JNI를 이용한 MMS 구현

장 경 수† · 신 동 렬††

요 약

MMS(Manufacturing Message Specification)는 PLC, NC, 로봇 등과 같이 서로 다른 제조회사의 서로 다른 단위제어기기 제품들간에 통신할 수 있는 ISO/IEC 9506으로 표준화된 공장자동화용 프로토콜이며 OSI 참조 모델의 최상위 층인 응용계층 프로토콜에 해당된다. 본 논문은 MMS를 TCP/IP상에서 동작할 수 있도록 유닉스 환경에서 ANSI-C 언어로 구현하고, 이 구현된 프로토콜을 JNI(Java Native Interface)를 이용해 JAVA 클래스화한다. JAVA 클래스화함으로써 기존에 제공되는 MMS 라이브러리를 이용하는데 있어 표준화되지 않은 서로 다른 API를 이용하는데 어려움과 GUI를 구현하는데 어려움을 극복하는 기본을 제공한다. 그리고 구현된 JAVA 클래스화된 MMS 라이브러리를 인터넷의 WWW상에서 동작시킬 수 있도록 자동화된 PCB(Printed Circuit Board) 조립라인을 대상 모델로 선정하여 응용 프로그램을 작성하여 구현된 JNI를 이용한 MMS가 인터넷상에서 동작하여 사용자에게 일관성있는 인터페이스를 제공하는 웹 브라우저를 통해 RMD(Real Manufacturing Device)를 동작·제어·감시할 수 있음을 보여준다.

Implementation of MMS using JNI

Kyung-Soo Jang† · Dong-Ryeol Shin††

ABSTRACT

Manufacturing Message Specification (MMS) is designed as a communication standard protocol, ISO/IEC 9506, on factory automation for messaging between heterogeneous programmable unit controller, PLC, NC, Robot, of different vendors on the networks. MMS is also a standard protocol of OSI reference model application layer. In this paper, we show an implementation of MMS over TCP/IP using ANSI-C programming language on the unix environment, and make java classification using java native interface (JNI) with MMS library. The use of java classification provides a basic environment to overcome a difficult programming with different MMS application programming interface (MMS-I) which requires a skilled programming technique of graphic user interface (GUI). In this paper, we implement a MMS application program of the automated assembly model for printed circuit board based on WWW which shows the operation, control and monitoring of real manufacturing device (RMD) with web browser providing users for consistent user interface.

1. 서 론

최근 정보처리분야의 네트워크 기술 도입으로 제어시스템 기술은 생산 공정의 자동화 분야에서 유연성을 갖게 되었다. 프로그래밍이 가능한 단위제어기기(programmable unit controller : 이하 단위제어기기)인 PLC(Programmable Logic Controller), 로봇(robot), NC(Numerical Controller)가 공장 자동화에 중추적인 역할을 담당하여 생산성 향상, 원가절감 등의 효과를 가져왔다. 단위제어기기들에 대한 보다 효율적인 운용과 자원을 공유

※ 본 연구는 기초전력공학공동 연구소(과제번호 : 97-039)의 연구비 지원 결과입니다.
† 준 회 원 : 성균관대학교 대학원 전기전자 및 컴퓨터공학부
†† 정 회 원 : 성균관대학교 전기전자 및 컴퓨터공학부 교수
논문접수 : 1999년 7월 16일, 심사완료 : 1999년 12월 7일

할 필요성이 요구되어 단위제어기기들을 상호 연결할 네트워킹 기술이 크게 증가되고 있다[1]. 그러나, 이러한 단위제어기기들은 각각의 제조회사마다 고유한 제어프로그램과 프로토콜을 제공하여 서로 다른 제조회사의 같은 기능을 갖는 기기 간에 통신하는 것이 어려울 뿐 아니라 서로 다른 제조회사의 서로 다른 기기간의 통신은 더욱더 어려운 상태에 이르게 되었다.

더욱이 컴퓨터 통합 생산(Computer-Integrated Manufacturing : CIM) 구현에서 가장 중요한 것 중 하나가 단위제어기기간의 상호운용성(interoperability)이다. 이러한 요구에 따라 ISO(International Organization for Standardization)에서는 ISO/IEC 9506인 MMS(Manufacturing Message Specification)를 발표했다. MMS는 CIM 환경에서 단위제어기기간에 서로 메시지 교환을 위한 OSI(Open Systems Interconnection) 참조모델의 응용계층 표준이다. MMS는 80년대 GM(General Motors)사에서 서로 다른 제조회사의 단위제어기기 제품들 간에 통신을 위해 발표된 MAP(Manufacturing Automation Protocol)이 그 시초이며 MAP의 최상위 계층의 프로토콜이다[2].

MMS는 OSI 상에서 뿐 아니라 Full-MAP, Mini-MAP 등의 환경에서 모두 지원되고 구현될 수 있지만, 현재는 TCP/IP(Transmission Control Protocol/Internet Protocol) 상에서 많이 구현되고 있다. 'MMS over MAP'이나 'MMS over Mini-MAP'은 비용 면에서 고가이고, OSI는 현재 대중화가 되어있지 않기 때문에 'MMS over TCP/IP'는 위 두 단점을 잘 보완해줄 수 있는 적절한 선택이며, 본 논문에서는 TCP/IP 상에서 MMS를 구현한다.

인터넷(Internet) 사용자의 증가는 TCP/IP를 대중화시켰고, 인터넷에는 다양한 서비스가 존재한다. 인터넷 서비스 중에 특히 WWW(World Wide Web) 서비스는 고퍼(gopher)와 같은 다른 인터넷 서비스와 달리 텍스트, 이미지, 사운드, 동영상 등과 같은 멀티미디어 형식이 포함된 정보를 서비스 받을 수 있음으로써 인터넷 사용자들이 텍스트 환경이 아닌 그래픽 환경에서 사용자에게 일관성 있는 인터페이스를 제공하는 웹 브라우저(web browser)를 이용해 서비스를 이용할 수 있다. 자바 프로그래밍 언어는 이러한 WWW 서비스 제공을 위해 사용되는 HTML(HyperText Markup Language) 언어의 한계를 극복하기 위해 많은 프로그래머들의 관심의 대상이 되고 있으며 자동화 분야에서의 적용도 늘어나고 있는 추세에 있다[3]. 본 논문에서는 인터넷의 WWW을 통해 단위제어기기를 제어하고 감시할 수 있도록 MMS를 자바 네이티브 인터페이스(Java Native Interface : JNI)[4, 5]를 이용해 구현하는데 그 목적이 있다.

