유비쿼터스를 위한 범용 미들웨어 응용 시스템에 관한 연구

장동욱*, 임재산*, 홍정보*, 임종훈*, 한광록*
*호서대학교 컴퓨터공학과
e-mail:{coco,ronz,clazed}@hcilab.net, wwa203@naver.com, krhan@hoseo.edu

A Study on General-Purpose MiddleWare for Ubiquitous

Dong-Wook Jang*, Jae-San Lim*, Jung-Bo Hong*, Jong-Hoon Lim*,
Kwang-Rok Han*
*Dept of Computer Engineering, Hose University

요 약

최근 급격한 스마트 기기와 통신기술의 발전은 다양한 분야에서 센세이션을 일으키며 우리 삶을 혁명적으로 바꿔 놓고 있다. 이러한 변화에는 주변 상황정보를 인식하여 처리하는 상황정보 처리 시스템의 발전이 큰 역할을 하고 있다. 그러나 상황정보를 받아 처리하는 미들웨어 시스템의 범용화, 규격화가 이뤄지지 않아 미들웨어의 재사용이 어려워 중복개발 등으로 유비쿼터스 저변확대에 저해가 되고 있다. 이에 본 논문에서는 상황정보 데이터를 규격화 하고 독립적이고 모듈화된 메시지 기반의 미들웨어를 설계하고, 외부와의 인터페이스에 XML을 이용한 범용 미들웨어 응용 시스템을 제안한다.

1. 서론

최근 급격한 기술 발전추세에 따라 Thin-Client, PDA의 발전과 더불어 몇 년 사이 스마트폰과 스마트 TV 등과 같이 컴퓨팅 모듈이 내장된 정보기기(Post PC)들이 최신의 통신기술(Bluetooth, Wireless, ThunderBolt, LTE, WIBRO, WIMAX, 4G)과 접목됨으로서 다양한 분야에서 센세이션을 일으키며 우리 삶을 혁명적으로 바꿔 놓고 있다. 제록스 팔로 알토(Xerox Palo Alto) 연구소의 마크 와이저(Mark Weiser)가 1988년 정의한 유비쿼터스 컴퓨팅[1]이 빠르게 현실화 되고 있다. 사용자가 필요로 하는 서비스를 제공하기 위해서는, 일상생활 곳곳에 편재된 센서 및 컴퓨터들이 수집한 각종 환경 정보를 효율적으로 상호 공유하여 주변 상황을 알아내는 상황 처리 기술이 필요하다[2]. 상황 처리 기술은 사용자를 중심으로 하는 주변 환경과 사용자간 혹은 사용자와 정보 기기간의 상호 관계를 지능적, 능동적으로 선택하여 지원해 줌으로써, 사용자로 하여금 정보 획득 및 실행을 보다 용이하도록 한다. 이러한 기술은 “일상 환경 속에 편재된 언제 어디서나 이용 가능한 컴퓨팅 환경”인 유비쿼터스 컴퓨팅 환경에 가장 중요한 핵심 기술 중에 하나이다[3].

유비쿼터스 컴퓨팅에서 상황인식 처리 응용시스템을 구현하기 위하여 데이터를 처리하는 미들웨어가 필요하며, 국내외 많은 미들웨어가 중복 개발되고 있어 유비쿼터스 저변확대에 저해가 되고 있다.

본 논문에서는 유비쿼터스 컴퓨팅에서 상황인식 처리 응용시스템 구현을 위한 모듈화 된 범용 미들웨어를 제안하고, 이를 응용시스템에 구현한다.

2. 관련연구

유비쿼터스 컴퓨팅에서의 상황인식 처리를 위한 미들웨어 시스템은 향후 시스템의 확장성 및 독립성을 고려하여 개념적으로 설계되어야 한다.

