u-응용 서비스 지원 멀티 자원 관리 시스템 설계 및 구현

장형근○, 정창원, 주수종
원광대학교 컴퓨터공학과
{jjling, mediblue, scjoo}@wonkwang.ac.kr

Design and Implementation of Multi-Resource Management System for u-Application Services

Hyung-Geun Jang○, Chang-Won Jeong, Su-Chong Joo
School of Computer Engineering, Wonkwang University

요 약

최근 유비쿼터스 컴퓨팅으로 변화됨에 따라 다양한 디바이스의 등장으로 인하여 컴퓨팅 환경을 유지 관리하는 비용이 증가함에 따라 시스템 자원에 대한 관리가 요구되고 있다.

본 논문에서는 여러 클라이언트의 시스템 자원에 대한 모니터링을 통해 다양한 u-응용 서비스 지원을 위한 멀티 자원 관리 시스템을 제안한다. 제안한 시스템은 다양한 클라이언트 디바이스상의 프로세스를 모니터링하여 실시간으로 정보 수집하여 데이터베이스에 저장하며, 또한 각 디바이스의 현재 수행 중에 있는 출력 화면을 확인할 수 있다. 이를 위한 설계 및 구현 환경에 대해 기술하고, 또한 실시간 모니터링을 수행한 결과를 보인다.

1. 서 론

유비쿼터스 컴퓨팅 환경의 변화에 따라 물리적인 인프라를 구성하는 센서, 기기, 장치들이 다양해지고 있다. 이로 인하여 컴퓨팅 환경의 유지 관리 비용이 증가하고 있다[1, 2]. 이와 관련된 연구로는 DMWG(Desktop & Mobile PC Working Group)과 DASH 이니셔티브[3]에서 데스크톱, 모바일, 블레이드 PC 등과 같은 시스템 자원을 관리하기 위해 플랫폼에 종속되지 않으며, 상호 호환성 지원을 위한 목표로 연구 개발 중에 있다.

본 논문에서 제안하는 시스템은 관련연구의 내용을 토대로 여러 클라이언트의 시스템 자원에 대한 모니터링을 통해 다양한 u-응용 서비스를 지원하는데 목적을 두고 있다. 이를 위해 데스크톱과 모바일 PC 관리를 위해 현재 수행 중에 있는 프로세스 및 수행 중인 프로그램에 대한 모니터링에 중점을 두었다. 이를 위해 통신 프로토콜과 메시지 정의 그리고 멀티 스레딩 기법을 적용한 멀티 디바이스 관리 방안과 데이터베이스 구조 그리고 모니터링 수행 결과를 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 대해 기술하고 3장에서는 제안한 시스템의 설계 부분 그리고 4장에서는 구현 내용과 수행 결과 화면을 보이고 5장에서는 결론 및 향후 연구 내용을 기술한다.

이 논문은 2010년 교육과학기술로부터 지원받아 수행된 연구임(지역거점연구단육성사업/헬스케어기술개발사업단)

2. 관련 연구

본 장에서는 관련 연구로 DMWG (Desktop & Mobile PC Working Group) 과 DASH 이니셔티브에서 제시하고 있는 사항에 대해 기술한다.

2.1 DMWG (Desktop & Mobile PC Working Group) 과 DASH 이니셔티브

"DASH Initiative"를 지원하는 주요 조직인 DMWG는 2006년 1월에 조직되어 활동하고 있다. DMWG의 목적은 데스크톱, 모바일, 블레이드 PC 등과 같은 "Targeted Platform"의 시스템 자원을 관리하기 위해, 플랫폼에 종속되지 않으며, 상호호환성을 지원하는 산업 표준 기반 관리 솔루션을 정의하는 것을 목표로 하고 있다. 특히, 경고 및 이벤트 로깅, 원격시스템 전원 제어 및 모니터링, 실행 가능한 (부트)이미지 선택, 제어 및 이동 그리고 부트 프로세스 모니터링, 플랫폼 자산 관리 및 운영체제 이미지 저장 및 복구 지원과 타겟 플랫폼 찾기와 타겟 플랫폼 프로비저닝에 세부 목표로 하고 있다. 이를 위해 소프트웨어 구조 및 CIM 스키마와 프로파일 정의 그리고 보안 기능을 강화한 관리 메커니즘 그리고 상호 운영성 지원을 위한 연구를 진행하고 있다.