구현 환경은 Sun SPARC 워크스테이션을 머신으로 선택했으며, 자바 애플릿의 실행은 웹 브라우저를 통해 이루어지므로 대상 머신뿐 아니라 웹 브라우저가 탑재된 어떤 머신에서도 실행될 수 있도록 구현한다. 운영체제는 Solaris 2.4에서 구현되며 자바 API(Application Programming Interface)를 지원할 수 있는 플랫폼에서 구현한다. 사용 언어는 JNI를 이용하기 전의 MMS는 표준 C 프로그래밍 언어인 ANSI-C를 이용하며, C언어로 구현한 MMS를 JNI를 이용해 자바 클래스화하기 위한 JDK(Java Development Kit) 1.1.6과 웹 페이지 작성을 위해 HTML을 이용한다. 사용 프로토콜은 초기의 웹 페이지와 자바 애플릿 전송을 위해 HTTP (HyperText Transfer Protocol)를 사용한다. 실제 단위제어기기간의 통신은 MMS 프로토콜을 이용한다. 사용자 인터페이스의 화면표시와 서비스 요청을 위한 프로토콜로써 자바 소켓을 사용한다.

논문의 구성은 먼저 MMS 프로토콜과 MMS 환경 구현, JNI를 이용한 MMS를 설계한다. 자바 프로그래밍 언어와 연결한 MMS를 구현하는 방법 제시와 JNI를 이용한 MMS를 구현하고 자동화된 PCB(Printed Circuit Board) 조립라인모델을 대상으로 응용 프로그램을 개발하여 인터넷상에서 MMS 프로토콜을 이용해 단위제어기기를 동작·제어·감시할 수 있음을 검증한다.

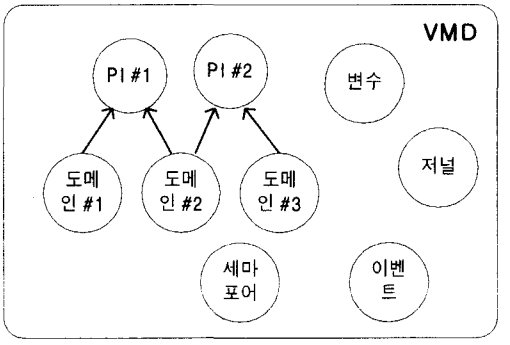
MMS 구현에 따라서는 응용 프로그램을 도스 모드[6, 7]에서 동작할 수 있도록 하거나 윈도우 모드[8]에서 동작할 수 있도록 할 수 있지만 JAVA를 이용한 구현은 본 논문에서 처음 시도되는 것이다. 이것은 응용 프로그램 작성시 GUI를 구현하는데 어려움을 해결할 수 있는 기반을 제공할 뿐 아니라 사용자가 웹 브라우저를 이용함으로써 사용상의 편리를 함께 제공한다.

2. MMS 프로토콜

MMS는 CIM 환경에서 서로 다른 제조회사의 서로 다른 단위제어기기간에 메시지 교환을 위한 ISO/OSI 참조모델의 응용계층의 공장자동화용 프로토콜로 ISO/IEC 9506-1(이하 Part 1)과 ISO/IEC 9506-2(이하 Part

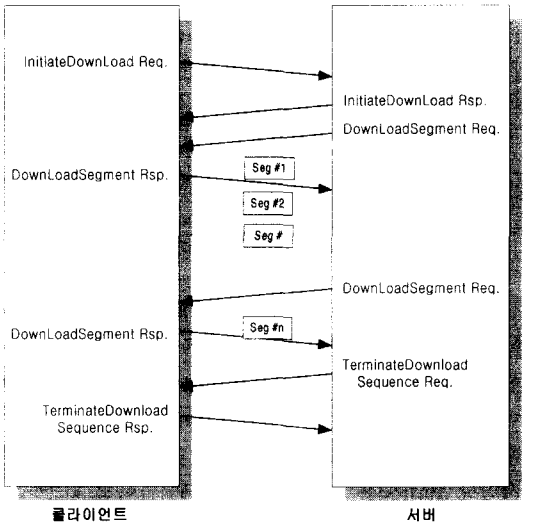
2)가 그 핵심 규약이다. Part 1은 서비스 사양에 대한 문서로 가상제조기기(Virtual Manufacturing Device : VMD), 네트워크 상에서 교환되는 메시지의 정의, VMD에 관련된 속성과 파라미터에 대해 기술한다[9]. Part 2는 프로토콜의 사양을 표현 계층(presentation layer)의 추상구문표현 1(Abstract Syntax Notation 1 : ASN.1)[11]으로 기술되어 있다[10]. Part 1과 Part 2 외에 부가표준안(companion standard)[6]을 두고 있어 PLC, 로봇, NC와 같은 특정 장비에 한정된 상호운용성에 관한 정보를 담고 있다. Part 1과 Part 2가 MMS의 전반적인 내용을 포함하고 특정 단위제어기기에 국한되지 않은 전체 단위제어기기를 포괄해서 기술하고 있기 때문에 본 논문에서는 부가표준안에 대한 구현은 포함하지 않는다.

MMS는 클라이언트-서버 모델(client-server model)을 기본으로 하고 있으며, 클라이언트 시스템과 서버 시스템의 통신은 구조화된 데이터 블록인 PDU(Protocol Data Unit)를 이용해 이루어진다. MMS에서는 VMD를 포함하고 있는 PLC, 로봇, NC와 같은 단위제어기기인 실제 제조 기기(Real Manufacturing Device : RMD)를 서버로 보며, 서버 시스템은 클라이언트 시스템의 서비스 요청에 대한 행동(action)을 취한 후 그것에 대해 응답한다. 클라이언트 시스템은 MMS 서비스를 통해 단위제어기기를 제어·감시한다.



(그림 1) 가상제조기기 객체 모델

MMS에서는 RMD의 모든 기능과 자원을 VMD 객체로 추상화시켜 표현한다. (그림 1)은 여러 개의 객체를 포함하고 있는 VMD를 표현하였다. VMD는 실행 가능한 함수로 그 내부에는 RMD의 논리적, 물리적 기능과 역할에 따른 도메인(domain), 프로그램 수행(Program Invocation : PI), 변수(variable) 등과 같은 객체를 정의한다. RMD의 기능과 자원에 매핑된 도메인, PI를 VMD 객체 내에 포함시킨다. VMD는 이들 객체 중 하나 혹은 그 이상의 객체들의 집합으로 볼 수 있다. 여기서 PI 객체는 도메인 객체의 집합으로 정의할 수 있다. RMD의 논리적, 물리적 자원과 능력에 매핑된 하나 이상의 도메인들의 집합을 PI 객체화할 수 있는데, 이 PI 객체는 미리 정의될 수도 있고 혹은 'Creat Program Invocation'이라는 MMS 서비스에 의해서 동적으로 생성될 수 있다. VMD 객체에 접근하는 것은 MMS 클라이언트 서비스에 의해서 가능하다.



