http://dx.doi.org/10.3745/KIPSTD.2011.18D.5.393

상황인식 미들웨어의 설계와 적용에 관한 연구

장 동 욱[†] · 손 석 원[††] · 한 광 록[†††]

요 약

본 논문에서는 상황인식 시스템 개발에 필수적인 미들웨어의 설계와 적용에 관하여 논한다. 다양한 상황 데이터들을 처리할 수 있도록 트랜스듀서 인터페이스 프로토콜을 정의하고, 미들웨어 모듈 간 일관성 있는 데이터 처리를 위하여 메시지 기반의 미들웨어를 설계하고 구현하였다. 상황인식 미들웨어는 서비스 기반 구조로 설계되어서 모듈간 기능이 독립적이고 확장성이 뛰어나다. 또한 모듈 간 통신은 메시지를 이용하기 때문에 응용 프로그램 개발이 용이해진다. 본 논문의 상황인식 미들웨어는 교량의 구조물 건전성 감시 시스템에 적용하여 실험하였고, 미들웨어의 효용성을 확인하였다.

키워드 : 상황인식 미들웨어, 구조물 건전성 감시, 트랜스듀서 인터페이스 프로토콜, 트랜스듀서 인터페이스 에이전트

A Study on Context Aware Middleware Design and Application

Dong-Wook Jang[†] · Surgwon Sohn[††] · Kwang-Rok Han[†††]

ABSTRACT

This paper describes a design and application of middleware that is essential to the context-aware system. We define a transducer interface protocol in order to deal with a variety of context data. For the purpose of systematic process of data between middleware modules, a message oriented middleware is designed and implemented. Memory improves the performance of high-performance computing system compared to previous strategies. Context aware middleware adopts service oriented architecture so that functions in modules may be independent and scalability can be remarkable. Using messages across modules decreases the complexity of the application development. In order to justify the usefulness of the proposed context aware middleware, we carried out our experiments in bridge health monitoring system and verified the efficacy.

Keywords : Context Aware Middleware, Structural Health Monitoring, Transducer Interface Protocol, Transducer Interface Agent

1. 서 론

최근 급격한 정보통신 기술발전 추세에 따라 스마트폰과 스마트 TV 등과 같이 컴퓨팅 모듈이 내장된 정보기기들이 최신의 통신기술과 접목됨으로서 다양한 분야에서 우리 삶을 혁명적으로 바꿔 놓고 있다. 제록스 팔로 알토(Xerox Palo Alto) 연구소의 마크 와이저(Mark Weiser)가 1988년 정의한 유비쿼터스 컴퓨팅이 빠르게 현실화 되고 있다[1]. 사용자가 필요로 하는 서비스를 제공하기 위해서는 일상생활 곳곳에 편재된 센서 및 컴퓨터들이 수집한 각종 환경 정보를 효율적으로 상호 공유하여 주변 상황(Context)을 인식하고 적절한 판단을 내리며 행동을 취하게 하는 상황인식 기술이 필요하다[2]. 상황인식 기술은 사용자를 중심으로 하는 주변 환경과 사용자간 혹은 사용자와 정보 기기간의 상호 관계를 지능적이고 능동적으로 선택하고 지원해줌으로서 사용자로 하여금 정보 획득 및 실행을 보다 쉽게 할 수 있도록 한다. 이러한 상황인식 기술은 일상 환경 속에 편재된 언제 어디서나 이용 가능한 유비쿼터스 컴퓨팅 환경에 가장 중요한 핵심기술 중의 하나이다[3].

유비쿼터스 컴퓨팅 시스템을 구축하기 위해서는 유비쿼터스 센서 네트워크(Ubiquitous Sensor Network) 와 상황인식 미들웨어 그리고 상황인식 시스템 등에 대한 연구가 필요하다[4]. 특히 상황인식 시스템을 구축하기 위해서는 상황 데이터를 처리하는 미들웨어의 개발이 요구된다.

따라서 본 논문에서는 다양한 환경의 상황정보를 처리하는 상황인식 시스템 개발에 적용할 수 있는 상황인식 미들

※ 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2010-0024242).
† 준 회 원 : 호서대학교 컴퓨터공학과 박사과정
†† 정 회 원 : 호서대학교 공과대학 뉴미디어학과 부교수
††† 종신회원 : 호서대학교 컴퓨터공학과 교수(교신저자)
논문접수 : 2011년 6월 7일
수 정 일 : 1차 2011년 7월 15일
심사완료 : 2011년 7월 16일

웨어의 설계에 관하여 기술한다. 이를 위하여 트랜스듀서 인터페이스(TI, Transducer Interface) 에이전트와 통신하기 위한 트랜스듀서 인터페이스 프로토콜(TIP, Transducer Interface Protocol)의 정의와 미들웨어의 기능별 모듈 설계 그리고 모듈간 데이터 처리를 위한 미들웨어 메시지 설계 방법 등을 논한다. 본 논문의 상황인식 미들웨어는 교량의 구조물 건전성 감시 시스템에 적용하여 실험하였다.

논문의 1 장과 2 장에서는 서론과 관련연구를, 2 장에서는 상황인식 미들웨어의 설계를, 4 장에서는 미들웨어 메시지 설계와 동작을, 그리고 5 장과 6 장에서는 실험과 결론을 기술하였다.

2. 관련 연구

초기 상황인식 서비스 연구에서는 다양한 상황정보와 상황정보 센싱 기술을 조합하여 특정 플랫폼만을 위한 개별적인 프로토타입 형태의 응용을 주로 연구하였다. 하지만 기존의 상황인식 시스템은 특정 플랫폼에만 연관되어 있어, 확장을 위해서는 많은 사전지식을 필요로 한다. 그리고 공통된 기능들의 모듈화가 되어 있지 않아 재사용이 어려운 실정이다. 이러한 문제점으로 인해 지난 10 여 년간 상황인식 응용 연구는 실험실 수준의 프로토타입 개발에 머물러 있었다. 근래에는 이러한 문제점을 인식하여 상황인식 서비스 인프라와 관련된 연구가 활발히 진행되고 있다[5].

일리노이즈 공대의 Scarlet-Context-aware infrastructure는 이질적 플랫폼간 상황정보 교환에 대해서 연구하였다. Scalet에서는 상황인식 컴퓨팅 환경의 모든 측면을 다루고 있지는 않다. 단지, 응용에 상황정보를 어떻게 제공할 것인가에 대한 문제에만 역점을 두고 있다. 특히, 이질적인 플랫폼간에 상황정보 전송을 위해 연구를 하였다. Scalet은 SOAP over HTTP와 WSDL을 활용하여 플랫폼간의 호환성을 유지하였다[7].