2.2 DASH 프로토콜 스택 및 프로파일

DASH는 표준과 웹서비스를 기반으로 데스크톱과 모

바일 클라이언트 시스템에 대한 플랫폼 관리를 위해, DMTF에서 이미 개발하여 보급한 “Web Services for Management(WS-Management)”와 “Common Informa-tion Model(CIM)” 규격을 활용한다. DASH 프로파일에 정의된 CIM 객체에 대한 오퍼레이션을 수행하기 위하여 메시지를 전송하기 위하여 “WS-Management”가 활용된다[4]. 다음 그림 1은 DASH에서 제안하고 있는 프로토콜을 보이고 있다.

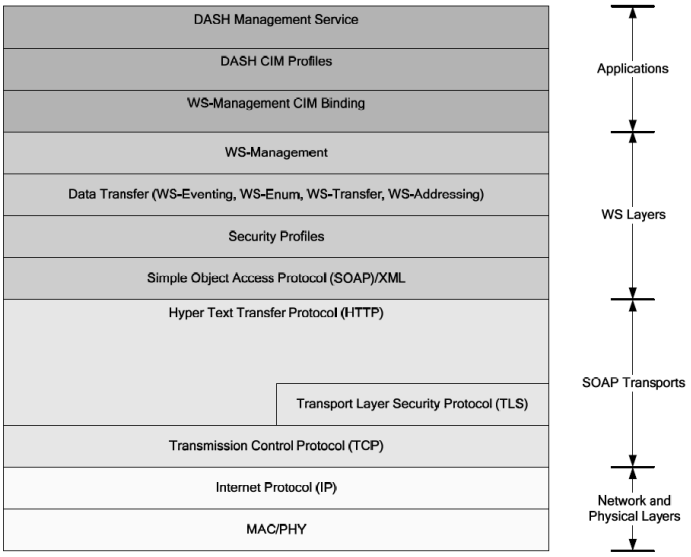

그림 1 DASH 프로토콜 스택

본 논문에서는 관련연구에서 제안하고 있는 목표에 해당하는 데스크톱 PC 또는 모바일 디바이스의 관리를 위한 MRMS(Multi-Resource Management System)을 제안한다. 이를 위해 DASH에서 제안하고 있는 환경과 프로토콜을 참고하여 통신 프로토콜을 정의하였으며, 클라이언트에 해당하는 멀티 디바이스의 관리를 위해 멀티 스레딩 기법을 적용하였다.

3. MRMS의 설계

제안한 MRMS(Multi-Resource Management System)은 다양한 디바이스들의 수행 상태를 관리 유지한다. 그리고 각각의 디바이스에 내제된 시스템과 이를 통합 관리하는 서버 상의 메인 시스템은 상호간의 커뮤니케이션을 위해 필요한 통신 프로토콜을 가져야 한다.

본 논문에서 설계된 MRMS는 그 범위를 좁혀 윈도우 운영체제를 지원하는 모든 데스크톱과 모바일 디바이스에 초점을 맞추어 각각의 디바이스 응용이 자원(실행중인 프로세스들)을 수집하고 서버로 전송하여 서버 응용에서 이를 데이터베이스에 저장하고 저장된 자원을 관리자가 모니터링 할 수 있도록 하였으며 또한, 디바이스의 디스플레이 화면을 서버에 전송하여 정확하고 빠르게 디바이스의 상태를 시각적으로 확인할 수 있도록 하였다.

3.1 디바이스 응용

클라이언트 디바이스를 지원하는 응용은 본 디바이스를 사용하는 사용자에 의해 조작되지 않도록 하기 위해 디바이스에 디스플레이 되지 않으며 부팅이 완료된 상태가 되면 백그라운드 상에서 실행되어 주기적으로 서버와의 연결을 시도한다. 서버 응용이 실행되어 해당 포트가 열리게 되면 디바이스 응용은 연결 시도 중 자동으로 서버와 연결되어 상호간 통신이 가능한 상태에 놓이게 된다.

서버와 통신이 가능한 상태에 놓인 응용은 주기적(분단위)으로 자신의 운영체제에서 실행 중인 프로세스들을 수집하여 서버 응용에 전송한다. 또한 서버 응용으로부터 수신된 특정 메시지를 판별하여 메시지에 따른 동작을 수행한다.