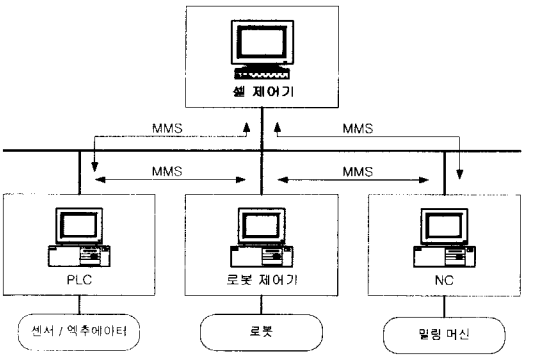
(그림 2) MMS 서비스 이용 예

(그림 2)는 MMS 서비스를 이용해 도메인을 다운로드하는 예를 보여주고 있다. 서비스를 요청하는 MMS 클라이언트는 VMD를 통해 단위제어기기인 RMD를 일관성 있게 보고 제어, 감시할 수 있다. 각 RMD에 매핑되는 VMD는 RMD마다 1개 이상 존재할 수 있다. 결국 MMS 프로토콜은 RMD의 기능과 자원을 VMD로 추상화시키고 MMS 서비스를 통해 VMD에 접근함으로써 RMD를 제어할 수 있는 것이다.

3. MMS 환경 구현

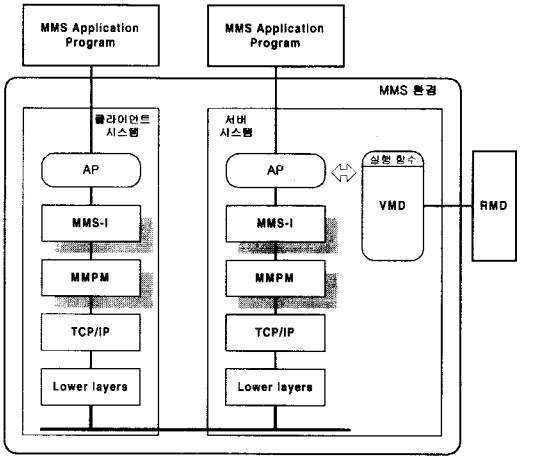
본 논문에서는 OSI 모델에 비해 대중화되어 있는 TCP/IP를 MMS구현을 위해 선택했다. 즉, 'MMS over TCP

/IP'를 MMS 구현의 환경으로 선택했다. (그림 3)은 CIM에서의 MMS의 역할을 보여주고 있다. 그림에서 셀 제어기가 있는 층을 셀 계층이라고 하고 그 상위의 계층은 표현되지 않았다. 셀 제어기가 공통된 언어인 MMS를 이용해 PLC, 로봇 등과 같은 RMD를 제어·감시하는 것이다. 또, RMD와 RMD 간에도 MMS를 이용해 통신할 수 있다.



(그림 3) CIM에서의 MMS

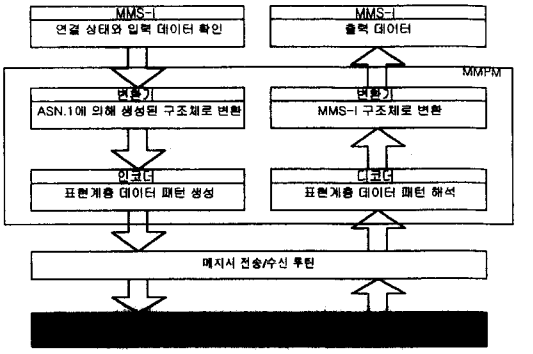
(그림 4)는 설계한 MMS의 전체 환경을 나타낸다. MMS 클라이언트가 MMPM(MMS Protocol Machine)에게 서비스 요청을 하면 MMPM은 클라이언트가 요청한 서비스에 해당되는 MMS-Request PDU를 생성한 후 네트워크를 통해 MMS 서버의 MMPM에게 보낸다. 서버의 MMPM은 요청한 PDU를 해석하여 응용 프로세스(Application Process : AP)에게 전달한다. AP는 요청한 서비스에 대해 VMD의 속성을 바꾸거나 VMD로부터 상태정보 등을 얻는 행동을 취한다. 행동에 대한 결과를 서버 MMPM에게 보내고, 서버 MMPM에서는 결과에 해당되는 MMS-Response PDU를 생성한 후 네트워크를 통해 서비스를 요청한 클라이언트 MMPM에게 전달한다. 통신하는 중에 에러가 발생하면 자세한 에러의 내용을 보내기 의해 MMS-Error PDU를 보내지만 본 논문에서는 모든 통신이 정상적으로 이루어진다고 가정하였다. VMD는 구조체를 중심으로 RMD의 자원을 구현하였고, VMD의 외부에서 보이는 행동, 즉 RMD의 행동을 모듈로서 구현하여 MMS를 지원하는 RMD를 연결함으로써 곧바로 통신을 할 수 있도록 했다.



(그림 4) MMS 환경 구현

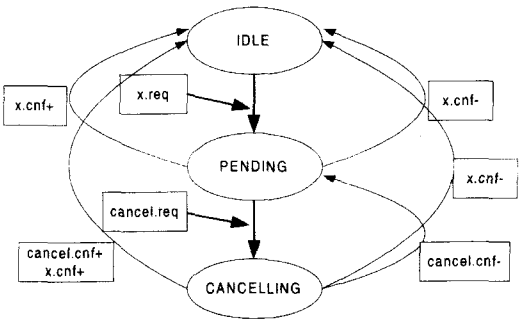
3.1 MMPM의 구현

MMPM은 MMS 환경에서 서비스를 담당하여 PDU의 생성과 해석을 하는 가장 핵심적인 부분이라 할 수 있다. (그림 5)는 MMPM을 통해 MMS 클라이언트로부터의 서비스 요청을 서버에게 메시지를 전송하는 루틴을 구현한 구조를 나타낸 것이다. MMS 클라이언트로부터 받은 서비스 요청을 ASN.1에 의해 생성된 데이터 구조체 형식으로 변환한 후 BER(Basic Encoding Rules)[12] 인코더(encoder)에 의해서 표현계층 데이터 형식으로 변환한다. 즉, MMS 클라이언트와 서버가 공통된 신택스(syntax)를 사용하여 정보를 양쪽 모두 이해할 수 있도록 데이터를 암호화한다. 변환된 데이터는 메시지 전송 루틴에 의해서 하위 프로토콜 스택으로 보낸다. 서비스 요청에 대한 응답 메시지 전송은 반대로 볼 수 있다. 그렇기 때문에 구현에서는 변환기와 디코더, 인코더 부분을 MMPM의 각 시스템 내에 각각 두었다.