TU 뮌휀의 ServiceGlobe는 고객 단말의 종류, 스크린 해상도 및 지원 색상수, 고객의 위치을 기반으로 하는 상황인식 웹 서비스 플랫폼(Context Model, Context transmission, Context processing)에 대한 연구로서 다양하고 이질적인 고객 단말의 능력과 고객의 위치 등과 같은 상황정보를 고려하여 더 나은 상황인식 웹 서비스 제공을 목적으로 하고 있다[8]. 고객의 상황정보는 SOAP over HTTP를 통해 서비스 플랫폼으로 전송된다. 서비스 플랫폼에서는 SOAP 헤더에 포함된 상황정보를 추출하고 처리하고, 고객의 상황에 적절한 서비스를 제공한다.

일리노이즈 대학교의 Gaia는 상황인식 서비스 구조에 관한 연구로서 상황인식 서비스 구조로 응용이 다양한 상황정보를 얻고 추론할 수 있게 해준다[9].

싱가포르 국립대학의 SOCAM(Service-oriented Context-awae Middleware)는 상황인식 모바일 서비스를 위한 미들웨어에 관한 연구로 상황인식 서비스 및 시스템 개발을 용이하게 하기 위해 SOCAM을 제안하고 있다. 또한 미들웨어 내에서 상황정보 모델링을 위해 OWL(Web Ontology Language)를 제안하고 있다. 서비스 위한 컴포넌트들을 구성한다[10]

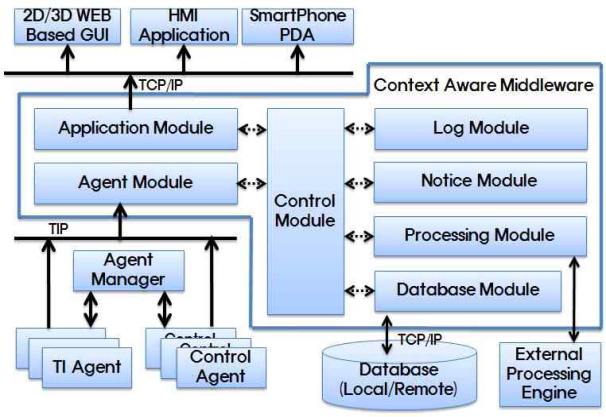
이와 같은 연구들은 서비스 인프라와 관련된 연구에서 상황인식 응용 개발에 필요한 공통 기능을 응용 레벨에서 분리하여 미들웨어 형태로 개발자에게 제공하는 방안을 연구하고 있다. 상황인식 서비스 인프라는 개발자에게는 응용과 관계된 기능에만 집중할 수 있도록 하고, 사용자에게는 일반화된 응용 서비스를 제공한다. 참고문헌 [6]에서는 상황인식 서비스 인프라의 요구 사항을 기능적인 요구 사항과 기능 외적인 요구 사항으로 분류하고 있다. 기능적 요구 사항(Functional Requirements)에는 상황정보 수집, 상황정보 저장 및 관리, 상황정보 구독 및 배달, 상황정보 분해 및 융합 등의 기능이 포함되며, 비기능적 요구 사항(Non Functional Requirements)에는 확장성, 모듈성, 보안성, 이동성, 인터페이스 적절성, 결함 포용성, 서비스 품질, 발전 가능성, 플랫폼간 호환성(cross-platform) 등이 있다[5].

이와 같은 연구동향에 따라 본 논문에서는 상황인식 시스템 개발에 필수적인 미들웨어의 개발을 목표로 하고 검토한 상황인식 시스템들의 특징을 적극적으로 반영하여 다양한 상황 데이터들을 처리할 수 있도록 트랜스듀서 인터페이스 프로토콜(TIP)을 정의하였다. 미들웨어는 컴포넌트 기반으로 모듈화하였으며, 모듈간 기능이 독립적이고 확장성이 뛰어나도록 설계하였다.

3. 미들웨어 설계

3.1 상황인식 시스템 구성

다양한 환경에서 다양한 상황 정보를 인식하여 처리하는 상황인식 시스템은 (그림 1)과 같이 구성된다. (그림 1)의 상황인식 시스템은 센서에서 상황 정보를 전송받아 이를 미들웨어로 전달하는 TI(Transducer Interface) 에이전트, 사용자로부터 또는 미들웨어로부터 명령을 받아 다양한 장치를 제어하는 장치제어 에이전트(Control Agent), 그리고 에이전트의 등록을 관리하는 에이전트 관리자(Agent



(그림 1) 상황 인식 시스템의 구성도

Manager), 상황정보를 모니터링하고 제어하는 사용자 프로그램(Application)과 미들웨어로 구성한다. 미들웨어는 크게 에이전트와 통신을 담당하는 에이전트 IO 계층(Agent IO Layer)과 사용자 프로그램과 통신을 담당하는 응용 계층(Application Layer), 상황정보를 처리하는 상황정보처리계층(Context Processing Layer) 그리고 UI 계층(User Interface Layer)로 구성된다.

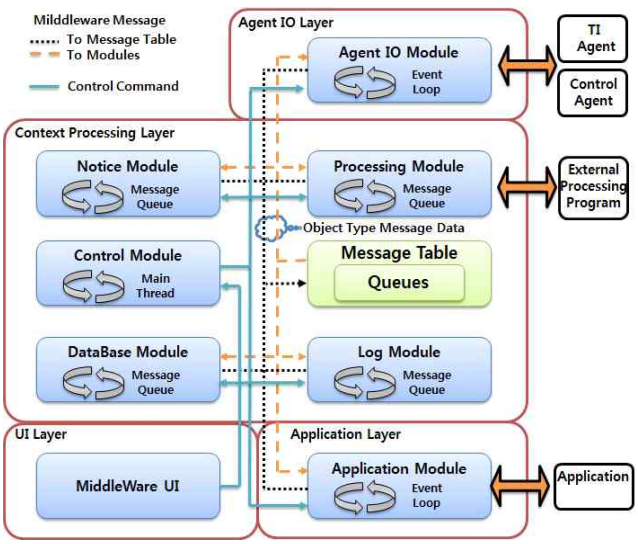
TI 에이전트는 다양한 센서나 장치에서 측정한 데이터를 TIP 프로토콜에 따라 패킷화하여 미들웨어로 전달하거나 미들웨어의 명령을 받아 센서나 장치로 전송하는 중계기 역할을 한다. 장치제어 에이전트는 장치를 제어하는 에이전트로서 미들웨어에서 전달받은 제어 명령어를 연결된 장치의 명령어로 변환하여 해당하는 장치에 전달한다. 에이전트 관리자는 TI 에이전트 또는 장치제어 에이전트의 종류를 식별하여 미들웨어와 통신을 할 수 있도록 연결해주는 역할을 한다. 사용자 프로그램은 미들웨어 메시지를 받아 사용자에게 표시해주고, 사용자의 명령어를 받아 미들웨어로 전달하는 역할을 한다.

3.2 미들웨어의 모듈 설계

TIP 프로토콜에 따라 TI 에이전트에서 패킷화된 센서들의 상황 데이터는 미들웨어로 전송된다. 미들웨어가 수신한 패킷 데이터는 미들웨어 메시지로 변환되어 각 모듈에서 처리되며, 웹 기반 UI와 응용프로그램, 스마트폰 등과 같은 사용자 프로그램을 이용하여 모니터링된다.