현재 국내외 많은 연구기관에서 미들웨어가 개발되어지고 있다. 국외의 대표적인 미들웨어 시스템은 다음과 같다. 첫째 IRISA/INRIA의 연구[4], 둘째 AT&T의 연구[5], 셋째 Lancaster 대학의 GUIDE 프로젝트[6]등이 있다. 그 중 IRISA/INRIA의 연구는 이동성을 가진 사용자에 대해서 사용자의 주변 환경을 감지하여 그에 적합한 최적의 서비스를 제공하기 위하여 상황 객체(Contextual Object:CO)를 정의하고 있다[4]. 전체적인 구조로는 서버 측과 클라이언트 측으로 구분된다. 서버 측에서는 CO를 보관하고 검색할 수 있는 역할을 담당하며, Co의 Variant들을 저장하고 있으며, 클라이언트 요청시 이에 따라 응답한다. 클라이언트 측은 크게 응용 계층(Application Layer), 적응 계층(Adaptice Layer), 탐지/전송 계층(Detection/Notification Layer)으로 구분되어 있다. 응용계층은 최상위 계층으로서 시스템에서의 상황에 따른 처리 결과를 열람하거나 다른 계층에 전달하는 역할을 담당한다. 적응 계층은 COs 관리기, 상황 관리기, 선택 관리기로 구성되어 있으며, 이중 COs 관리기는 응용계층에서 사용되는 CO의 정보를 관리하고, 상황 관리기는 탐지/전송 계층에서 상황의 변화가 감지되었을 때 그에 대한 정보를

관리하는 역할을 담당한다. 선택 관리기는 사용자의 취향이나 선호도를 기반으로 상황에 대한 최적화된 결과를 얻을 수 있도록 필터링 하는 역할을 담당한다. 마지막으로 탐지계층은 시스템, 네트워크의 상황의 변화, 주변의 환경을 감지하는 역할을 담당한다.

AT&T의 연구는 실내에서 사용자의 위치를 탐지 및 추적하여 사용자의 주변 환경에 맞는 상황인식 응용 서비스를 제공하고 있다[5]. AT&T의 연구의 특징은 미들웨어가 없이 오라클 데이터베이스의 튜플 등을 이용하여 응용시스템을 구현 하였다.

Lancaster 대학의 GUIDE 프로젝트는 도시를 방문한 여행객의 편의와 효율적인 여행 서비스를 제공하기 위해 개발된 응용시스템이다[6]. 즉 도시를 방문한 여행객에 PDA와 같은 단말기를 제공하고, 주변의 관광 명소 등을 HTML 기반으로 안내하도록 설계되었다. 이 시스템 역시 미들웨어를 자체개발하여 다른 시스템에 적용하기가 어렵다는 문제가 있다.

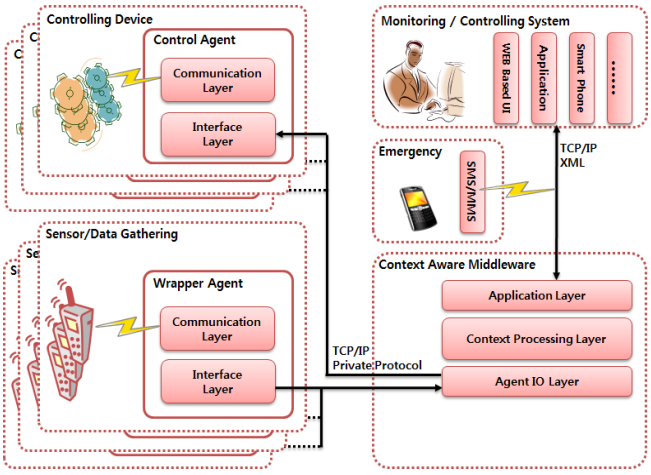
국내의 경우 광주과학기술원의 연구[7], 순천향대학교의 연구[8], 그리고 부산대학교의 연구[9]가 있다.