(그림 5) MMPM을 통한 메시지 처리 루틴

(그림 6)은 구현된 MMPM의 클라이언트 상태 천이도를 표현한 것이다. IDLE상태는 클라이언트 MMS 사용자로부터 어떤 서비스 요청도 받지 않은 상태이고, PENDING상태는 클라이언트 MMS 사용자로부터 확인형(confirmed) 서비스를 요청 받고, 요청 PDU(request PDU)를 생성해 다른 MMPM에게 보낸 후 확인형 응답 PDU(response PDU)를 기다리는 상태이다. 그리고, CANCELLING상태는 이미 확인형 서비스를 요청하고 있으나 응답 PDU를 받지 못한 상태에서 이미 요청한 서비스에 대한 취소 요청(cancel.req)을 클라이언트 MMS 사용자로부터 받은 후 그 응답 PDU를 기다리는 상태를 나타낸다. x.cnf+ 등의 +는 정상적으로 응답을 받았을 때를 나타내고, x.cnf-에서와 같이 -는 에러가 감지됐거나 요청한 자원을 사용할 수 없을 경우를 표시한다. x.cnf는 MMS 확인형 서비스를 나타내고, cancel.req는 'Cancel' 서비스 요청을 나타낸다. x.cnf-, cancel.cnf-는 x.ErrorPDU와 cancel.ErrorPDU를 전송해야 하지만 본 구현에서는 통신 중에 에러가 없는 것으로 가정했기 때문에 구현에서 제외되었다. IDLE에서 PENDING 상태로 천이되는 경우는 MMS 사용자로부터 확인형 서비스를 요청 받아 요청 PDU를 생성해 서버 MMPM에게 전송했을 때이다. 서버 MMPM으로부터 어떤 응답을 받았을 때 비로소 PENDING에서 IDLE상태로 천이된다. PENDING중에 클라이언트 MMS 사용자로부터 서비스 취소 요청을 받았을 때, 앞에 요청한 서비스를 취소하지 못하고 그에 대한 응답을 받거나 취소 요청에 대한 정상적인 응답을 받으면 IDLE상태로 천이한다.



(그림 6) MMPM 상태 천이도

(그림 7)은 호출하는 프로토콜 머신의 ASN.1으로 표현된 READ 서비스의 일부를 나타낸 것이다. 이것을 구조체로 변환한 것이 (그림 8)이며 각 MMS 서비스 구현시 참조하도록 했다. 이 변환된 구조체는 MMPM의 상태 천이에 관계된 모듈뿐 아니라 인코딩과 디코딩을 위한 모듈에도 같이 쓰인다.

```
Read-Request ::= SEQUENCE {
  specificationWithResult     [0] IMPLICIT BOOLEAN DEFAULT FALSE,
  variableAccessSpecification [1] VariableAccessSpecification
}

VariableAccessSpecification ::= CHOICE {
  listOfVariable   [0] IMPLICIT SEQUENCE OF SEQUENCE {
    variableSpecification VariableSpecification,
    alterAccess         [5] IMPLICIT AlternateAccess OPTIONAL
  },
  variableListName [1] ObjectName
}
```

(그림 7) Read 서비스 일부의 ASN.1 표현

MMS는 MMPM과 함께 MMS-I 함수들을 라이브러리 형태로 사용자에게 제공된다. 물론 변환기와 BER 인코더/디코더 부분을 포함해서이다. 사용자 응용 프로그래머는 MMS-I 함수들을 자동화 라인의 형태와 기능에 따라서 적절히 함수호출하여 응용 프로그램을 작성할 수 있다.

```
typedef struct Read_Request  /* SEQUENCE */
{
   AsnBool* specificationWithResult; /* IMPLICIT BOOLEAN DEFALT FALSE
*/
   struct VariableAccessSpecification* variableAccessSpecification;
} Read_Request;

typedef struct VariableAccessSpecification /* CHOICE */
{
   enum VarableAccessSpecificationChoiceId
   {
      VARIABLEACCESSSPECIFICATION_LISTOFVARIABLE,
      VARIABLEACCESSSPECIFICATION_VARIABLELISTNAME
   } choiceId;
   union VariableAccessSpecificationChoiceUnion
   {
   VariableAccessSpecificationSeqOf* listOfVariable;
   struct ObjectName* variableListName; /* [1] ObjectName */
   } a;
} VariableAccessSpecification;
```

(그림 8) 구조체로 변환된 Read 서비스의 일부

3.2 하위 프로토콜 스택과의 통신

본 구현에서는 하위 계층이 TCP/IP를 사용하므로 소켓 API(Socket Application Programming Interface)를 이용해 연결 설정한 후 하위계층으로 PDU를 전송하도록 설계·구현하였다. 소켓 API는 유닉스에 기본적으로 포함된 시스템 함수를 이용했다. 본 논문에서는 Solaris 2.4를 운영체제로 하는 워크스테이션에서 MMS를 구현하였지만, 표준 C 언어로 작성되었기 때문에 MMS 라이브러리 자체는 플랫폼에 상관없이 동작할 수 있다. 하

위 프로토콜 스택이 TCP/IP가 아닌 경우는 연결 설정 부분만을 수정·구현함으로써 MMS 프로토콜을 이용해 통신할 수 있도록 했다.

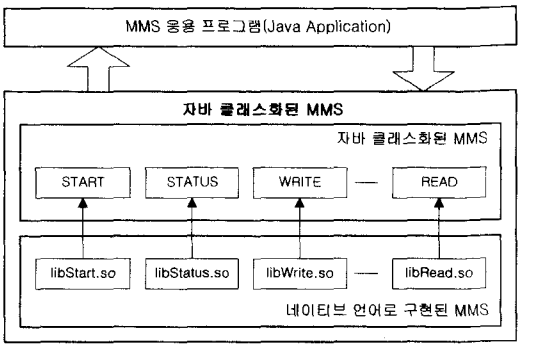
4. JNI를 이용한 MMS 구현

대부분의 MMS 제품이 MMS-I 함수들을 라이브러리 형태로 제공한다. 이것들을 이용해 응용 프로그램을 작성하는데는 제공되는 라이브러리에서 사용된 프로그래밍 언어에 능숙해야될 뿐 아니라 MMS의 개념을 알고 있어야만 주어진 환경 즉, 자동화 라인에서 필요한 프로그램을 작성할 수 있다. MMS의 원래 취지를 RMD간의 상호운용성, 비용절감, 신속한 프로그래밍을 통한 유연 생산 등으로 들 수 있는데, MMS의 사용자 응용 프로그램을 작성하는데 있어 MMS를 이해하는데 많은 시간을 소비한다면 신속하게 응용 프로그램을 작성하기는 어려울 것이며 비용도 증가하게 될 것이다. 또한, 자동화 라인을 동작시키기 위한 응용 프로그램을 작성하는데 급급하다보면 응용 프로그램에 사용자 인터페이스를 제대로 제공하는데 어려움이 따른다.