수신된 상황정보의 패킷을 분석하여 미들웨어 메시지로 변환하고, 장치를 제어하며 사용자와 인터페이스를 담당하는 미들웨어의 상세구조는 (그림 2)와 같이 7개의 모듈과 1개의 메시지 테이블, 그리고 사용자 UI로 구성된다.

앞 절에서 기술한 것과 같이 미들웨어는 크게 에이전트 IO 계층과 상황 처리 계층, 그리고 응용 계층으로 구분된다. 에이전트 IO 계층은 TI 에이전트와 통신하는 계층으로서 미



(그림 2) 상황 인식 미들웨어의 구조

〈표 1〉 미들웨어의 모듈

Module	Running Based	dependence
Control	Thread	Independence
UI	Event	Depend on Control
Notice	Message Queue	Independence
Awareness	Message Queue	Independence
Database	Message Queue	Independence
Log	Message Queue	Independence
Application	Event Loop	Independence
Agent IO	Event Loop	Independence
MessageTable	Queue Storage	Depend on Modules

들웨어가 수신한 데이터 패킷은 미들웨어의 에이전트 IO 모듈에서 미들웨어 메시지로 변환된 후 목적지 모듈을 표기하여 메시지 테이블의 해당하는 모듈큐에 저장된다.

응용 계층은 사용자 프로그램과 통신하는 계층으로 응용 모듈이 사용자 프로그램으로부터 데이터 요청을 받아 미들웨어 메시지를 외부 사용자의 요청에 맞게 가공한 후 전송하는 역할을 한다. 상황 처리 계층의 통지(Notice), 프로세스(Process), 데이터베이스(Database), 로그(Log) 모듈들은 메시지 테이블의 모듈큐에서 해당하는 미들웨어 메시지 읽어와 임무를 수행하는 구조이다. 각 모듈별 요약은 <표 1>과 같다.

각 모듈들의 역할을 간단히 기술하면 다음과 같다.

- 제어 모듈(Control Module) : 각 모듈의 이상여부와 메시지의 이상여부가 감지되면 정해진 규칙에 따라 조치한다.
- 미들웨어 UI : 각 모듈의 상황과 메시지 처리 현황을 사용자가 모니터링할 수 있도록 표시한다.
- 통지 모듈(Notice Module) : 이벤트가 발생할 경우 정의된 방법에 따라 사용자에게 통지한다.
- 프로세스 모듈(Processing Module) : 미들웨어 메시지를 분석하고 현재 상황을 판단하는 모듈이다. 필요에 따라 미들웨어가 외부 프로그램과 연동할 때 외부 프로그램과 통신을 담당한다.
- 데이터베이스 모듈(Database Module) : 모든 메시지의 내용을 외부 데이터베이스에 맞게 변환 후 저장하고, 요청에 따라 데이터를 읽어온다.
- 로그 모듈(Log Module) : 미들웨어의 모든 메시지의 흐름에 대한 기록과 함께 모듈별 메시지 처리 결과를 기록하는 모듈이다. 미들웨어의 신뢰성 확보를 위한 모듈이다.
- 응용 모듈(Application Module) : 사용자가 시스템의 상황을 모니터링 하기 위해서 모니터링 프로그램과 통신하는 모듈이다. 다양한 플랫폼에 대응하기 위해 미들웨어 메시지를 XML로 변환하여 외부 플랫폼에 전달하고, 외부 플랫폼에서 수신된 XML을 미들웨어 메시지로 변환한다.
- 에이전트 IO 모듈(Agent IO Module) : TI 에이전트 및 장치제어 에이전트와 통신하며 TI 에이전트에서 수신된 데이터 패킷을 미들웨어 메시지로 변환하는 역할을 한다.

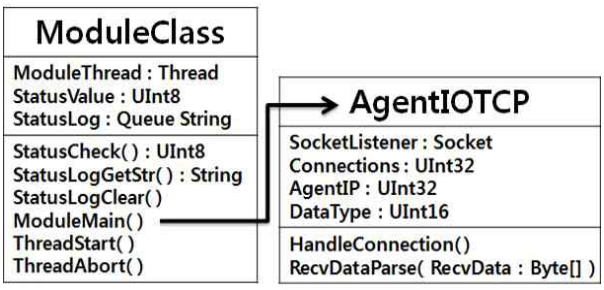
• 메시지 테이블(Message Table) : 미들웨어 메시지를 저장하기 위한 곳으로 각 모듈별 큐로 구성되어 있다. 미들웨어는 에이전트와 사용자 프로그램으로부터 데이터 패킷을 받아 객체 형태의 미들웨어 메시지로 변환 후 이 메시지를 목적지 정보에 따라 메시지 테이블의 해당하는 모듈큐에 저장한다.

<표 1>과 같이 UI와 메시지 테이블을 제외한 각 모듈은 DLL(dynamic linking library) 형태로 독립적으로 구성되어 있어 있기 때문에 기능의 추가 및 수정이 필요할 때는 해당 모듈만 수정하면 된다. 예를 들어, 데이터베이스를 Oracle에서 MS-SQL로 변경하고자 한다면 데이터베이스 모듈만 수정하고, 응급상황 감지 시 사용자 휴대폰의 SMS 통지에서 해당상황의 영상 등을 담아 통지하기 원한다면 통지 모듈에서 SMS를 MMS로 수정하면 다른 모듈의 수정 없이 즉시 적용할 수 있다.

3.3 미들웨어 모듈 제어

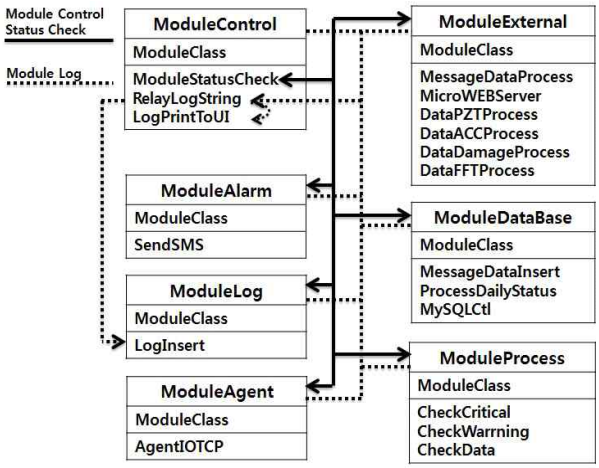
미들웨어의 각 모듈은 모듈클래스에 의해 운영된다. 모듈클래스는 모듈의 기본 기능을 수행하는 클래스이며, 모듈의 상태를 UI 및 다른 모듈에 전달하는 클래스이다.

