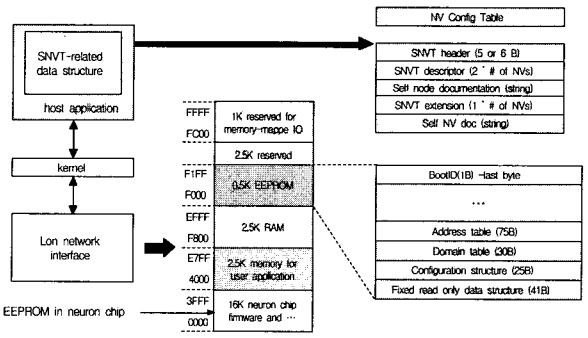
LonWorks에서 각기 다른 디바이스들이 연동하여 서비스를 제공하기 위해서는 각 디바이스들의 NV간 연결(Binding)이 필요하다. 이것은 구성 관리 툴에서만 지원할 수 있는 기능이다. 각기 다른 디바이스의 NV가 서로 연결되기 위해서는 두 NV가 연결가능 해야 한다. 연결 가능하기 위해서는 두 NV의 네트워크 변수 타입과 구체적인 성분이 일치해야 한다. 네트워크 변수의 표준 타입으로는 LonMark에서 정의한 SNVT(Standard Network Variable Type)이 있다. 현재 200여개의 표준 변수 타입이 정의되어 있다. 또한, 타입이 일치하더라도 구체적인 성분 변수의 방향, 폴링 방법, 동기화 여부, 우선순위 등등 이 일치하지 않으면 두 변수는 연결될 수 없다. 이러한 복잡하고, 구체적인 내용을 검증한 후 두 변수를 연결하는 작업을 구성 관리 툴에서 수행한다.

LonWorks에서 NV간의 연결은 단방향이다. 즉, 출력 방향 변수와 입력 방향 변수간에만 연결이 가능하고, 이들 간에는 단방향 링크가 형성된다. 또한, 연결에는 1 : 1 연결과 1 : N 연결이 가능하다. 구체적인 연결 방법은 다음 장에서 기술한다.

4. LonWare API 설계 및 구현

4.1 LonWorks 구성 관리 시스템 및 LonWare 설계

4.1.1 주요 자료구조



(그림 5) LonWorks 디바이스의 주요 자료구조와 위치

(그림 5)는 Host-based 노드에서 주요 자료구조의 위치를 보여준다. Neuron-chip-hosted 노드의 경우에는 모든 자료구조가 NC와 메모리 영역에 존재한다. 그러나, Host-based 노드의 경우에는 응용 프로그램이 호스트 프로세서에서 동작하고 그 코드도 호스트의 메모리에 존재하기 때문에 모든 네트워크 변수 관련 정보는 호스트의 메모리 영역으로 옮겨지게 된다. LonWorks 디바이스는 LonWorks 서비스를 위해서 다음과 같은 자료구조를 갖는다.

- Read Only Data Structure

디바이스의 가장 기본이 되는 자료구조로서, Neuron ID, 프로그램 ID, 네트워크 및 응용 버퍼 수 와 크기 등등을 가지고 있다.

• Configuration Structure

통신 미디어와 관련된 자료 구조로서, 채널 ID, 노드의 위치 정보, 각종 Clock 정보 등을 포함한다.

• Domain Table

노드의 논리 주소를 갖는다. 모든 LonWorks 디바이스는 NC에 하드웨어 적으로 기록되어 있는 NC ID를 통해서 구성관리의 첫 단계 통신을 하게 된다. 그러나, 구성 관리 툴에 의해서 논리 주소를 할당 받은 후에는 도메인 테이블에 있는 주소를 통해서 통신을 하게 된다.

• Address Table

LonTalk의 네트워크 계층에서 관리하는 주소 테이블로서, 주소 테이블 인덱스라고 하는 논리 번호를 보고 실제 네트워크 주소로 변환하기 위해서 사용되는 자료구조이다.