본 논문에서는 이러한 어려움을 조금이라도 해결할 수 있는 해결책으로써 C 언어로 구현된 MMS에 JNI를 이용해 인터페이싱하여 자바 클래스화된 MMS 라이브러리를 제공하여 WWW을 통해 MMS를 이용해 RMD를 제어·감시할 수 있는 기반을 마련하고자 한다. MMS를 자바로 전체 구현하지 않은 이유는 네이티브 언어(본 구현에서는 C언어)에 비해서 자바가 처리 속도 면에서 늦기 때문이다[4]. 또 다른 이유는 네이티브 언어와 자바 언어로 된 두 가지 MMS 라이브러리를 사용자에게 제공함으로써 프로그래밍 언어 선택의 폭을 넓히기 위해서이다. 물론 네이티브 언어로 작성된 라이브러리와 JNI를 적용했을 경우의 네이티브 언어 라이브러리는 같지 않다.

4.1 기본 구조와 JNI 적용 방법

(그림 9)는 네이티브 언어로 구현된 MMS 라이브러리를 자바에서 로딩(loading)하는 것을 보여준다. 맨 하위 층이 네이티브 언어로 구현된 MMS 라이브러리를 나타내고 이것을 이용해 자바 클래스화 하였다. 이 자바 클래스화된 MMS 라이브러리는 MMS 응용 프로그램 작성시 필요에 따라 적절히 호출하여 이용된다. 자바 클래스화된 MMS 층의 'START', 'STAUS' 등은 MMS에서 서비스 이름으로 자바 어플리케이션(Java Application)에서 쉽게 서비스 식별을 할 수 있게 하였다.



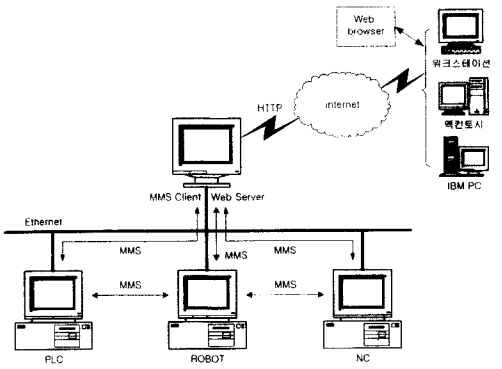
(그림 9) MMS-C Lib.와 자바와의 연결

본 구현에서는 네이티브 언어를 이용해 먼저 MMS를 구현하였다. 따라서 [4]에서 제시하는 순서와 다르다. 네이티브 언어로 작성된 MMS는 그대로 두고, MMS 서비스 이름을 클래스이름으로 하는 네이티브 메쏘드(native method)를 선언한 자바 파일을 구현한다. JDK (Java Development Kit) 패키지에 포함되어 있는 javah를 이용해 헤더와 스터브(stub)를 생성한다. 그런 후 앞서 생성된 헤더를 네이티브 언어에 포함시키고, JNI 사양서[5]에 따라 네이티브 언어로 작성된 MMS를 수정한다. 본 구현에서는 유닉스 시스템을 이용하였기 때문에 수정된 MMS는 ld(link editor for object files : man page 참조)를 이용해 *.so를 생성한다. *.so는 libxxx.so의 형식을 취한다. xxx는 자바 언어에서 로딩하는데 쉽게 구별하기 위해 서비스 이름으로 하였다. libRead.so는 READ 서비스의 네이티브 언어로 구현된 MMS 라이브러리라는 것을 나타낸다. 이것은 자바 클래스의 네이티브 메쏘드내에서 로딩하게 된다.

4.2 실행 환경 및 접근 구조

자바 클래스화된 MMS를 이용하기 위한 응용 프로그램은 유닉스 환경에서 작성되어야하며, (그림 10)은 구현된 MMS 라이브러리를 이용한 사용자 응용 프로그램에 접근하여 MMS를 이용해 RMD를 동작·제어·감시하는 모델을 보여준다.

MMS에 의해 통신하는 시스템은 (그림 10)의 MMS Client와 각 RMD의 제어기 부분이다. MMS 클라이언트는 CIM에서 셀 제어기에 해당된다. 동시에 WWW

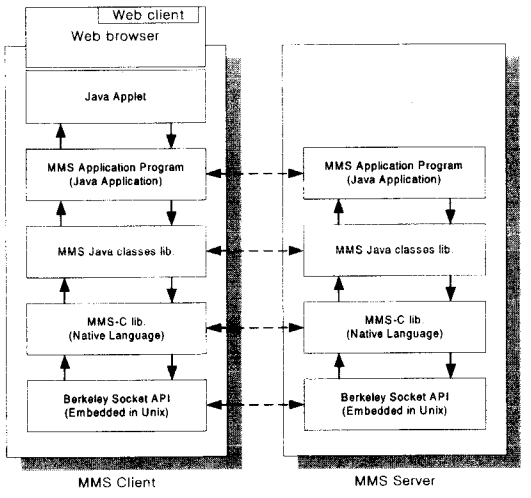


(그림 10) WWW을 통한 MMS 이용 모델

클라이언트에 대해서는 WWW 서버로써의 역할을 한다. 제 3의 컴퓨터에서 셀 제어기를 제어·감시함으로써 셀 제어기에 연결된 RMD을 동작시키고 감시할 수 있다. 기존의 셀 제어기 운영자는 셀 제어기가 위치한 곳에서만 오퍼레이션을 할 수 있었다. 그러나 본 구현을 통해 사용자(오퍼레이터 : Operator)는 위치에 상관없이 인터넷이 연결된 어떤 곳에서도 제어와 감시가 가능하다.

4.3 클라이언트

구현된 응용 프로그램의 구조는 (그림 11)에서 보여주고 있으며, 웹브라우저는 원격지에서 접속할 수도 있고, MMS 클라이언트 시스템 자체에서도 사용할 수 있다.



(그림 11) 응용 프로그램의 메시지 전달 구조

웹브라우저와 자바 애플릿과의 통신은 HTTP를 사용한다. 자바 애플릿과 자바 어플리케이션의 통신은 자바 소켓을 사용하였다.