광주과학기술원의 연구에서는 개인화된 서비스를 구축하기 위해 컨텍스트 인식 프레임워크인 ubi-UCAM 2.0을 제안하였다[7]. ubi-UCAM은 5W1H(Who, What, Where, When, How, Why)의 각 항목별 세부요소를 지정함으로써 여러 서비스에서 동시에 사용할 수 있는 구조를 제공한다. 순천향대학교의 연구의 특징은 상황인식 정보를 블루투스를 이용하여 획득하고, 이 정보를 XML로 표현하여 서비스를 제공한다는 점에 있다[8]. 부산대학교의 연구에서는 JINI 미들웨어 기반의 상황인식 채팅 프로그램을 구현하였다[9]. JINI 미들웨어의 특징은 자바의 특징을 잘 살려 구현하였으며, 프로토콜을 API 형태로 제공하고, 컨텍스트 정보는 XML로 표현하고 있다.

본 논문에서는 위의 미들웨어 시스템들의 장점들을 적극 결합하여 상황인식에 기반한 다양한 응용 서비스에 쉽고 빠르게 적용할 수 있는 범용 미들웨어 시스템을 제안한다.

3. 범용 미들웨어 응용 시스템

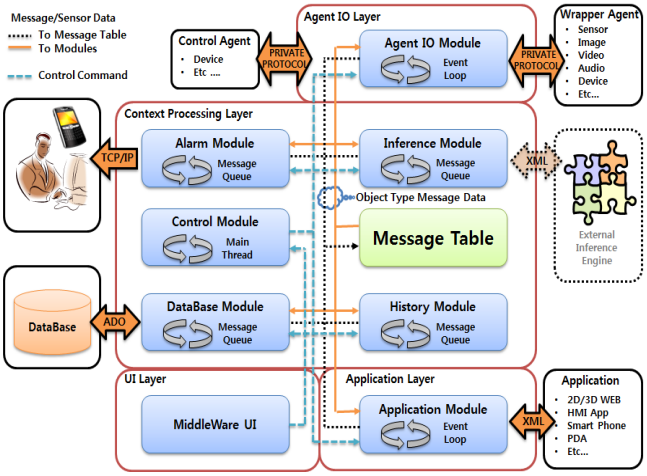
범용 상황인식 처리를 위한 미들웨어 응용 시스템은 그림 1과 같이 Wrapper Agent, Control Agent, Middleware, Monitoring/Controlling System으로 구성된다. Wrapper Agent는 다양한 센서 등에서 얻어진 상황정보를 미들웨어에서 요구하는 데이터 형식에 맞게 변환하여 미들웨어로 전달하는 역할을 담당한다. 미들웨어는 Control Agent와 Wrapper Agent와의 효율적이고 통일된 정보 교환을 위해 새롭게 설계된 프로토콜을 사용하고 있다. Wrapper Agent를 통해 수집된 상황정보는 미들웨어의 Agent IO Layer를 통해 미들웨어의 메시지 데이터로 저장되어 Context Processing Layer에서 처리된다. 사용자는 다양한 인터페이스(WEB Based UI, 응용프로그램, 스마트폰등)를 이용하여 미들웨어의 상황정보를 모니터링 한다.



(그림 1) 범용 미들웨어를 이용한 응용시스템의 구조

3.1 범용 미들웨어의 설계

제안하는 범용 미들웨어는 전체적으로 그림 2와 같이 7개의 모듈과 1개의 메시지 테이블, 그리고 UI로 구성되어 있다.



(그림 2) 범용 미들웨어의 구조

Agent IO 모듈과 Application 모듈에서 전달되는 모든 데이터는 메시지 형태로 변환된 후, 목적지의 모듈을 메시지에 표기한 후, 메시지 테이블에 저장된다. 이후 Alarm, Inference, DataBase, History 모듈은 자신이 목적지로 표기된 메시지를 메시지 테이블에서 가져와 본연의 임무를 수행하는 구조이다.