• NV Config Table

하나의 NV가 갖는 속성에 대한 변경 가능한 내용을 가지고 있는 중요한 자료구조 이다. 필드는 변수의 연결 ID (Selector), 변수의 방향성(Direction), 우선순위, 연결이 설정된 후 참조하는 주소 테이블의 인덱스(addr_index), 인증 여부, 통신 서비스 타입, 자기 회귀 연결 가능 여부등을 갖는다.

• SNVT header

디바이스 응용 프로그램에 선언된 변수들에 대한 제반 정보를 가지고 있는 메모리 블록의 가장 앞부분에 오는 자료구조로서, 전체 블록의 크기, 변수의 수 등에 대한 정보를 가지고 있다.

• SNVT Descriptor

변경할 수 없는 변수의 속성을 가지고 있는 자료구조로서, 주로 NV Config Table에 있는 필드들의 값을 구성 관리 툴에서 변경할 수 있는지 여부를 나타낸다. 그밖에, 변수가 Config 변수인지 (nv_config_class), 변수의 접근 방법 (nv_polled, nv_offline, nv_sync), 그리고 변수의 표준 변수 타입 번호(snvt_type_index)를 가지고 있다.

• Node Documentation

노드에 대한 전체적인 설명을 문자열로 나타낸 필드로서, 노드에 대한 부가적인 설명을 보여준다.

• SNVT Extension Record

각 변수에 대한 추가적인 속성을 가지고 있는 필드로서, 변수의 이름, 변수에 대한 설명 등이 있는지 여부를 보여준다.

• NV Documentation

SNVT Extension Record 에서 지시한 변수 이름과 변수 설명이 실제로 문자열로 존재하는 부분이다.

4.1.2 관리 메시지

〈표 1〉 LonWorks 관리 메시지

NM Messages	Request Code	Success Response	Failed Response
Query ID	0x61	0x21	0x01
Respond to Query	0x62	0x22	0x02
Update Domain	0x63	0x23	0x03
Leave Domain	0x64	0x24	0x04
Updata Key	0x65	0x25	0x05
Updata Address	0x66	0x26	0x06
Query Address	0x67	0x27	0x07
Query NV Config	0x68	0x28	0x08
Updata Group Address Data	0x69	0x29	0x09
Query Domain	0x6A	0x2A	0x0A
Updata NV Config	0x6B	0x2B	0x0B
Set Node Mode	0x6C	0x2C	0x0C
Read Memory	0x6D	0x2D	0x0D
Write Memory	0x6E	0x2E	0x0E
Checksum Recalculate	0x6F	0x2F	0x0F
Wink	0x70	0x30	0x10
Memory Refresh	0x71	0x31	0x11
Query SNVT	0x72	0x32	0x12
NV Fetch	0x73	0x33	0x13
Device Escape Code	0x7D	0x3D	0x1D

〈표 2〉 LonWorks 진단 메시지

ND Messages	Request Code	Success Response	Failed Response
Query Status	0x51	0x31	0x11
Proxy Command	0x52	0x32	0x12
Clear Status	0x53	0x33	0x13
Query XCVR Status	0x54	0x34	0x14

<표 1>과 <표 2>는 LonWorks의 관리/진단 메시지를 보여준다. 디바이스 Reset을 위해 송신되는 Set Node Mode 메시지를 제외한 모든 관리 메시지의 송수신은 Request/Response 서비스를 기반으로 한다. 즉, 구성관리자는 디바이스에게 각 관리 메시지를 보내서 특정한 정보를 요구하거나, 특정한 동작을 요구하게 되고, 각 디바이스는 관리 메시지에서 요구하는 내용을 수행한 후, 그 성공 여부를 응답해 주어야 한다.

LonWorks의 구성관리자는 <표 1>과 <표 2>는 같은 관리/진단 메시지를 통해서 현재 네트워크 상에 존재하는 다양한 디바이스들을 구성 관리한다. 관리/진단 메시지의 용도는 구성 관리의 흐름에서 자세히 설명한다.