3.2 서버 응용

관리자에 의해 운영되는 서버 응용은 실행 후 그림 2와 같은 초기 화면을 보인다.

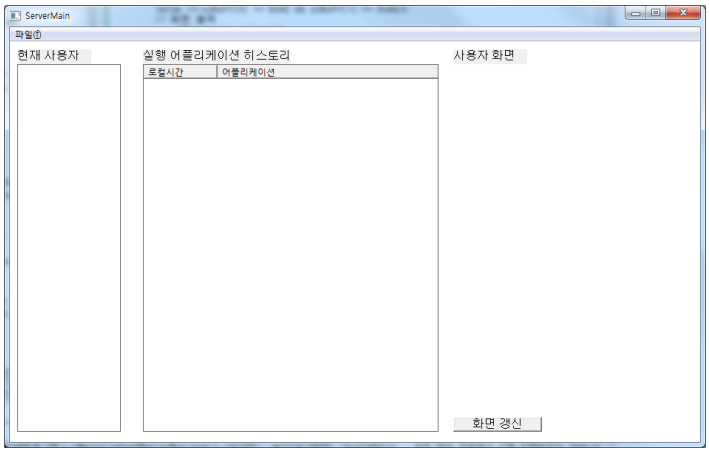

그림 2 서버에서 실행된 초기 응용 화면

화면 구성은 메뉴를 제외한 현재 사용자, 실행 어플리케이션 히스토리, 사용자화면으로 이루어져있다.

① 현재 사용자 - 서버 응용에 접속되어진 디바이스이며 유일한 번호가 부여되어 표시된다.

② 실행 어플리케이션 히스토리 - 데이터베이스에 저장되어 있는 데이터들이 ‘현재 사용자’ 리스트에서 디바이스가 선택될 시 이에 해당하는 로컬시간과 어플리케이션 정보만 추려져 화면에 표시된다.

③ 사용자화면 - 서버 응용에서 디바이스 응용에 특정 메시지인 ‘screen’ 을 보내게 되면 디바이스 응용은 이 메시지를 수신 후 화면정보를 본래화면의 1/5

사이즈로 축소하여 메모리에 저장하며 저장된 데이터를 서버 응용에 전송한다. 서버 응용은 수신된 화면 정보를 변환하여 지정된 위치에 디스플레이 한다. ‘screen’ 메시지가 보내지는 경우는 ‘현재 사용자’ 리스트에서 디바이스를 선택했을 때와 ‘화면 갱신’ 버튼을 눌렀을 때 발생하게 된다.

위와 같이 서버 응용은 디바이스 목록과 프로세스 목록 및 수행된 시간 그리고 사용자화면을 제공함으로 관리자가 이를 확인하여 모니터링 할 수 있도록 하는 역할을 담당한다.

4. MRMS 구현

본 장에서는 3장에서 제안한 시스템의 구현 내용에 대해 기술한다.

4.1 응용간의 통신

디바이스와 서버 각각에 실행되는 두 응용이 통신을 하기 위해선 통신 프로토콜이 필요하며 통신 프로토콜의 규약에 따라 상호간에 통신을 수행한다.

4.1.1 통신 절차

통신 절차는 다음의 두가지 방식으로 수행한다. 첫 번째 방식은 일방 통신이며 두 번째 방식은 양방 통신이다. 그림 3에서는 두 통신 방식 절차를 하나의 시간의 흐름에 따라 나타내고 있다.