각각의 모듈별 역할을 간단히 기술 하면 다음과 같다.

- Agent IO Module : 외부의 Wrapper Agent와 Control Agent와의 데이터 송수신 역할과 함께, Agent에서 수신된 데이터와 미들웨어의 메시지 형태의 데이터를 상호 변환하는 역할을 담당한다. 또한 외부 Agents들의 상태 여부를 정기적으로 확인하여 외부 Agents의 상태를 전달하는 역할을 담당한다.
- History Module : 미들웨어의 모든 메시지의 흐름에

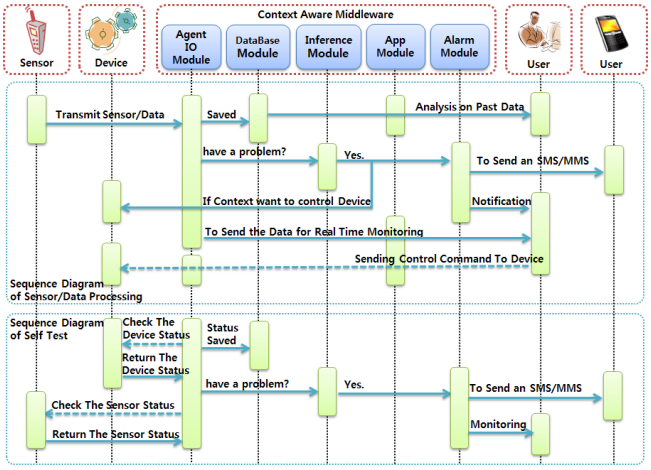
대한 기록과 함께 각각의 모듈별 메시지 처리 결과를 기록하는 모듈이다. 미들웨어의 신뢰성 확보를 위한 모듈이다.

- DataBase Module : 미들웨어는 상황정보를 처리하도록 설계되어 있으며, 상황정보 처리의 효율성과 범용성을 갖기 위해 스스로 저장하지 않는다. 따라서 상황정보 보관을 위하여 외부 데이터베이스 시스템과의 연동이 필요하며, DataBase Module이 이 역할을 담당한다. 이 모듈은 모든 메시지의 내용을 외부 데이터베이스(Oracle, MSSQL, MySQL등)에 맞게 저장하고, 요청에 따라 데이터를 읽어오는 역학을 담당한다.
- Inference Module : 상황정보가 저장된 메시지를 분석하고 현재 상황을 판단하는 모듈이다. 필요에 따라 외부 상황인식 엔진을 이용할 수 있으며, 이때는 상황인식 엔진과 통신을 담당하는 역할을 한다.
- Alarm Module : 응급상황이 인지되었을 경우 이 모듈에 정의된 방법에 따라 사용자에게 통지하고, 사용자에 의해 정해진 룰에 의해 상황에 대처하는 모듈이다.
- Application Module : 사용자가 상황에 대해 모니터링하기 위해 경우에 따라 다양한 플랫폼의 모니터링 SW가 요구되며, 이와 통신하는 모듈이다. 다양한 플랫폼에 대응하기 위해 미들웨어 내부의 메시지를 XML로 변환하여 외부 플랫폼에 전달하고, 외부 플랫폼에서 수신된 XML을 내부 메시지에 맞게 변환하는 역할을 담당한다.
- Control Module : 미들웨어의 각 모듈의 이상여부와 메시지의 이상여부를 감지하여, 이상여부 발생 시 정해진 룰에 의해 조치함으로서 미들웨어의 전체적인 운영을 담당하는 모듈이다.
- Middle UI : 미들웨어의 각 모듈별 현황과 메시지 처리 현황을 사용자에게 보여주는 역할을 담당한다.