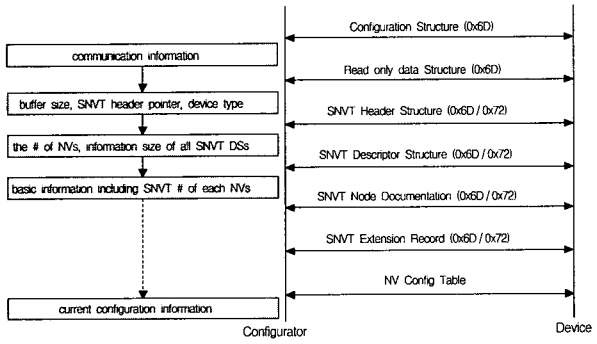
4.1.3 구성 관리의 흐름

구성 관리 서비스는 크게 세 단계로 나눌 수 있다. 첫째는 네트워크의 초기화 과정이다. 네트워크의 초기화라 함은

물리적으로 네트워크에 연결되어 있는 디바이스들을 찾고, 디바이스의 수집 가능한 모든 정보를 모아서 구성 관리자의 DB에 저장하는 것이다. 둘째는 디바이스의 구성 및 서비스 설정 과정이다. 디바이스들의 모든 정보를 가지고서 실제 사용자가 원하는 서비스를 위해서 디바이스들을 연결하는 과정이 여기에 속한다. 셋째는 서비스 해제 과정이다. 서비스 해제를 위해서 실제적으로 동작하는 것은 서비스 생성과정과 거의 동일하므로, 앞의 두 과정에 구성관리자의 모든 기능이 포함된다 할 수 있다.

4.1.3.1 네트워크 초기화 과정

네트워크에 물리적으로 연결되어 있는 디바이스에 관한 정보를 얻기 위해서는 먼저 네트워크 상에 존재하는 디바이스의 주소(Neuron ID)를 알아내야 한다. 디바이스의 주소를 알아내는 방법은 서비스 핀 메시지를 이용하는 방법과 Query ID(0x61) 메시지를 이용하는 방법이 있다. ETRI LonWare ver 1.0에서는 Query ID 메시지를 이용하여 현재 Application - Unconfigured 상태의 노드에 대해서 주소를 물어보는 방법을 이용한다.



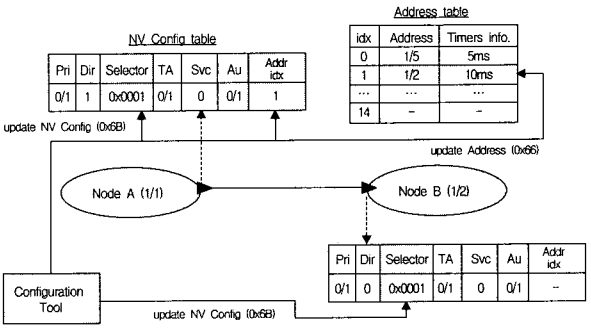
(그림 6) 디바이스 정보 수집 과정의 메시지 흐름

디바이스의 네트워크 주소를 알아낸 후에는 디바이스가 가지고 있는 정보를 수집하는 과정이 필요하다. 디바이스의 정보를 수집하는 방법에는 "Upload from Device"방법과 "XIF 파일"을 이용하는 방법이 있다. (그림 6)은 구성 관리자가 네트워크에 연결된 디바이스를 찾은 후 "Upload from Device" 방법을 통하여 디바이스에게서 주요 자료구조를 읽어 들이는 과정을 보여준다. 그림에서와 같이 디바이스로부터 얻어오는 정보는 Configuration, Read Only, SNVT header-descriptor-node doc.-extension, 그리고 NV Config 테이블 순서이며, 이 정보를 읽어오는데 활용되는 관리 메시지는 Read Memory(0x6D) 와 Query SNVT(0x72) 이다. XIF 파일은 LonMark에서 정의한 디바이스 정보 파일이다. 디바이스가 가지고 있는 모든 정보를 파일 형태로 표현한 것으로서 디바이스 개발자에 의해서 제공되어야 한다. 어느 디바이스에 대한 XIF 파일이 있을 경우에는 (그림 6)과 같이 디바이스로 길고 복잡한 과정으로 거쳐서 정보를 얻어올 필요 없이 XIF 파일을 분석하여 필요한 정보를 가져올 수 있다.