(그림 3)은 미들웨어의 7개의 모듈 중 에이전트의 통신 요청을 받아 처리하는 에이전트 IO 모듈의 클래스 동작과정을 나타낸다. (그림 3)에서 모듈은 실행될 때 모듈클래스의 모듈메인(ModuleMain) 메서드를 실행함으로써 구동된다. 모듈클래스는 클래스의 상태값(StatusValue) 멤버와 처리 결과를 저장하고 있는 상태로그(StatusLog) 멤버가 있으며, 다른 모듈의 요청에 따라 상태값이나 결과를 전달하는 상태확인(StatusCheck) 메서드와 상태로그 전달(StatusLogGetStr) 메서드가 있다. 각 모듈은 기능에 따라 클래스의 모듈메인 메서드를 오버라이팅함으로써 각 모듈별 기능을 수행한다.



(그림 3) 에이전트 IO 모듈의 모듈 클래스

에이전트 IO 모듈은 모듈클래스의 모듈메인 메서드에서 소켓의 연결을 대기하며 에이전트의 통신 요청을 기다린다. 통신 요청을 받을 경우 에이전트와 통신 연결을 AgentIOTCP 클래스에 전달된다. AgentIOTCP 클래스는 핸들러(HandleConnection) 메서드에서 에이전트로부터 수신된 데이터 패킷을 처리한 후 수신 데이터 파서(RecvDataParse) 메서드에서 TIP 프로토콜에 따라 분석하여 미들웨어 메시지로 변환 후 메시지 테이블에 저장한다.



(그림 4) 모듈 클래스 구성과 제어 방법

이와 같이 미들웨어의 각 모듈들은 기본 클래스인 모듈클래스에 의해 동작하고 모듈의 기능에 따라 모듈메인 메서드를 오버라이팅하여 정의한다.

(그림 4)는 본 논문의 미들웨어의 모든 모듈들의 구성과 제어방법을 나타낸다.

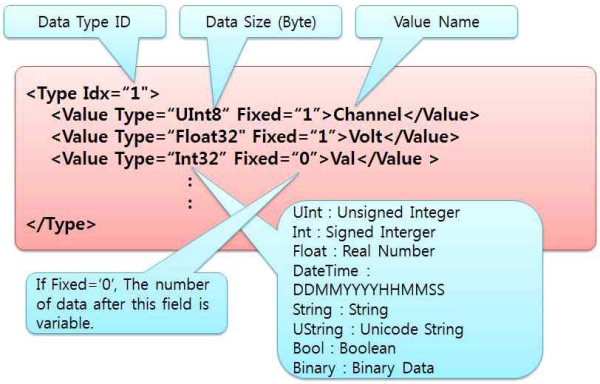
특히 (그림 4)의 제어모듈(ModuleControl)은 다른 6개 모듈의 상태를 확인하고 제어하는 역할을 한다. 제어모듈은 다른 6개의 모듈들의 상태를 확인하는 모듈상태확인(Module StatusCheck) 클래스, 각 모듈에서 생성된 로그를 로그 모듈로 전달하는 로그전달(RelayLogString) 클래스, 그리고 각 모듈의 로그를 미들웨어의 UI로 출력하는 로그출력 UI(LogPrintToUI) 클래스로 구성되어 있다. 따라서 제어모듈의 모듈메인 메서드가 모듈상태확인 클래스, 로그전달 클래스, 로그출력 UI 클래스를 지속적으로 확인하여 구동함으로써 다른 6개 모듈의 상태를 확인하고 로그모듈로 전달하는 기능을 수행한다.

(그림 4)의 각 모듈들은 기능에 따라 모듈클래스의 모듈메인 메서드에서 필요한 클래스만 추가하거나 수정함으로서 쉽게 구현이 가능하다.

4. 미들웨어 메시지 설계와 동작과정

4.1 TI 에이전트의 데이터 패킷화

(그림 5)는 센서들이 수집한 상황 데이터를 TI 에이전트가 TIP 프로토콜에 따라 패킷화하여 미들웨어로 전송하는 XML 데이터의 예를 나타낸다. (그림 5)의 예에서 데이터 형식은 1번으로 정의되어 있다. 1번 데이터 형식은 1바이트 크기의 부호 없는 정수 데이터 'Channel'과 4바이트 크기의 실수 데이터 'Volt', 4바이트 크기의 정수 데이터 'Val'로 구성된다. 특히 'Val' 데이터는 Fixed 값이 0이면 데이터가 한 개 이상 전달되는 가변 데이터이다. 즉, 4바이트 크기의 실수 데이터 'Volt' 뒤에 최소 한 개 이상의 4 바이트 크기의 정수 데이터가 이어진다. 이때 'Val'의 개수는 전체 패킷의 크기를 참조하여 계산한다.



(그림 5) 상황 데이터 패킷의 XML 구조

상황인식 시스템은 다양한 외부 센서들과 장치들이 연결되어 있기 때문에 TI 에이전트는 각 센서들의 형식에 따라 (그림 5)와 같은 XML 데이터를 포함하는 데이터 패킷을 생성하여 미들웨어로 전송한다.

4.2 TIP 프로토콜 설계

미들웨어와 에이전트 및 미들웨어와 응용프로그램간의 데이터 전송을 위한 TIP 프로토콜은 헤더와 데이터로 구성된다. 헤더는 데이터 형식에 관계없이 모든 데이터 앞에 붙어서 전송된다. 데이터 형식은 (그림 5)와 같이 XML 형식으로 센서의 특징에 따라 정의된다. <표 2>의 헤더에는 다음과 같은 정보가 기술된다.

〈표 2〉 TIP 프로토콜 헤더와 데이터

Protocol	Name	Size	Type
Header	Destination ID	2 Byte	Unsigned Int
	Group ID	1 Byte	Unsigned Int
	Source ID	1 Byte	Unsigned Int
	Data Type	2 Byte	Unsigned Int
	Date : Year	2 Byte	Unsigned Int
	Date : Month	1 Byte	Unsigned Int
	Date : Day	1 Byte	Unsigned Int
	Time : Hour	1 Byte	Unsigned Int
	Time : Minute	1 Byte	Unsigned Int
	Time : Second	1 Byte	Unsigned Int
	Time : 10 MilliSec	1 Byte	Unsigned Int
	Sequence No	2 Byte	Unsigned Int
	Length (Size)	4 Byte	Unsigned Int
Data	Data		Object

- Destination ID : 패킷의 최종 목적지 ID이다.
- Group/Source ID : 데이터를 송신하는 에이전트의 그룹과 그룹 내의 고유 ID이다. 에이전트는 그룹을 이용해 관리할 수 있다.
- Data Type : 헤더에 붙여 전송될 데이터의 구조를 표시하기 위해 사용한다. 각 데이터 구조마다 고유의 식별자를 가지고 있으며, 이를 이용하여 동시에 다양한 정보를 송수신할 수 있다.
- Sequence No : 에이전트에서 미들웨어로 전송하는 데이터의 전송 순서이다.
- Length : 헤더의 크기와 데이터의 크기를 포함한 실제 패킷의 전체 데이터 크기이다.