각각의 모듈은 DLL 형태로 완벽하게 독립적으로 실행되며, 각각의 모듈별 통신은 메시지 테이블을 이용한 메시지 전달로 이루어진다. 예를 들어 데이터베이스 시스템을 Oracle에서 MSSQL로 변경하고자 한다면 DataBase Module만 수정하고, 응급상황 감지 시 사용자 휴대폰의 SMS 통지에서 해당상황의 영상 등을 담아 통지하기 원한다면 Alarm Module에서 SMS를 MMS로 수정하면 해당 기능을 즉시 적용할 수 있다.

3.2 범용 미들웨어의 운영

범용 미들웨어의 상황정보 처리는 그림 3과 같이 처리된다. 센서 등에서 얻어지는 상황 정보는 미들웨어의 Agent IO 모듈을 통해 미들웨어 메시지로 변환되고 이 정보는 3개의 모듈을 목적지로 하여 메시지 테이블에 저장된다. DataBase 모듈에서는 상황정보 메시지를 받아 보관하고, Inference 모듈에서는 상황정보의 이상 여부를 확인하며, Application 모듈에서는 외부 모니터링 도구에서 실시간 모니터링을 위한 XML로 변환된다.



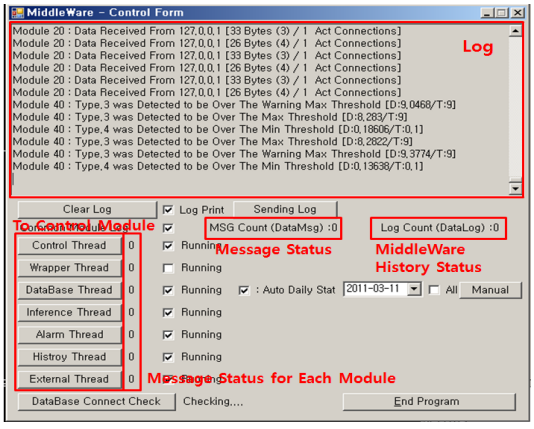
(그림 3) 범용 미들웨어의 상황 처리 다이어그램

범용 미들웨어는 전체 응용 시스템의 안정된 운영을 위해 Wrapper Agent와 Control Agent의 상태를 미들웨어의 Agent IO Module에서 정기적으로 상태를 확인하여 문제 발생 시 사용자에게 통지한다.

4 응용 시스템의 구현 및 성능 고찰

범용 미들웨어 시스템의 성능을 확인하기 위해 저사양의 서버 시스템인 Pentium-III 1.0GHz CPU와 메인메모리 512M, Windows Server 2003 환경에서 실험하였으며 DataBase는 독립된 별도의 서버에서 구동중인 MySQL을 이용하였고, Inference Module은 Rule Based로 구현하였다. 저사양 시스템 환경에서 Wrapper Agent의 데이터를 초당 50개의 데이터를 실시간 모니터링과 처리가 가능하였으나, 50여개가 초과 되면 메시지테이블에 데이터가 모두 처리되지 못하는 지연이 발생하였다.

Q6600 CPU와 메인메모리 4GB, Windows 7 환경에서는 초당 670개까지는 문제없이 처리가 되었으나 680개부터는 상황정보 처리에는 문제가 없었으나 실시간 모니터링이 지연되는 현상이 발생되었다. 이는 실시간 모니터링에 사용되는 TCP/IP 프로토콜을 이용하여 소켓을 생성하고 처리하는데 지연이 발생한 것으로 확인하였다. 이 문제는 실시간 모니터링 부분에서 실시간 데이터 요청을 초당 10프레임 정도로 낮추면 해결되었다.