디바이스를 찾고, 그 정보를 수집한 후 구성관리자는 디바이스에게 NC ID가 아닌 논리 네트워크 주소를 할당한다. 이 주소는 Subnet 주소와 노드 주소로 이루어져 있어서, 망 관리에 유용한 주소 체계를 이루게 된다. 이때, 구성관리자에게 새롭게 할당 받은 주소는 Update Domain 관리 메시지를 통해서 디바이스의 Domain 테이블에 쓰여지게 된다. 네트워크에 물리적으로 연결되어 있는 모든 디바이스에 대해서, 이러한 과정이 수행되면 네트워크 초기화 과정이 완료된다.

4.1.3.2 디바이스 구성 및 설정 과정

모든 디바이스에 대한 정보를 DB화 하여 구성관리자가 구축한 후에는 사용자의 요구를 수용하는 서비스 생성 절차가 필요하다. 사용자가 요구하는 서비스는 결국 연동 서비스를 말한다. 즉, 두 개의 NV를 연결하는 것이 서비스 생성의 시작이자 끝이다. 두 개의 디바이스에 있는 두 변수를 연결하는 것을 내부적으로 살펴보면 두 변수의 속성 체크와 NV 컨피그 테이블의 Selector 값 변경, 그리고 출력 변수를 갖는 노드의 주소 테이블 변경으로 설명할 수 있다. (그림 7)은 네트워크 주소가 1/1 (Subnet/Node)인 노드 A와 1/2인 노드 B의 변수를 연결하는 과정을 보여준다. 그림에서 노드 A의 변수는 출력 변수이고, 노드 B의 변수는 입력 변수이다. 기타 속성 auth, priority, service type, 등등은 일치한다고 가정한다. 이 경우 구성관리자(Configurator)는 두 노드의 속성을 검증한 후, 현재 이 LonWorks 네트워크에서 사용되지 않고 있는 Selector 값을 할당하여 두 변수의 Config Table의 해당 필드에 써준다. 이때 사용하는 관리 메시지는 Update NV Config(0x6B)이다. 그리고, 출력 변수가 있는 노드-(그림 7)에서는 노드 A-의 주소 테이블에서 유휴(idle)한 필드를 찾아서 입력 변수가 있는 노드(B)의 네트워크 주소를 입력한다. 이때 사용하는 관리 메시지는 Update Address(0x66)이다. 또한, 새로운 주소가 입력된 주소 테이블의 인덱스를 노드 A출력변수의 NV Config Table의 addr_index 필드에 삽입한다.



(그림 7) NV Binding 과정

반대로 이러한 연결을 해제하고자 할 경우에는 Selector 값을 0x2FFF 보다 큰 값으로 변경하고, 노드 A의 주소 테이블에 있는 노드 B의 주소를 지워버린다. <표 3>은 지금까지 설명한 구성관리자의 기능을 정리해서 보여주고 있다.

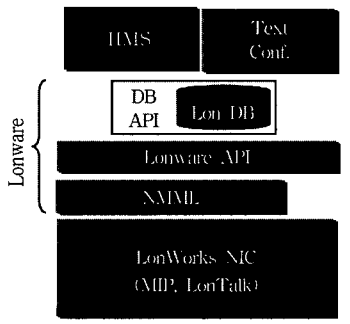
〈표 3〉 구성관리자의 주요 기능

#	과 정	방 법	Details
1	디바이스 찾기	Service Pin	물리적인 방법으로 서비스 핀 메시지 발송
		Query ID	디바이스에게 주소 요청
2	디바이스 정보 DB 구축	Upload from device	(그림 6)
		Read XIF file	XIF 파일을 통해 정보 수집
3	논리 주소 할당	Update Domain	Subnet/Node 주소 할당
4	서비스 구성 및 설정	Update NV Config, Update Address	(그림 7)