<표 2>의 헤더에서 데이터 형식은 XML로 정의된 데이터의 종류를 의미한다. 데이터 형식는 2 바이트의 부호 없는 정수 크기이므로 최대 65536 종류의 데이터를 처리할 수 있다.

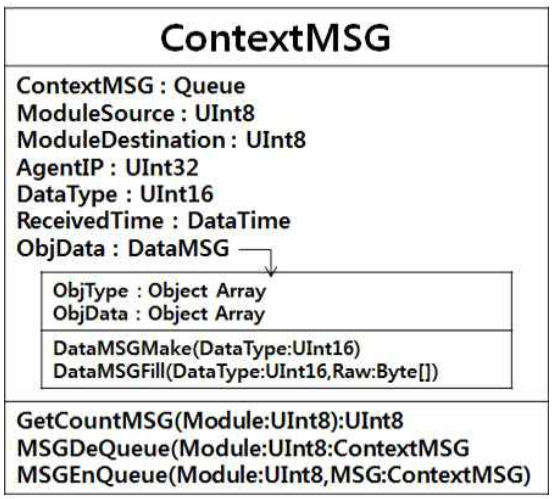
4.3 미들웨어 메시지 설계

에이전트 또는 응용프로그램은 센서들이 인식한 상황 데이터나 응용 프로그램에 필요한 명령들은 TIP 프로토콜에 따라 패킷화하여 미들웨어로 전송한다. 미들웨어의 에이전트 IO 모듈 또는 응용 모듈은 이 데이터를 수신한 후 미들웨어 메시지로 변환하여 메시지 테이블에 저장한다. 저장된 메시지는 각 모듈들이 주기적으로 메시지 테이블의 모듈큐를 검색하여 해당하는 모듈의 메시지를 읽어 온다.

이 미들웨어 메시지는 다양한 상황정보를 수용하기 위하여 (그림 6)과 같이 객체(Object)로 구성한다.

미들웨어 메시지 객체인 ContextMSG는 미들웨어 메시지의 기본정보와 데이터의 값을 저장하는 DataMSG 객체로 구성된다. ContextMSG의 기본정보는 미들웨어 메시지의 목적지 모듈(ModuleSource)과 출발지 모듈(ModuleDestination), 생성시간(DataType)과 함께 저장된 데이터 형식(DataType), 그리고 현재 모듈이 수신한 메시지의 개수를 돌려주는 GetCountMSG 메서드와 미들웨어 메시지를 메시지 테이블에 저장하고 읽어오는 MSGDeQueue 메서드와 MSGEn Queue 메서드로 구성된다.

ContextMSG 객체에 포함되어 있는 DataMSG 객체는 미들웨어의 에이전트 IO 모듈 또는 응용 모듈에서 수신된 다양한 형태의 데이터를 저장한다. ObjType은 헤더와 데이터



(그림 6) 미들웨어 메시지의 클래스

의 크기와 형식을 보관하는 가변 배열이고, ObjData는 헤더와 데이터의 실제 값을 보관하는 가변 배열이다. DataMSG 객체에서 ObjType과 ObjData를 비교하여 실제 데이터 형식과 값을 알아낸다. DataMSG의 DataMSGMake 메서드는 <표 2>의 TIP 프로토콜을 참조하여 수신된 패킷의 헤더와 데이터의 크기와 형식 정보만을 ObjType에 저장하고, DataMSGFill 메서드는 패킷의 실제 데이터를 추출하여 ObjData에 저장한다.

즉, 수신된 패킷을 분석하여 ContextMSG의 DataMSG 객체에 해당하는 데이터를 채워 넣음으로서 미들웨어 메시지를 생성한 후 메시지 테이블의 모듈큐에 저장한다. 각 모듈들은 저장된 ContextMsg의 Module Destination을 체크하여 해당하는 메시지를 읽어 온다.

4.4 미들웨어의 메시지 처리과정

(그림 7)은 TIP 프로토콜에 의해 생성된 상황 데이터의 패킷이 미들웨어 메시지로 변환되어 미들웨어에서 처리되는 과정을 나타낸다.

TI 에이전트가 외부 센서에서 수집된 상황정보를 TIP 프로토콜에 따라 패킷을 생성하여 미들웨어로 전송한다. 미들웨어의 에이전트 IO 모듈은 수신된 데이터 패킷을 미들웨어 메시지로 변환한다. 변환된 미들웨어 메시지는 메시지 테이블의 DB 모듈큐, 프로세스 모듈큐, 응용 모듈큐에 저장된다.

저장된 메시지는 각 모듈들이 주기적으로 메시지 테이블의 모듈큐를 검색하여 해당하는 모듈의 메시지를 읽어 온다.

또한 어플리케이션의 명령은 미들웨어의 어플리케이션 모듈에서 수신된 후, 미들웨어 메시지로 변환하여 동일한 방법으로 미들웨어의 DB 모듈 또는 에이전트 IO 모듈로 전송된다. 또한 미들웨어의 모든 모듈에서 처리된 결과는 미들웨어 메시지로 변환되어 로그 모듈에 전달되면, 로그모듈은 제어모듈에 메시를 전송한다. 제어모듈은 불필요한 메시지를 제거한 후에 필요한 메시지들을 UI에 전송하고, 오류 등과 같이 중요한 메시지는 다시 로그 모듈로 전송하여 DB에 저장하도록 한다(그림 4). 이때 전송되는 미들웨어 메시지들은 모든 과정에서 메시지 테이블의 모듈큐를 매개로 한다.

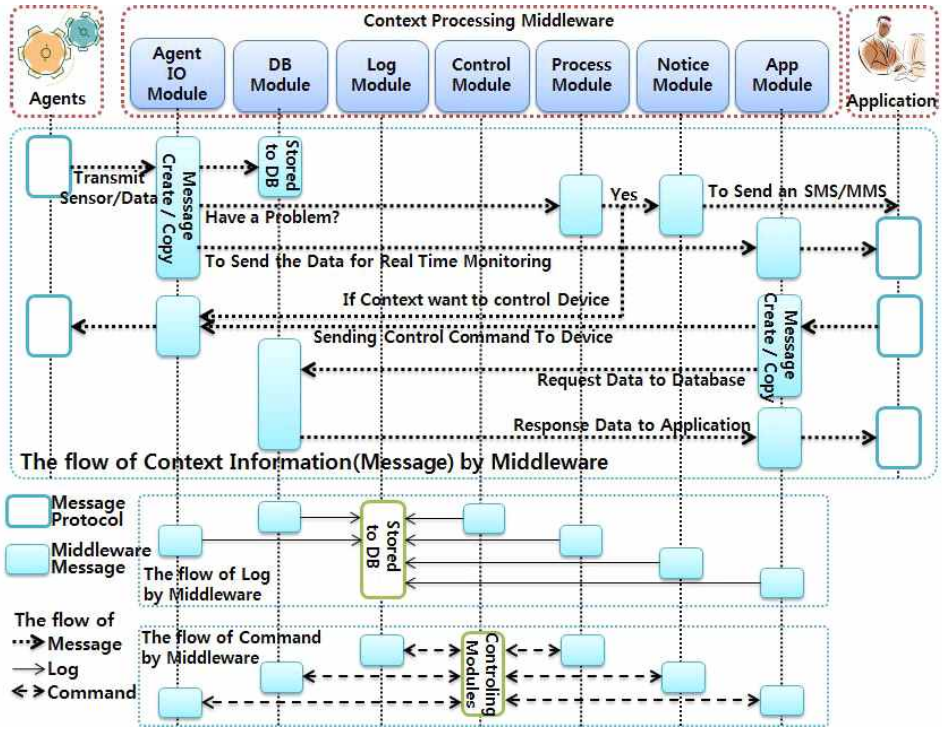
이와 같이 각 모듈은 미들웨어 메시지만을 이용하여 통신함으로써 모듈의 독립성을 유지할 수 있고, 모듈의 수정이 쉬우며, 미들웨어 메시지 또한 유기적인 설계가 가능하므로 어떠한 정보도 저장하고 전송할 수 있다.