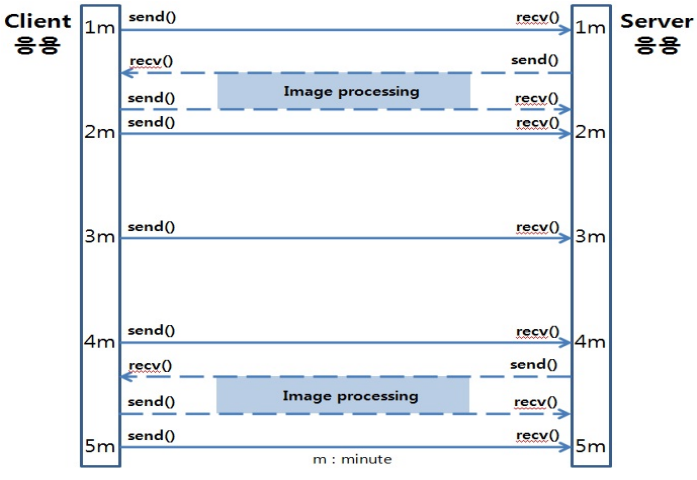

그림 3 두 응용간의 통신 절차

① 일방 통신 - 클라이언트 응용에서 서버 응용으로 보내는 동작만 수행하는 방식이다. 3장에서 언급한 바와 같이 클라이언트 응용은 1분마다 주기적으로 자원을 수집하여 임시공간에 보관한다. 보관된 자원들은 즉시 일정한 구조에 맞게 하나의 데이터 형태로 다시 만들어지게 되며 이렇게 만들어진 데이터는 send() 메소드를 통하여 전송 되고 recv() 메소드로 대기중이던 서버 응용은 곧바로 데이터를 수신 후 데이터 종류를 판별하여 처리한다.

② 양방 통신 - 두 응용이 서로 주고받는 동작을 수행하는 방식이다. 서버 응용은 필요한 경우 임의의 시간에 send() 메소드를 통하여 특정 메시지를 클라이언트 응용에 보내게 되는데 본 시스템에서는 ‘screen’ 메시지가 특정 메시지로 사용되었다. 일단 recv() 메소드를 통해 메시지를 수신하게 되면 클라이언트 응용은 메시지에 맞게 작업을 수행하고 수행된 결과를 일정한 구조에 맞게 다시 만들고 만들어진 데이터를 send() 메소드를 통하여 전송한다. 서버 응용은 반대로 recv() 메소드를 통해 보내진 데이터를 수신하고 데이터 종류를 판별 후 처리한다.

4.1.2 데이터 구조와 역할

전송되는 데이터는 세 가지 형태로 구분되어 각각의 종류에 맞추어 기능을 수행하도록 하였다.

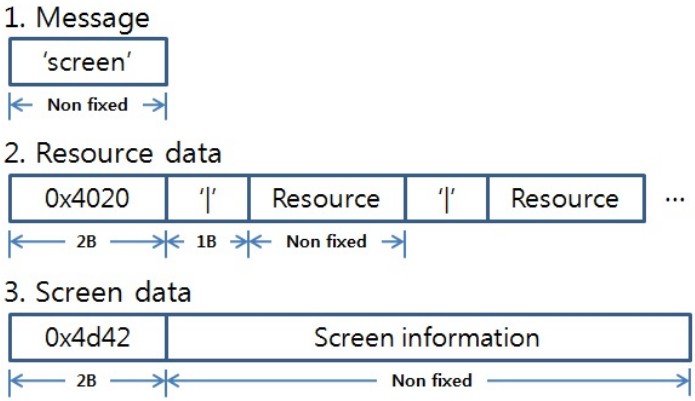

그림 4 데이터 구조

응용 내에서 생성되는 각각의 데이터는 그림 4에서 보는 바와 같다. 먼저 메시지의 경우 서버 응용에서 클라이언트 응용에 어떤 작업을 지시하기 위한 역할을 담당하고 있다. 현재 본 응용에서 메시지는 ‘screen’ 하나만 사용되고 있다. 다음으로 리소스 데이터는 클라이언트 응용에서 생성되며 구성은 헤더 2바이트가 0x4020으로 되어 있고, ‘|’ 문자 1바이트를 구분자로 하여 자원들 즉, 프로세서 목록들이 구분되어 있다. 서버 응용은 본 데이터를 수신하면 데이터베이스에 기록하는 작업을 수행한다. 마지막으로 스크린 데이터 역시 클라이언트 응용에서 생성되며 그 구성은 헤더 2바이트가 0x4d42로 되어 있고 나머지는 실제 스크린 정보로 구성되어 있다. 서버 응용은 본 데이터를 수신하면 서버 응용에 즉시 화면을

보여준다.

4.2 멀티 스레딩을 이용한 멀티 디바이스 관리

서버 응용에서 다수의 디바이스를 관리하기 위한 방법으로 두가지 방법이 존재할 수 있다. 첫번째는 디바이스가 접속될 때마다 큐에 등록한 후 특정 디바이스를 제어할 때 큐에서 해당 디바이스를 선택하는 방법이다. 두번째는 디바이스가 접속될 때마다 하나의 작업 공간을 할당 하여 디바이스가 자동으로 관리되게 하는 방법이다.