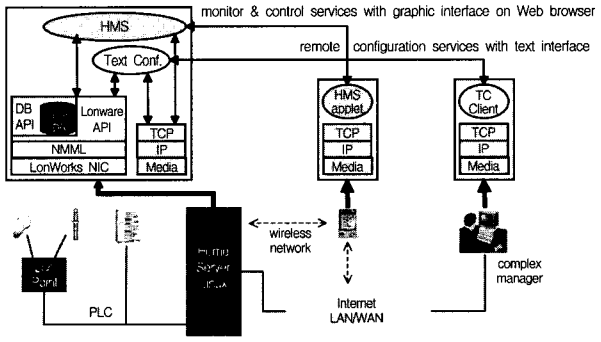
4.2 ETRI LonWare 구현

4.2.1 LonWare ver 1.0 Overview

ETRI LonWare ver 1.0은 리눅스상에서 구현된 LonWorks 구성 관리 하부 구조이다. (그림 8)은 LonWare의 위치와 전체 소프트웨어 스택을 보여준다. LonWare ver 1.0은 NMML, LonWare API, DB 모듈로 구성되어 있다. NIC는 Neuron Chip을 가지고 있는 LonWorks 네트워크 인터페이스로서, 호스트와의 인터페이스를 위해서 MIP를 이용하고 있다. NMML(Network Management Messaging Layer)은 LonWorks 네트워크에 대한 메시징 시스템이다. <표 1>과 <표 2>에서 소개한 모든 메시지에 대한 인터페이스를 가지고 있다. LonWare API는 구성관리자의 동작에 맞추어서 설계한 추상화된 사용자 프로그램 인터페이스이다. DB는 앞 단원에서 살펴본 바와 같이 디바이스 및 변수들의 정보를 저장하고 있고, 더불어 네트워크 관리를 위한 정보 논리 네트워크 주소, Selector 값 등 를 저장하고 있게 되며, 이러한 정보를 응용 프로그램들에게 제공하기 위해서 설계된 API를 함께 가지고 있다. HMS(Home Management System)와 Text Configurator는 LonWare ver 1.0을 활용해서 만들어진 응용 예제로서, HMS는 가정 내 디바이스를 활용한 정적인 서비스 설정 기능, 인터넷 상의 원격 단말에서 디바이스의 모니터 및 제어를 가능하게 하는 기능을 제공한다. Text Configurator는 전문가 수준의 Interactive한 인터페이스를 제공하는 텍스트 기반 원격 구성관리 응용 예제이다. (그림 9)는 두 응용 예제의 서비스 개념도를 보여주고 있다.



(그림 8) 홈 서버의 LonWorks 관련 소프트웨어 스택



(그림 9) ETRI LonWare의 서비스 패러다임

4.2.2 구현 환경 및 구현 범위

ETRI LonWare는 리눅스 기반의 LonWorks 구성관리 기반 소프트웨어이다. LonWare ver 1.0은 Linux(Kernel ver 2.2 이상)를 기본 운영체제로 개발되었으며, 개발 언어는 C이다. 현재 LonWare ver 1.0에서 실험된 LonWorks NIC는 독일 Gesytec의 Easylon 카드(PCI, ISA 인터페이스)와 Echelon에서 제공하는 SLTA(Serial LonTalk Adapter : RS-232C 인터페이스) 모듈이지만, MIP를 탑재한 모든 LonWorks NIC가 활용될 수 있다.

LonWare는 LonWorks 시스템의 디바이스에 대한 구성관리를 위한 하부 구조로서 설계 및 구현되었다. LonWare는 상위 응용 프로그램에게 LonWorks 시스템을 효율적으로 관리할 수 있는 인터페이스를 제공한다.

LonWare ver 1.0에서 수용할 수 있는 LonWorks 디바이스에는 제한이 없다. 그러나, LonWare ver 1.0은 LonMark에서 정의하고 있는 Functional Block을 지원하지 않는다. Functional Block은 Echelon에서 제공하는 구성관리자에서 활용하기 위한 상업적 표준의 성격이 강하다.

LonWare ver 1.0에서는 LonMark에서 최근에 추가한 Dynamic NV 기능을 지원하지 않는다. 즉, 동적으로 네트워크 변수를 할당할 수 있는 기능이 있는 디바이스에 대한 지원 기능이 구현되어 있지 않다. 그러나, LonWare가 동작하고 있는 노드 주로 홈 서버나 게이트웨이에서 동적으로 네트워크 변수를 선언하는 기능을 지원한다.