5. 실 험

5.1 실험 환경

본 논문에서 제안한 상황인식 미들웨어는 구조물 건전성 감시 시스템에 적용하여 실험하였다. 실험을 위해 모형 교량을 설치하고 교량에는 Imote2의 압전(PZT) 센서와 3축 가속도(ACC) 센서를 장착하여 시스템을 구성하였다. 교량의 ACC 센서와 PZT 센서에서 측정된 교량의 상황 데이터를 TI 에이전트가 TIP 프로토콜에 따라 패킷화한 후에 미들웨어로 전송한다. 미들웨어는 수신된 패킷을 분석하여 미들웨어 메시지로 변환하고 프로세스 모듈을 통하여 구조물 건전성 감시 프로그램과 연동한다.

ACC 센서는 교량에서 가속도를 측정하여 상관 함수(Correlation Function)를 계산하고 이로부터 고유 진동수



(그림 7) 미들웨어 메시지 처리과정

(Natural Frequency)와 모드 형상(Mode Shape)을 구하여 구조물의 건전성을 판단하는데 사용된다. 즉, ACC 센서에 의해 구해진 모드형상은 구조물의 건전성 지표를 나타낸다.

또한 교량 구조물의 세 영역에 설치한 PZT 센서는 교량의 파손이나 볼트 체결 여부를 감시하여 손상 정도를 판단하는데 사용한다.

구조물의 건전성을 감시하기 위한 프로그램은 MATLAB을 이용하여 작성하였다. <표 3>은 본 실험에 사용된 센서를 나타내었다.

〈표 3〉 모형 교량 실험에서 사용된 센서

Sensor	Sampling Rate	Channel
ACC 센서	250Hz	3 Ch
PZT 센서	250Hz	3 Ch

교량에는 모두 6개의 센서가 사용되었으며, TI 에이전트는 ACC 센서와 PZT 센서로부터 각각 초당 250개의 데이터를 수집하여 미들웨어로 전송한다.

ACC와 PZT는 서로 데이터 형식이 다르기 때문에 4장의 4.1절에서 기술한 바와 같이 각각의 데이터형식을 <표 4>와 <표 5>와 같이 정의하였다.

<표 4>는 ACC 센서의 데이터를 수신하기 위하여 데이터 형식 1번을, <표 5>는 PZT의 데이터 형식 2번을 정의하였다.

<표 4>에 의하면 첫 번째 1바이트에는 4개의 센서를 구분하기 위한 채널, 두 번째부터 네 번째 데이터는 4바이트의 실수 데이터로 구성되어 있다.

<표 4> 가속도 센서를 위한 1번 데이터의 정의

```
<Type Idx="1">
    <Value Size="1" Type="UInt">Channel</Value>
    <Value Size="4" Type="Float">ValX </Value>
    <Value Size="4" Type="Float">ValY </Value>
    <Value Size="4" Type="Float">ValZ </Value>
</Type>
```

〈표 5〉 PZT 센서를 위한 2번 데이터의 정의

```
<Type Idx="2">
    <Value Size="1" Type="UInt">Channel</Value>
    <Value Size="1" Type="UInt">Cnt</Value>
    <Value Val="1" Size="4" Type="Float">Value</Value>
</Type>
```

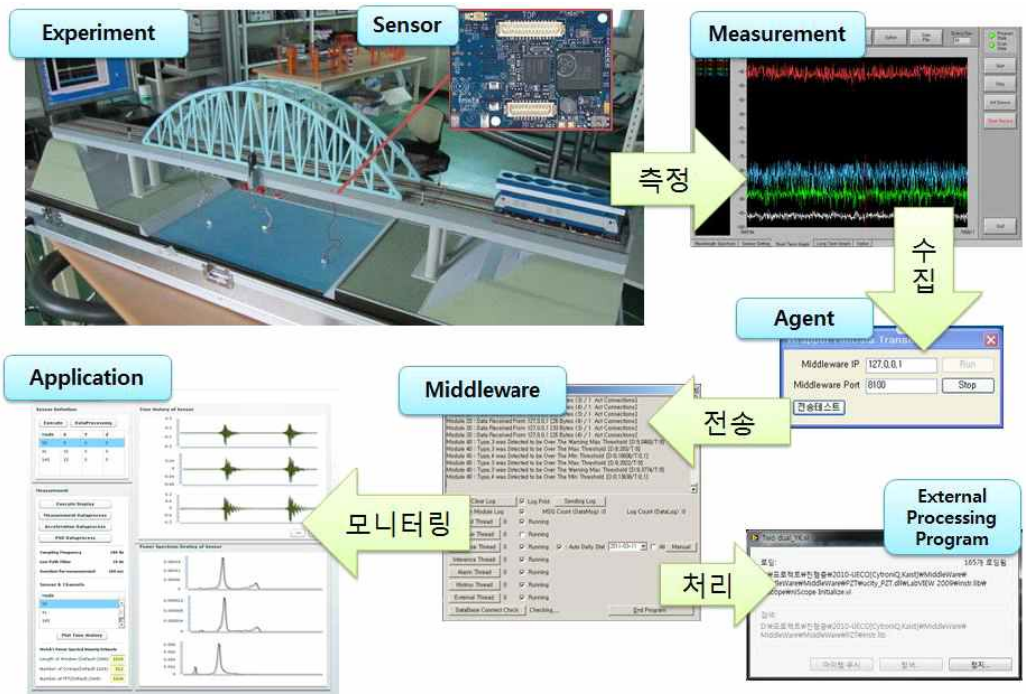
<표 5>에 의하면 첫 번째 1바이트에는 4개의 센서를 구분하기 위한 채널, 두 번째 1바이트에는 전송할 데이터의 개수를 저장한 카운트(Cnt), 다음으로 카운트 개수만큼의 데이터를 실수 4바이트 크기로 각각 이어서 전송한다.