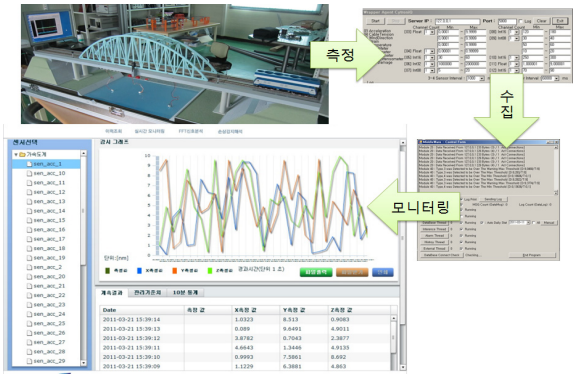


(그림 4) 범용 미들웨어 동작화면

4.1 응용 시스템의 검증

본 논문에서 제안한 범용 미들웨어 시스템을 모형 교량 관리시스템과 모형 철도 관리 시스템에 적용하여 검증하였다.

우선 모형 교량관리 시스템과 모형 철도관리 시스템은 범용성을 검증하기 위해 서로 다른 센서를 이용하였다. 모형 교량에서는 인텔의 IMote와 CytroniQ에서 생산한 광격자 센서를 이용하였으며, 모형 철도관리 시스템에서는 HUINS에서 생산한 UBee 430 센서를 이용하였고, 장치 제어를 위해 철로 교환 시스템을 구현하였다.



(그림 5) 교량관리 시스템의 동작화면

(그림 6) 모형 철로관리 시스템의 동작화면

교량과 철로 관리 시스템을 구현하는데 있어 동일한 미들웨어를 각각 사용하였으며, 서로 다른 센서와 장치는 Agent만 개발하여 이용하였다. 또한 서로 다른 플랫폼의 사용자 모니터링을 위하여 Application 모듈만 각각 개발하여 이용하였으며, 다른 상황정보 처리를 위해 각 시스템별로 알맞은 Rule을 Inference 모듈에 적용하였다.

5. 결론 및 향후 연구

본 논문에서 제안한 범용 미들웨어를 교량관리 시스템과 철로관리 시스템에 각각 적용한 결과, 센서나 장치가 다르더라도 Agent를 이용한 규격화된 상황정보 전송으로 인하여 미들웨어를 수정 없이 바로 이용할 수 있었다. 또한 상이한 상황처리 시스템은 약간의 모듈 수정으로 기능을 바로 수정하여 적용할 수 있었다. 그러나 진정한 상황정보 처리와 인식을 위한 미들웨어가 되기 위해서는 S. Hadim과 N. Modamed가 제안한 기능[10]인 다양한 질의 유형 지원, 센싱 정보 관리, 메타정보 관리, 이기종의 센서 네트워크 통합 지원, 상황정보 생성 및 관리, QoS 보장, 센서노드 미들웨어의 갱신, 센싱 정보의 보안, 센서노드의 위치인식 기능을 모두 보장할 수 있도록 기능의 개선과 검증이 필요하다.

참고문헌

[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 desige challenges for ubiquitous computing applications," Communication of ACM, 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, 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] A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster, "The anatomy of a Context-aware application," Wireless Networks Vol. 8, Issue 2/3, pp. 187-197, 2002.

[6] K. Cheverst, N. Davies, K. Mitchell, A. Friday, "Experiences of developing and deploying a context-aware tourist guide: the GUIDE project," Proceedings of the sixth annual international conference on Mobile computing and networking, pp. 20-31, 2000.

[7] 장세이, 우운택, "ubiHome을 위한 컨텍스트 기반 응용 서비스 모형", 정보과학회논문지: 소프트웨어 및 응용, 제30권 제6호, pp. 550-558, 2003.

[8] 송재훈, 김동균, 이상정, "블루투스를 이용한 상황인식 서비스", 한국통신학회 추계종합학술발표집 제28권, pp. 254-257, 2003.

[9] 박한솔, 최태욱, 정기동, "jini 기반의 context-aware chatting program", 한국정보처리학회 추계 학술발표 논문집, 제10권 제2호, pp. 1177-1180, 2003.

[10] Salem Hadim and Nader Mohamed, "Middleware Challenges and Approaches for Wireless Sensor Network." IEEE Distributed Systems Online, Vol.7, No.3, 2006