본래 자바에서는 보안의 이유로 웹 브라우저에서만 동작할 수 있는 자바 애플릿과 디렉토리의 파일에 접근할 수 있는 어플리케이션으로 나누어져 있기 때문에 분리하여 구현하였다. 여기서 자바 어플리케이션 부분이 MMS 사용자 응용 프로그램에 해당된다. 사용자의 환경과 생산 현장의 특성에 따라 바뀌어지는 부분이다. 사용자 응용 프로그램은 꼭 자바 애플릿과 연계할 필요는 없다. 자바 어플리케이션 자체에도 AWT(Abstract Windows Toolkit)[4, 13] 패키지를 이용해서 GUI를 구현할 수 있다. 자바에서 제공하는 AWT 패키지에는 GUI를 위한 여러 가지 클래스를 포함하고 있다. AWT에는 버튼, 리스트, 메뉴와 같은 여러 종류의 컴포넌트(component)들이 있으며, 윈도우, 패널, 메뉴바와 같은 컨테이너(container)를 제공한다. 이밖에도 그래픽, 이벤트, 폰트, 이미지 등과 같은 클래스를 포함하고 있다. 그러나 이것을 이용해 구현한 GUI는 단지 로컬에서만 실행될 수 있으며 분산된 환경에서는 이용할 수 없다. 따라서, 본 논문의 응용 프로그램의 구현에서는 RMD 오퍼레이터의 위치에 상관없고 웹 브라우저를 이용해 일관성있는 사용자 인터페이스를 제공하기 위해 애플릿 부분도 추가하여 구현하였다.

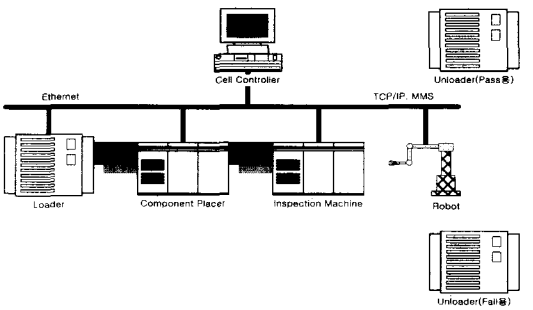
동작은 먼저 웹 클라이언트는 웹 서버인 MMS 클라이언트 시스템과 연결을 설정한다. 웹 클라이언트는 MMS 클라이언트 시스템으로부터 자바 애플릿을 읽어들인다. 웹 클라이언트에서 사용자가 RMD를 동작시키기 위해서는 내려 받은 애플릿을 통해 서비스를 선택하여 요청한다. 이 요청은 MMS 클라이언트에 전송된다. 요청을 받은 MMS 클라이언트 시스템에서는 그것이 MMS의 어떤 서비스인가를 분별한 후 분별된 서비스에 따라 네이티브 언어로 작성된 MMS 라이브러리 함수를 호출한다. 네이티브 언어로 작성된 MMS 라이브러리에서 ASN.1에 따라 생성된 구조체로의 변환과 BER에 따른 인코딩이 이루어진다. 생성된 비트 패턴은 메시지 전송 루틴에 의해 하위 프로토콜 스택으로 전송되어 MMS 서버에 전송한다. 여기서 실제 MMS PDU가 생성되어 전송되는 부분은 'MMS-C lib.'이다. 'MMS-C lib.'내에서는 MMS-I 함수들, 구조체 변환기, 인코더, 그리고 인코딩된 패턴의 하위 프로토콜 스택으로 전송을 담당하는 모듈을 포함하고 있다.

4.4 서버

MMS 서버 시스템은 클라이언트와 비교했을 때 자바 애플릿 부분과 웹 클라이언트 부분이 제외되어 있다. MMS에서 서버는 VMD와 매핑되어 있는 RMD를 가리키기 때문에 서버 시스템 내에 오퍼레이터를 위한 인터페이스를 제공하지 않았다. 단지 추후 사용자가 필요할 경우 확장할 수 있도록 하였다. 즉, 자바 어플리케이션과 메시지 교환을 할 수 있는 자바 애플릿을 추가한다면 서버 시스템에서도 사용자의 요구에 맞는 GUI를 구현할 수 있다. 응용 프로그램의 서버 구조는 (그림 11)에서 보여주고 있다. MMS 클라이언트로부터의 서비스 요청은 버클리 소켓에 의해 연결된 채널을 통해 받는다. 'MMS-C lib.'에서는 디코딩과 구조체 변환이 이루어지며 서버의 VMD로부터 RMD의 상태 정보와 같은 요청된 서비스의 결과를 MMS-Response PDU를 통해 클라이언트에게 전송한다.

5. 자동화된 PCB조립라인 모델

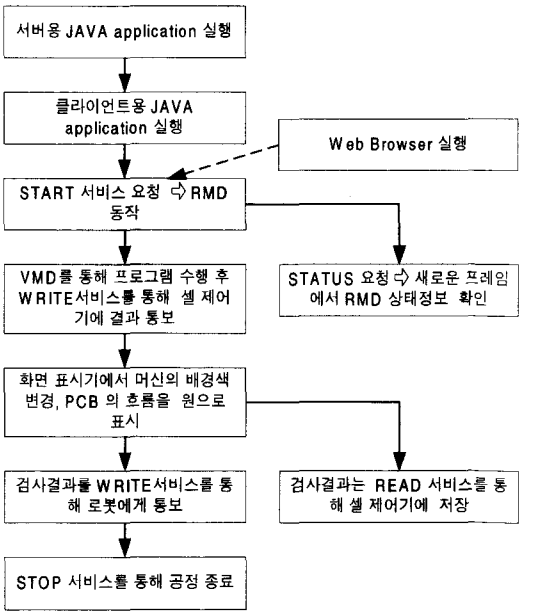
(그림 12)는 JNI를 이용해 구현한 MMS를 검증하기 위해 설정된 모델이다. 자동화라인은 그 시스템 설계자에 따라 다를 수 있다.



(그림 12) 자동화된 PCB조립라인 모델

응용 프로그램은 유닉스환경에서 구현되었다. (그림 12)는 전체의 구성을 나타낸 것이고 서버는 RMD와의 매핑을 위한 VMD까지만 구현하였다. 클라이언트는 셀 제어기이며 서버는 네 개의 프로세스로 구현하여 실행시켰다. 셀 제어기에 접근하는 웹 브라우저는 Windows 95 환경에서 실행시켰다. 서버 1은 로더(loader), 서버 2는 칩 마운터(chip mounter), 서버 3은 검사장비(inspection machine)이고 서버 4는 검사가 완료된 PCB를 언로더(unloader)에 분리하여 이동시켜 주는 로봇이다. 언로더는 본 데모라인의 구현에서는 고려되지 않았다. 4개의 서버 프로그램을 각각 실행시킨 후 클라이언트 프로그램을 실행시킴으로써 셀 제어기를 통한 웹 브라우저에서의 서버들을 제어하기 위한 모든 준비는 끝난다.