5.2 구조물 건전성 감시 시스템 실험

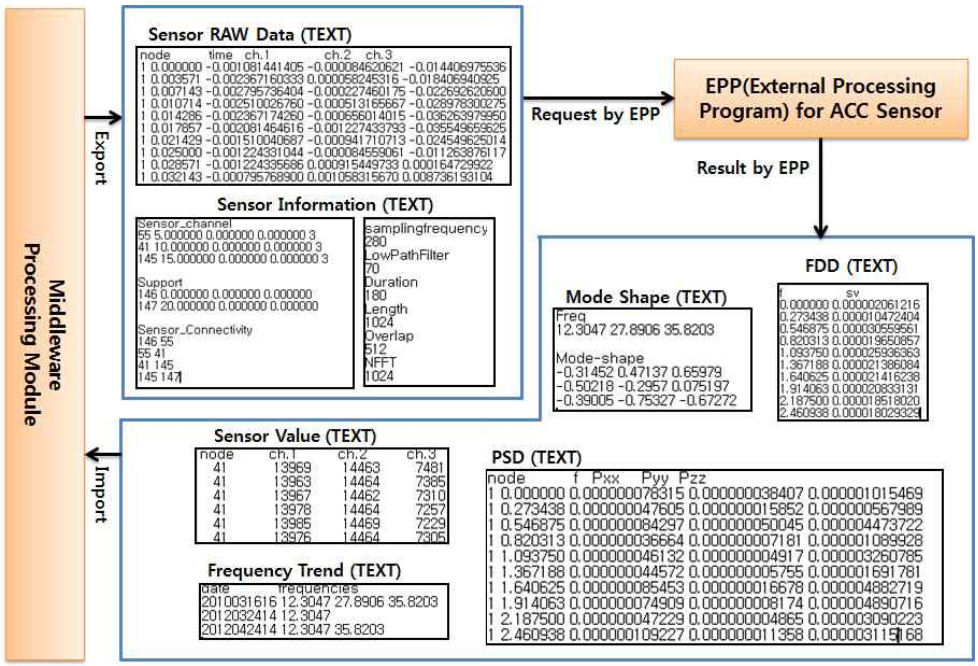
교량의 구조물 건전성 감시 시스템은 (그림 8)과 같은 방법으로 구축하여 실험하였다.

(그림 8)의 교량의 센서 데이터는 측정 후 에이전트를 거쳐 TIP 프로토콜에 따라 패킷 형태로 미들웨어에 전송된다.

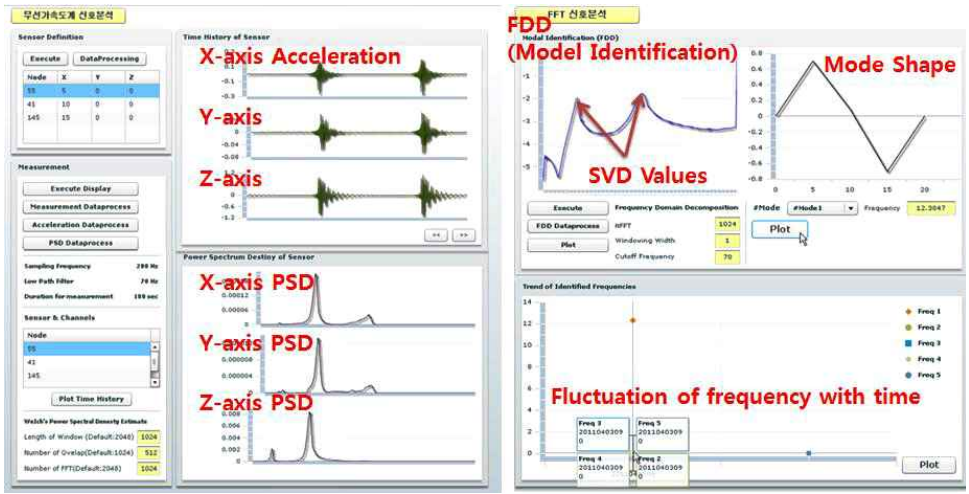
ACC 센서 데이터를 처리하기 위한 미들웨어와 외부 처리프로그램의 통신은 (그림 9)와 같다.



(그림 8) 교량 건전성 감시 시스템의 실험



(그림 9) 구조물 건전성 감시 프로그램 연동된 ACC 데이터 처리 과정



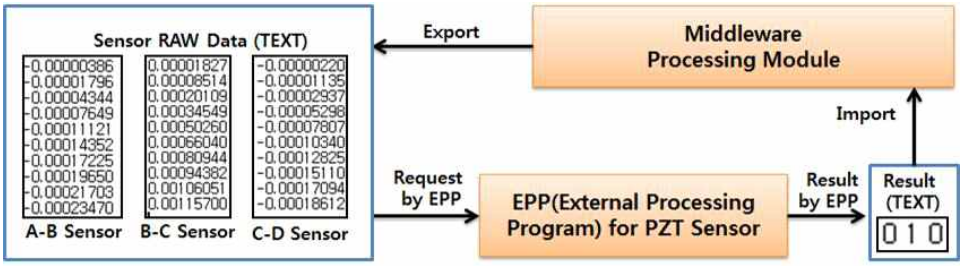
(그림 10) ACC 데이터 처리 결과의 웹 기반 UI

(그림 9)에서 미들웨어의 프로세스 모듈이 ACC센서 데이터를 처리하여 텍스트 파일로 외부 처리 프로그램에 전달하면 외부 처리 프로그램은 센서 값, FDD (Frequency Domain Decomposition), PSD(Power Spectrum Density), Frequency Trend, SVD(Singular Value Decomposition), 모드형상(Mode Shape) 값을 계산하여 그 결과를 프로세스 모듈에 다시 반환한다. 미들웨어는 그 결과 값을 4.4절에서 설명한 미들웨어의 메시지 처리과정을 거쳐 응용 모듈로 전달하여 모니터링 프로그램에서 결과를 출력하도록 한다.

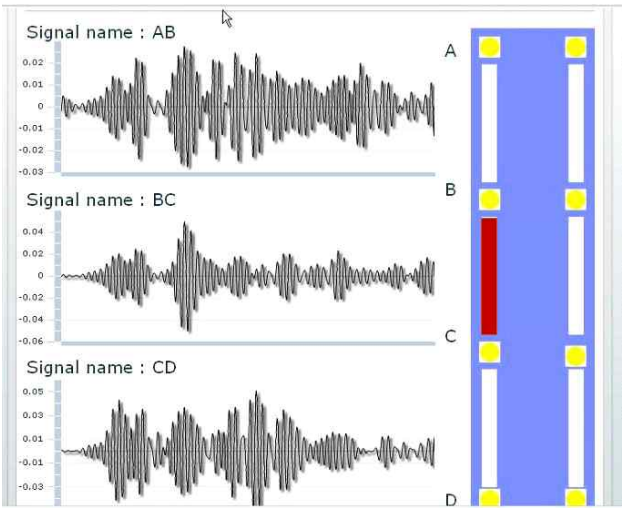
(그림 10)은 웹 기반의 사용자 모니터링 프로그램의 처리 결과를 나타낸다. (그림 10)의 실험결과에서 FDD의 SVD 값과 3차 모드형상이 명확하게 나타나기 때문에 ACC 센서에 의한 계측이 문제없이 실행된 것을 알 수 있다[11]. 또한 열차가 교량을 지나갈 때 나타나는 교량의 진동을 측정한 모드 형상 역시 유한요소모델을 해석해서 얻는 모드 형상을 참고하여 비교하면 교량의 상태가 정상임을 알 수 있다.