4.2.3 구현 내용

상술한 바와 같이 LonWare를 구성하고 있는 S/W 모듈은 NMML, LonWare API, DB이다. 본 단원에서는 NMML과 LonWare의 API를 설명하고, DB의 구조를 간단하게 설명한다.

4.2.3.1 NMML API

NMML(Network Management Messaging Layer)는 LonWorks 메시지를 생성하고, 전송하는 기능을 갖는다. LonWare API 계층에 존재하는 API들이 자신의 기능을 수행하려면 LonWorks 관리 시스템에서 정의하고 있는 메시지들을 활용하여 정보를 송수신해야 한다. NMML의 API들은 LonWorks에서 정의하고 있는 관리 메시지들을 패킷화하고, 이를 하부 NIC를 통해서 목적노드에 전송하는 역할을 수행한다. LonWare ver 1.0에서 구현된 NMML API들은 <표 4>와 같다. 현재 구현된 API는 모두 15개 이며, <표 4>의 마지막 2개는 노드의 상태를 쓰기 가능 상태와 컨피그 온라인 상태로 변경해 주는 인터페이스로서, LonWare에서 디바이스를 편리하게 컨트롤 할 수 있도록 한 API이다.

〈표 4〉 NMML API

NM/ND Messages	NMML API
Query ID	direct MIP interface
Respond to Query	-
Update Domain	update_domain_request
Leave Domain	-
Update Key	-
Update Address	update_address_request
Query Address	query_address_request
Query NV Config	query_nvconfig_request
Update Group Address Data	-
Query Domain	query_domain_request
Update NV Config	update_nvconfig_request
Set Node Mode	set_node_mode_request
Read Memory	read_memory_request
Write Memory	write_memory_request
Checksum Recalculate	-
Wink	-
Memory Refresh	-
Query SNVT	query_snvt_request
NV Fetch	nv_fetch_request
Device Escape Code	-
Query Status	query_status_request
Proxy Command	-
Clear Status	-
Query XCVR Status	-
	set_node_mode_to_write_enable
	set_node_mode_to_configured_online

4.2.3.2 LonWare API

LonWare ver 1.0에서는 응용 프로그램 개발자에게 의미있는 관리 API를 제공하기 위해서 <표 5>와 같은 API를 제공하며, 자세한 내용은 다음과 같다.

● NIC Open/Close 함수

LonWorks NI의 디바이스 파일과 관련된 동작을 하는 API들이다. 디바이스를 열고 닫는 작업 이외에 파일로 존재하는 DB를 메모리로 로딩하고, 프로그램을 끝낼 경우에는 다시 디스크에 파일 형태로 쓴다. 이로써 모든 LonWare를 기반으로 프로그래밍된 응용은 같은 DB 포맷을 유지하여 서로 호환이 가능해 진다.

〈표 5〉 LonWare API

구성관리 절 차	Lonware APIs	요 약
NIC Open / Close	initialization_lonware	DB 초기화 및 로드, NI open
	close_lonware	NI close, DB를 파일로 저장
네트워크 초기화	lookup_device,	Query ID 관리 메시지를 통한 네트워크 주소` 검색, DB 엔트리 할당
	get_node_info_from_node	(그림 6)
	get_node_info_from_file	XIF 파일을 이용한 노드 정보 수집
	set_node_address	새로운 논리 주소 할당
서비스 설정	bind_nv	(그림 7)
	release_bound_nv	NV 연결 해제
모니터 및 제어	get_nv_value	특정 노드의 특정 변수 값 읽기
	update_nv_value	특정 노드의 특정 변수 값 변경
동적 NV	make_dynamic_nv,	새로운 NV를 로컬에 선언
	release_dynamic_nv	로컬 NV 해제

● 네트워크 초기화 함수

구성관리의 초기 단계인 디바이스 검색(Detection), 정보수집, 논리 주소 할당의 기능을 갖는다. LonWare 1.0에서는 디바이스의 정보를 수집하는 방법으로 Upload from Device와 XIF 파일 방식을 모두 지원한다. 정보수집 과정을 완료하면 메모리에 있는 LonWare DB에 디바이스에 관한 모든 정보를 저장하게 된다.