구현된 응용 프로그램에서는 이미 네이티브 언어로 작성된 자동화용 프로토콜인 MMS에 JNI를 이용해 자바 클래스화한 라이브러리를 이용했다. 그리고 인터넷 상의 원격지에서 조작자는 웹브라우저를 실행시켜 버튼 등을 이용해 파라미터를 전달하거나 실시간으로 데이터를 전달받아 자동화 기기들의 현재 동작 상황을 보여주도록 했다. 만약 자바를 사용하지 않았다면 조작자는 제어기가 위치한 장소에서만 자동화 기기 들을 제어·감시할 수 있었을 것이다. 자바를 사용함으로써 조작자의 위치에 상관없이 또, 조작자가 사용하는 플랫폼이 어떤 것이든 상관없이 실시간으로 감시와 제어가 가능한 것이다.



(그림 13) 데모라인의 동작순서

동작은 다음과 같은 순서로 하도록 구성하였다.

① 네트워크에 연결된 RMD는 모두 4개 - 로더, 부품 장착기, 검사장비, 로봇이며, 데모 라인을 위해 각각의 RMD를 위한 사용자 응용 프로그램인 자

바 어플리케이션 프로그램을 실행시킨다.

② 셀 제어기에서는 MMS 클라이언트를 위한 자바 어플리케이션을 실행시킨다. 서버와의 실질적인 통신을 이 응용이 담당하며 또한 자바 애플릿과 통신하여 웹 브라우저에서 제어, 감시할 수 있도록 하는 역할을 한다.

③ 셀 제어기에서 MMS의 START 서비스 요청을 통해서 각각의 RMD을 동작시킨다(이때 RMD의 전원은 ON된 것으로 가정한다.).

④ START 요청을 받은 머신은 VMD를 통해 프로그램을 수행시키고 그 결과를 다시 셀 제어기에게 보낸다. 정상적인 응답을 받은 셀 제어기는 머신이 동작을 시작했다는 표시로 그림으로 표시된 머신의 배경색을 바꾼다. 컨베이어의 PCB 흐름을 색이 칠해진 원으로 표시함으로써 현재 PCB의 진행 상황을 확인할 수 있게 했다.

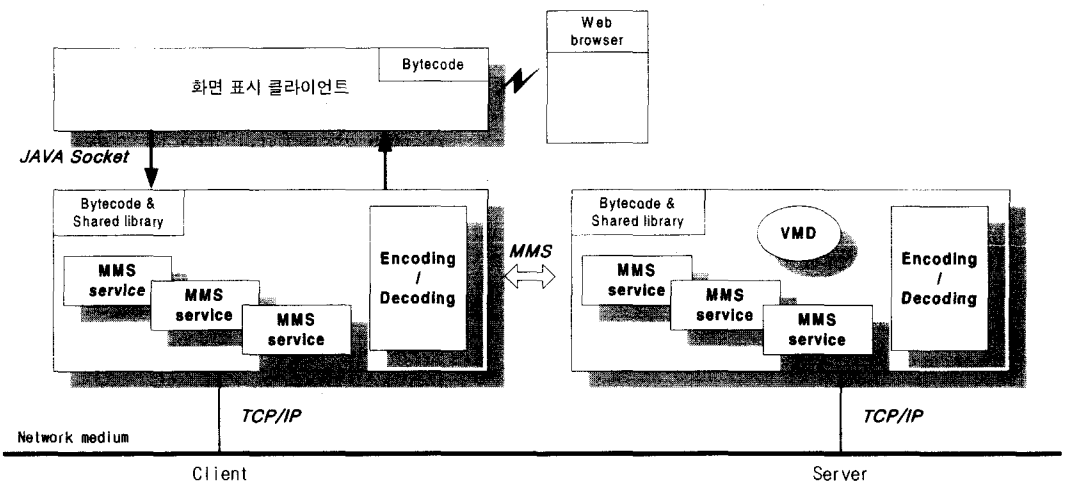
⑤ 동작하고 있는 RMD의 상태를 알고 싶으면 STATUS 서비스를 이용해 각각의 머신 상태를 체크할 수 있다. (RMD는 항상 정상 동작하는 것으로 가정했다.)

⑥ 로더로부터 나온 PCB는 부품 장착기에서 PCB를 조립하고 검사장비에서 양품 혹은 불량품인지를 검사한다. 검사장비는 WRITE 서비스를 이용해 로봇에게 결과를 알려준다. 셀 제어기는 이때 READ를 이용해 검사장비의 판정 결과를 읽어들인다.

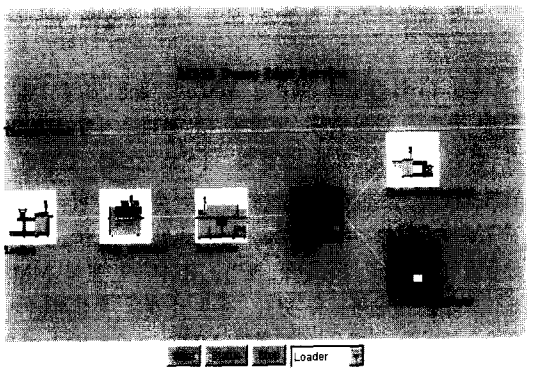
⑦ STOP 서비스를 이용해 RMD의 동작을 멈춤으로써 데모라인은 멈추고, 모든 조립 공정이 끝난다.

(그림 14)는 JNI를 이용한 MMS의 응용 프로그램의 전체 구조를 나타냈다. 초기에 웹 브라우저로 HTML과 화면 표시를 위한 애플릿 바이트코드를 로딩하기 위해 HTTP 프로토콜을 이용하도록 했다. 그리고 나서 애플릿과 MMS 클라이언트와의 통신은 자바 소켓을 사용했다. 실제로 RMD를 제어·감시하기 위해 MMS 프로토콜을 사용하도록 했다. 자바 소켓을 통해서는 사용자로부터의 제어 파라미터와 MMS 서버로부터 받은 결과를 MMS 클라이언트로부터 전송받아 화면에 표시하게 된다. 클라이언트와 서버는 각각 MMS 서비스 라이브러리를 가지고 있도록 했고, 서버에는 RMD와의 매핑을 위한 VMD를 두었다. MMS를 이용해 RMD를 제어·감시하기 위해 사용자는 웹 서비스를 이용하는 것처럼 웹 브라우저를 이용하기만 하면 되도록 했다. 사용자는 웹 브라우저를 이용함으로써 RMD를 동작시킬 수 있고, 감시할 수 있는 것이다.