PZT 센서는 (그림 11)과 같은 과정으로 처리된다.

A-B영역과 B-C영역 그리고 C-D영역의 PZT 센서 값들이 미들웨어를 거쳐서 외부 프로그램에 전달되면 외부 프로그램은 문제가 있는 부분을 부울 값으로 처리하여 미들웨어로 다시 반환한다. 그 이후의 처리 결과는 ACC 센서와 동일한 과정에 의해서 (그림 12)와 같이 모니터링 프로그램에



(그림 11) 구조물 건전성 감시 프로그램 연동된 PZT 데이터 처리 과정



(그림 12) PZT 데이터 처리 결과의 웹 기반 UI

의해서 출력된다. 실험에서는 B-C영역에 교량의 손상을 가정하고 실험하기 위해 볼트를 느슨하게 풀어 놓은 상태이다 [12]. 따라서 측정된 데이터와 교량의 손상이 없는 상태에서 측정한 데이터를 비교하면 B-C 영역에서 임계치 이상으로 변화가 측정되어 문제가 있다는 것을 (그림 11)과 (그림 12)를 통해서 알 수 있다.

6. 결 론

본 논문에서는 상황인식 미들웨어를 설계하여 교량의 구조물 건전성을 감시하는 시스템에 적용하였다. 다양한 형태로 인식되는 센서 데이터는 TIP 프로토콜 정의에 의하여 TI 에이전트에서 패킷화하여 미들웨어로 전송하도록 하였다.

기능별로 객체화된 모듈들로 구성된 서비스 지향의 미들웨어(Service-oriented Middleware)를 설계하였으며 각 모듈간의 데이터 교환은 미들웨어 메시지를 설계하여 처리하였다. TI 에이전트와 TIP 프로토콜에 의하여 다양한 형태와 환경의 상황 데이터를 미들웨어에서 수용할 수 있고, 상황인식 시스템의 플랫폼이 변경될 경우에는 해당하는 미들웨어 모듈의 컴포넌트만을 변경함으로써 새로운 환경에 쉽게 적용할 수 있게 된다.

모형 교량에 ACC 센서와 PZT 센서를 설치하여 구조물의 건전성과 손상여부를 감시하는 외부 프로그램과 연동하는 실험을 통하여 미들웨어의 유용성을 테스트하였다.

본 논문에서 제안한 상황인식 미들웨어를 활용한다면 u-Bridge, u-Health와 같은 시스템 뿐 아니라, 공공안전관리 시스템, 상하수도 원격관리 시스템, 해양환경 감시 시스템 등과 같은 상황인식 시스템에서 감시하고자 하는 기능에 따라 미들웨어의 필요한 모듈만을 수정하거나 교체하면 비교적 쉽게 적용이 가능하기 때문에 개발비용과 시간을 절약할 수 있을 것으로 기대된다.

참 고 문 헌

[1] M.Wesier. “Some Computer Science Issues In Ubiquitous Computing.” Communication of the ACM, Vol.36(7), pp.75-84, 1993.

[2] G. Banavar, A. Bernstein, “Issue and challenges in ubiquitous computing : Software infrastructure and design challenges for ubiquitous computing applications,” Communication of ACM, Vol.45, No.12 , 2002.

[3] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. Mynatt, T. E. Starner and W. Newstetter, “The Aware Home : A Living Laboratory for Ubiquitous Computing Research,” Proc. of the 2nd Int'l. Workshop on Cooperative Buildings, LNCS 1670, pp.191-198, 1999.

[4] P. Couderc, A. M. Kermarrec, “Improving Level of Service for Mobile Users Using Context-Awareness,” 18th IEEE Symposium on Reliable Distributed Systems, pp.24-33, 1999.

[5] 김재호, 신경철, “상황인식 서비스 기술 연구 동향,” 주간기술동향 통권 1178호, 정보통신진흥원, 2004. 12.

[6] Bill Schilit, Norman Adams and Roy Want, Context - aware computing applications,” In proceedings of IEEE Workshop on Mobile Computing Systems and Applications, 1994.

[7] Abhay Daftari, Nehal Mehta, Shubhanan Bakre and Xian-He Sun, “On the Design Framework of Context Aware Embedded Systems,” Monterey Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation, 2003.

[8] The ServiceGlobe Project, http://www.db.fmi. uni-passau.de/projects/sg/

[9] Gregory Biegel and Vinny Cahill, “A Framework for Developing Mobile, Context-aware Applications,” IEEE International Conference on Pervasive Computing and Communications (PerCom), 2004.
[10] T. Gu, H.K. Pung and D.Q. Zhang, “A Middleware for Building Context-Aware Mobile Services,” In Proceedings of IEEE Vehicular Technology Conference(VTC), 2004.
[11] Rune Brincker, Lingmi Zhang, Palle Andersen, “Model identification of output-only system using frequency domain decomposition”, Smart Materials and Structures Journal Vol.10, pp.441-445, 2001.
[12] 서혜원, 박민석, 이수헌, 신경재 “PZT 센서를 이용한 철골보 손상계측”, 한국강구조학회 논문집, 제22권 제5호, pp.477-485, 2010. 10.

장 동 욱
e-mail : coco@hcilab.net
2005년 호서대학교 컴퓨터공학과(학사)
2007년 호서대학교 컴퓨터공학과 (공학석사)
2007년～현 재 호서대학교 박사과정
관심분야 : HCI, e-Health, 상황인식 시스템, 무선센서네트워크 등

손 석 원
e-mail : sohn@hoseo.edu
1985년 인하대학교 전자공학과(공학사)
1987년 인하대학교 전자공학과 정보공학 전공(공학석사)
2007년 인하대학교 컴퓨터정보공학과 (공학박사)
1999년～현 재 호서대학교 공과대학 뉴미디어학과 부교수
관심분야 : 무선센서네트워크, 이동통신, 제약만족최적화 등

한 광 록
e-mail : krhan@hoseo.edu
1984년 인하대학교 전자공학과(공학사)
1966년 인하대학교 전자공학과(공학석사)
1989년 인하대학교 전자공학과(공학박사)
1991년～현 재 호서대학교 공과대학 컴퓨터공학과 교수
2001년～2002년 ISI/University of Southern California 초빙 연구원
2009년～2010년 Southern Oregon University 객원교수
관심분야 : 정보검색, HCI, e-Health, 시멘틱웹 무선센서네트워크, 상황인식 시스템 등