● 서비스 설정

LonWorks 네트워크 상에 물리적으로 연결된 디바이스에 대한 모든 정보가 수집된 후에는 사용자가 요구하는 연동 서비스를 제공해 주기 위한 서비스 설정을 수행해야 한다. LonWorks 네트워크에서 제공해 주는 연동 서비스는 모두 NV들의 연결(Binding)작업에 의해서 이루어진다. LonWare 1.0에서는 NV의 1 : 1 연결을 지원하는 API를 제공한다.

● 모니터 및 제어

연동 서비스 이외에 제어 네트워크 뿐만 아니라 모든 홈 네트워크 분야에서 기본적으로 제공되어야 할 서비스가 모니터 및 제어 서비스 이다. LonWare 1.0에서는 모니터 및 제어 서비스를 LonWorks 네트워크 관리 메시지(nv_fetch)

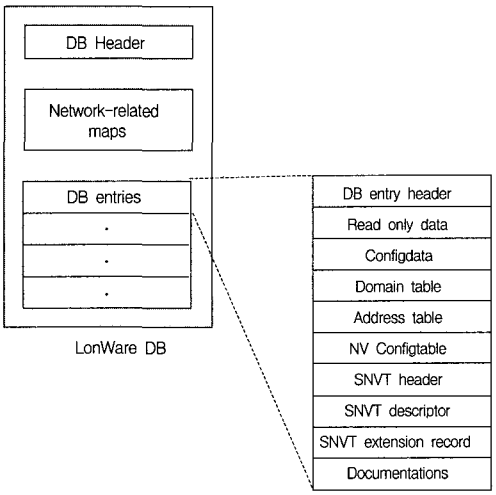
와 NV 메시지(update_nv) 메시지를 통해서 수행한다.

● 동적 NV 할당

LonWare 1.0의 동적 NV할당은 앞에서 기술한 바와 같이 로컬 호스트, 즉 LonWare를 기반으로 하는 구성관리자가 동작하고 있는 호스트(ex. Home Server)에 동적으로 네트워크 변수를 선언하고, 사용할 수 있게 하는 기능을 말한다. 동적으로 선언된 변수는 디바이스의 상태를 모니터 하는데 있어서 융통성을 제공할 수 있다. 즉, 구성관리자에서 디바이스의 상태를 모니터하기 위해서는 폴링을 이용해야 하는 현재의 방식을 개선하여, 디바이스의 상태가 변화될 때마다 디바이스의 메시지를 받아서 알려줄 수 있는 NV를 선언 함으로서 인터럽트 방식으로 디바이스의 변화를 모니터 할 수 있다.

4.2.3.3 Data Base

LonWare 1.0의 DB는 LonWorks 디바이스가 가지고 있는 모든 자료구조를 포함해야 한다. 또한, LonWorks 네트워크를 관리하기 위한 자료도 저장하고 있어야 한다. (그림 10)은 LonWare DB의 구조를 보여주고 있다.



(그림 10) LonWare DB 구조

DB Header는 현재 DB에 저장된 디바이스의 개수를 나타낸다. Network-related maps 영역에는 네트워크를 관리하는데 필요한 정보들을 포함하고 있다. 구성관리자는 NV의 연결(Binding) 작업을 수행할 때 새로운 NV Selector를 할당받아서 사용해야 하는데, 전체 네트워크에 유니크한 Selector값을 할당받아야 한다. 이를 위해서는 현재 할당된 Selector값이 무엇이며, 그렇지 않은 값은 무엇인지를 알 수 있는 MAP의 관리가 필요하다. Network-related maps 영역에는 이러한 값들이 들어가 있다. 또한, 네트워크의 논리주소를 할당할 때에도 주소의 MAP이 필요하며 이 자료구조 역시 이 영역에 포함된다.