첫 번째 방법의 특징은 한 번에 하나의 디바이스만 제어가 필요할 때 사용할 수 있다. 두 번째 방법의 특징은 한번에 다수의 디바이스 제어가 가능하다. 서버 응용의 성능저하면에 있어선 전자의 경우 O(1) 일 때 후자의 경우 O(N)이 되며 이는 디바이스가 접속될 때마다 성능이 저하됨을 의미한다. 본 시스템은 두 번째 방법을 사용하였다. 멀티 디바이스 응용은 각각 주기적으로 데이터를 보내게 된다. 이를 처리하기 위해서 선택적으로 처리하는 방법은 구현하기가 힘들다. 하지만 멀티 스레드로 각각의 디바이스를 처리하게 되면 구현이 간단해 진다. 또한 첫 번째의 경우 모든 디바이스를 한꺼번에 처리하기 위해서는 오히려 O(N)이라는 시간이 걸리지만 후자의 경우 O(1)이면 된다. 이는 모든 디바이스가 한꺼번에 관리되기 때문이다. 그림 5는 멀티 스레딩 기법을 이용한 방법을 보이고 있다.

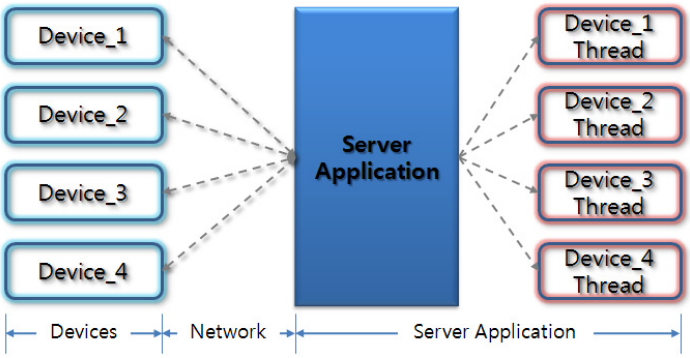

그림 5 멀티 스레딩 기법

서버 응용에서 쓰레드 처리된 AcceptProc() 메소드는 디바이스 응용들의 접속을 위해 항상 대기하고 있는다. 이때 디바이스 응용에서 Connect() 메소드를 이용하여 서버 응용에 접속 요청을 시도하게 되면 서버 응용은 요청을 수락하고 즉시 _beginthreadex() 메소드를 호출해 해당 디바이스의 쓰레드를 생성하여 디바이스의 응용이 종료시까지 유지하도록 한다.

4.3 디바이스 화면 뷰

디바이스의 화면을 서버 응용에서 확인함으로써 디바이스의 상태를 한눈에 알 수 있도록 한 이 기능은 앞서 설명한 바와 같이 먼저 서버 응용에서 'screen' 메시지를 전송하게 되면 디바이스 응용이 이 메시지를 수신하고 화면 정보를 수집한다.

먼저 GetSystemMetrics() 메소드를 이용해 화면 사이즈를 조사하고 CreateDC() 메소드를 이용해 화면 정보를 얻는다. 다음으로 CreateCompatibleBitmap() 메소드로 화면을 데이터로 변환할 수 있도록 비트맵 핸들을 생성 후 StretchBlt() 메소드를 이용하여 비트맵 데이터를 생성하며 마지막으로 다른 장치에서도 동일한 색상으로 표현 될 수 있도록 DDB2DIB() 메소드를 이용하여 변환작업을 수행한다. 이렇게 생성된 화면 데이터는 send() 메소드로 보내지게 되며 서버 응용은 이를 recv() 메소드로 수신한다. 수신된 데이터는 데이터 구조에 따라 Screen Data로 처리되게 되는데 이를 처리 하는 메소드는 DIB2HDC() 이다. DIB2HDC() 메소드는 내부적으로 SetDIBitsToDevice() 메소드를 거쳐서 최종적으로 서버 응용 화면에 출력하게 된다. 그림 6은 디바이스별 화면이 서버응용에서 출력된 결과 화면을 보인다.

그림 6 각각의 디바이스 화면

서버 응용은 디바이스 화면을 1/5 축소된 넓이로 출력하여 보여주며 또한 각각의 디바이스 화면을 유지하기 위해 각각의 디바이스 화면 넓이를 하나로 고정하지 않고 디바이스에 맞는 유연성을 갖도록 하였다.