(그림 15)는 PCB 조립라인 모델의 MMS 응용 프로그램의 실행 결과이다. 이것은 웹 브라우저에서 실행된 것을 갈무리한 그림으로 여기에 적용된 MMS 서비스는 STATUS, START, STOP, READ, WRITE 이다. 네이티브 언어로 구현된 다른 서비스들이 있지만 본 데모 라인을 위해 자바와 인터페이싱된 다른 MMS 서비스는 데모 라인을 위한 응용 프로그램에는 적용되지 않았다.



(그림 14) 데모라인의 전체 구조



(그림 15) PCB 조립라인 모델 실행 결과

각 RMD을 그림으로 표현함으로써 현재 작업이 진행되고 있는 상황을 한 눈에 알아볼 수 있게 했다. 현재 작업이 이루어지고 있는 RMD는 그림의 배경색을 변화시켰고, 컨베이어를 통해 이동되는 PCB를 채워진 원으로 표시하여 PCB의 흐름을 볼 수도 있다. RMD가 동작 중에도 상태 정보를 읽을 수 있도록 메뉴를 두어 STATUS 서비스 요청 버튼을 누름으로써 새로운 프레임(frame)을 띄우고 현재의 상태를 알려준다.

6. 결 론

지금까지 자동화용 표준 프로토콜인 MMS를 구현하고, JNI를 이용해 MMS를 자바와 인터페이싱하여 자바 언어로 MMS 사용자 응용 프로그램을 작성할 수 있게 하였다. 그리고 구현된 자바와 인터페이스된 MMS를 이용해 자동화된 PCB 조립라인을 모델로 응용 프로그램을 작성하여 구현된 JNI를 이용한 MMS가 사용자에게 일관성있는 인터페이스를 제공하는 웹 브라우저를 통해 인터넷상에서 RMD를 제어·감시할 수 있음을 보였다. 네 개의 서버와 하나의 클라이언트를 실행시켜 데모라인의 동작을 확인하였다.

대부분의 MMS가 네이티브 언어로 구현되어 라이브러리 형태로 제공된다. 제공되는 MMS는 각기 다른 API를 제공하고 있어 사용자 응용 프로그램을 작성하는데 어려움이 있으나, 본 구현에서는 MMS를 자바와 인터페이싱함으로써 간단한 HTML 언어로써 사용자 응용 프로그램을 작성하여 인터넷상의 어떤 곳에서도 MMS를 이용해 RMD를 제어·감시할 수 있는 기반을 구축했다.

참 고 문 헌

[1] 김정호, 이상범, "MAP 네트워크에서 MMS 운영을 위한 가상제조기기의 설계", 한국 정보처리학회 논문지 제2권 제3호, pp.397-405, May 1995.

[2] Peter A. Lagoni, Christophei, Crall, and Thomas G. Bartz, "HP MAP 3.0 Manufacturing Message Specification/800," Hewlett-Packard Journal, Aug. 1990.

[3] Robert W. Atherton, "Moving JAVA to the factory," IEEE SPECTRUM, Dec. 1998.

[4] Gray Cornell, Cay S. Horstmann, "Core JAVA," SunSoft Press, U.S,A, 1997.

[5] JavaSoft, "Java Native Interface Specification Release 1.1," Sun Microsystems Inc., 1997.

[6] 박정우, 김영신, 권욱현, "생산자동화용 국제표준 메시지통신규약-MMS", 제어·자동화·시스템공학회지, 제2권, 제4호, pp.30-41, Jul. 1996.

[7] 서울대학교 자동화시스템공동연구소, "생산정보통신망 기술개발에 관한 연구(최종보고서)", 통상산업부 과학기술처, Dec. 1996.

[8] SISCO, "MMS-EASE reference manual, revision 11," SISCO Inc., 1997.

[9] ISO/IEC 9506-1 Industrial automation systems - Manufacturing Message Specification, Part 1: Service Definition, 1990.

[10] ISO/IEC 9506-2 Industrial automation systems - Manufacturing Message Specification, Part 2: Protocol Specification, 1990.

[11] ISO 8824, Information proecessing systems-Open Systems Interconnection-Specification of Abstract Syntax Notation One (ASN.1), 1995.

[12] ISO 8825, Information processing systems-Open Systems Interconnection-Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), 1987.

[13] David Flanagan, "JAVA In a Nutshell, 2nd Ed.," O'Reilly, 1997.

[14] Jed Caben and John Jackman, "An Icon-Based Approach to System Control Development," IEEE Trans. on Industrial Electronics, Vol.37, No.3, pp.259-264, Jun. 1990.

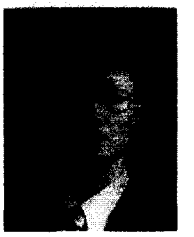
[15] Raymond Seng-Sim Cheah *et al*, "Design and implementation of an MMS environment on ISODE," Elsevier Science B.V. Computer Communications 20, pp.1354-1364, 1997.

[16] A. Valenzano, C. Dermartini and L. Ciminiera. "MAP and TOP Communications," Addison-Wesley, Massachusetts, 1992.

[17] E. Freund, H.-J. Buxbaum and U. van der Valk, "PC-based hierarchical manufacturing cell control," Control Eng. Practice, Vol.1, No.6, pp. 1047-1054, 1993.

[18] Sengoda G. Shanmugham, Terrence G. Beaumariage, Chell A. Roberts, "Manufacturing Communication : A Review of the MMS Approach," Computer ind. Engng Vol.28, No.1, pp.1-21, 1995.

[19] J. P. T. Mo and Y. Wang, "Integrated Robot Control using Manufacturing Message Specification Protocol based on NetBIOS," Control Eng. Practice, Vol.1, No.6, pp.971-978, 1993.



장 경 수

e-mail : kschang@ece.skku.ac.kr

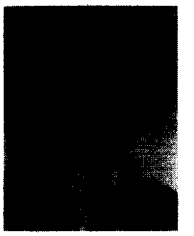
1994년 성균관대학교 전기공학과 졸업(학사)

1998년 성균관대학교 대학원(공학석사)

1999년~현재 성균관대학교 대학원 전기전자 및 컴퓨터공학부 박사과정

1994년~1995년 LG산전

관심분야 : Industrial Networks, ATM, Discrete Event Systems 등



신 동 렬

e-mail : drshin@ece.skku.ac.kr

1980년 성균관대 졸업(학사)

1982년 한국과학기술원(공학석사)

1992년 Georgia Institute of Tech. (Ph.D)

1994년~현재 성균관대학교 전기전자 및 컴퓨터공학부 부교수

1982년~1986년 Assistant researcher, Daewoo Heavy Ind., Limited

1992년~1994년 Senior researcher, Samsung Data Systems

관심분야 : Industrial networks including MMS, ICCP, and Fieldbus. High-speed telecommunication networks with emphasis on ATM modelling and switching. Performance evaluation with markovian and queueing networks 등