DB entry 필드에는 디바이스가 가지고 있는 모든 주요 자료구조의 정보를 가지고 있다. DB entry header 필드에는 디바이스 정보 중에 접근이 가장 많이 되는 디바이스의 NV수, 네트워크 ID, Host-based 노드 여부 등의 정보를 Cache 하여 저장하고 있다. 나머지 필드들은 디바이스가 가지고 있는 4.1.1에서 살펴본 자료구조를 그대로 가지고 있다.

5. 결론 및 향후 연구 방향

LonWare 1.0의 구현은 리눅스를 기반으로 개발되고 있는 많은 홈 서버와 게이트웨이, 그리고 세톱박스와 같은 홈 네트워크 서비스 지원 장비에 홈 제어 네트워크 서비스를 지원할 수 있도록 하는 중요한 인프라 역할을 할 수 있다. 또한, 가정 내의 홈 서버 류 장비에 가정의 홈 제어 네트워크 디바이스들에 대한 DB를 보관 함으로서, 인터넷 상의 구성관리자가 DB를 관리하여 가정 내 디바이스를 제어 했던 보안상의 문제점을 해결할 수 있다. 또한, 응용 프로그램의 개발을 용이하게 할 수 있는 API를 지원 함으로서, 다양한 방법으로 사용자의 서비스 요구를 표현할 수 있도록 해주고 있다.

현재 ETRI에서는 LonWorks 시스템의 Neuron Chip에 하드웨어적으로 구현되어 있는 LonTalk 프로토콜을 S/W적으로 구현하는 작업을 진행하고 있으며, 또한 Neuron C 개발 환경을 대체할 수 있는 응용 프로그래밍 환경을 설계하고 있다. 이러한 일련의 작업이 완성되면 Java 프로그램을 이용한 디바이스 응용 프로그램이 동적으로 디바이스에 로딩되고, 업그레이드될 수 있는 환경이 구축 될 것으로 기대된다.

참 고 문 헌

[1] Echelon Co., "Neuron Chip Data Book," Feb., 1995.
[2] Echelon Co., "LonTalk Protocol Specification ver 3.0," 1994.
[3] LonMark Interoperability Guideline, "LonMark Application Layer Interoperability Guidelines," 1999.
[4] Echelon Co., "LonMark External Interface File Reference Guide, revision 4.0A," Apr., 2000.
[5] Motorola Inc., "LonWorks Technology Device Data, revision 5," 1998.
[6] Echelon Co., "Neuron C Programmer's Guide".
[7] 황승구, "인터넷 정보가전 동향", 정보처리학회지, 제8권 제1호, pp.17-27.
[8] 문경덕 외, "홈 네트워크 제어 미들웨어 개요 및 표준화 동향", 정보처리학회지, 제8권 제5호, pp.45-52.
[9] 문경덕 외, "인터넷 정보가전 미들웨어 기술 소개", 한국통신학회지, 제18권 제12호, pp.96-104.

박 준 희

e-mail : juni@etri.re.kr

1995년 충남대학교 컴퓨터과학과(학사)

1997년 충남대학교 대학원 컴퓨터과학과 (석사)

1997년~1999년 시스템공학연구소 네트워크컴퓨팅연구부 연구원

1999년~현재 한국전자통신연구원 정보가전연구부 선임연구원

관심분야 : 이동 Ad Hoc 네트워크, 홈 네트워크, 홈 네트워크 미들웨어, Pervasive Computing

손 영 성

e-mail : ysson@etri.re.kr

1997년 부산대학교 전자계산학과(석사)

1997년~현재 ETRI 근무

관심분야 : 분산컴퓨팅, 클러스터컴퓨팅

문 경 덕

e-mail : kdmoon@etri.re.kr

1990년 한양대학교 전산학과(학사)

1992년 한양대학교 전산학과(석사)

1992년~1997년 시스템공학연구소 연구원

1997년~2000년 한국전자통신연구원 선임연구원

2000년~현재 한국전자통신연구원 정보가전 제어 S/W 연구팀장

관심분야 : 홈 네트워크 미들웨어, 액티브 네트워크, 실시간 Java, Pervasive Computing