4.4 데이터베이스 테이블 설계

테이블의 설계는 그림 7에서 보는 바와 같이 4개의 요소로 구성되며 표 1에서 각 요소에 대한 의미를 설명하고 있다.

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID
COM_NAME	VARCHAR2(20 BYTE)	No	(null)	1
APP_NAME	VARCHAR2(100 BYTE)	No	(null)	2
SAVE_DATE	VARCHAR2(9 BYTE)	No	(null)	3
SAVE_TIME	VARCHAR2(9 BYTE)	No	(null)	4

그림 7 테이블 설계

표 1 테이블 요소와 의미

컬 럼 명	의 미
COM_NAME	디바이스 구분을 위한 이름
APP_NAME	실행 중인 자원(프로세스) 이름
SAVE_DATE	서버 호스트의 로컬 날짜(데이터 저장시)
SAVE_TIME	서버 호스트의 로컬 시간(데이터 저장시)

5. 모니터링 수행 결과와 적용 예

본 시스템에서 각각의 디바이스로부터 수집된 정보는 그림 8과 같이 보인다. 사용된 디바이스의 개수는 4개이며 '현재 사용자'에서 그 목록이 나타나 있다. '1104'라는 디바이스를 선택하게 되면 오른쪽에 수행했던 시간과 프로세스 이름이 보인다. 아래쪽에 나타난 화면은 데이터베이스에 기록된 결과를 나타내고 있다.

그림 8 응용 수행 및 database 저장 결과

본 시스템의 수행 결과로 여러 디바이스의 수행 중인 자원에 대해 관리자는 서버 응용을 통해 개별적으로 모니터링하며, 데이터베이스에 저장된 정보의 분석을 통해 다양한 응용에 적용 가능하다. 그림 9은 임의의 제한된 게임응용을 수행하고 있는 디바이스에 대한 모니터링한 결과 화면이다.

그림 9 전체 수행 화면

6. 결론 및 향후 방향

유비쿼터스 시대에 있어서 각종 센서나 시스템들은 하나로 통합하여 관리되는 분산시스템 형태의 시스템이 계속해서 요구 되고 있으며 그 응용 또한 증가하고 있다. 이에 따라 시스템의 성능 향상을 위한 가상화 방법들도 현재는 초기 단계이지만 점차 성행을 보이고 있다.

본 논문은 멀티 디바이스를 유지 관리할 목적으로 MRMS 시스템을 제안하였다. 현재는 멀티 디바이스들의 프로세스 자원 정보를 수집하거나 현재 수행중인 화면을 보이는 기능들을 수행하여 관리자로 하여금 모니터링 기능을 할 수 있도록 제공하는 역할만 담당하고 있다.

향후에는 센서와 관련 디바이스의 전원 및 제어 기능을 포함할 예정이며, 상호운영성 지원을 위해 분산객체 그룹 프레임워크[5, 6, 7] 기반의 소프트웨어 구조를 적용할 것이다. 또한 이를 기반으로 다양한 u-응용 서비스에 적용하고자 한다.

[참 고 문 헌]

[1] Distributed Management Task Force, http://www.dmtf.org/

[2] Shihong Huang, Michael VanHilst, Junwei Cao, Jan Mangs "Remote computing resource management from mobile devices by utilising WSRF", International

Journal of Computer Aided Engineering and Technology 2010 - Vol. 2, No.2/3 pp.199 - 217.
[3] Desktop and Mobile PC Working Group, http://www.dmtf.org/about/committees/DMWGCharter_4-2-2008.pdf
[4] Systems Management Architecture for Mobile and Desktop Hardware, white paper http://www.dmtf.org/standards /published_documents/DSP2014_1.1.0.pdf
[5] 윤영민, 정창원, 주수종, "u-헬스케어를 위한 TMO기반의 액티브 모델", 한국정보과학회논문지 제 13권 5호, pp282-292, 2007.10.
[6] 정창원, 신창선, 주수종, "유비쿼터스 컴퓨팅을 위한 모바일 협업 환경 구축 및 응용", 한국인터넷정보학회 논문지, 제9권 3호, pp35-41, 2008.6.
[7] 정창원, 신창선, 주수종, "u-병원 정보 시스템의 응용 서비스를 위한 멀티에이전트 기반 분산 프레임워크 구축", 정보과학회논문지 15권 11호, pp861-865, 2009. 11.