碩士學位論文

# 센서 네트워크 기반의 Pull/Push 서비스를 위한 개방형 응용 인터페이스 설계 및 구현

濟州大學校 大學院

컴퓨터工學科

金 珪 利

2007年 12月

# 센서 네트워크 기반의 Pull/Push 서비스를 위한 개방형 응용 인터페이스 설계 및 구현

指導敎授 金 度 縣

金 珪 利

이 論文을 工學 碩士學位 論文으로 提出함

2007年 12月

金珪利의 工學 碩士學位 論文을 認准함

審査委員長 李 尙 俊 印

委 員 邊 翔 廈 印

委 員 姜 三 永 印

濟州大學校 大學院

2007年 12月

# Design and Implementation of Open Application Interface for Pull/Push service based on Sensor Networks

**Gyu-Li Kim**

(Supervised by professor Do-Hyeun Kim)

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Computer Engineering

2007. 12.

This thesis has been examined and approved.

Thesis director, Gang-Soon Lee

Thesis director, Byun Sang-Yong

Thesis director, Do-Hyeun Kim

November 2007

Department of Computer Engineering

Graduate School

Cheju National University

# 감사의 글

대학원에 입학한 지가 어제 같은데 벌써 2년이라는 세월이 흘렀습니다. 지난 2년 동안 참으로 많은 일들이 있었는데, 이제 막상 졸업이라 생각하니 시원하고, 섭섭하고, 슬퍼집니다. 석사과정 2년 동안 건강상의 이유로 모든 것에 포기하고 싶었고, 개인의 욕심이 너무 지나쳐서 일이 잘 안 풀릴 때에는 저 자신에 대해서 회의를 느낄 때도 있었지만, 이렇게 대학원 생활을 마무리하고, 새로운 시작을 할 수 있다는 것이 가슴 벅차 오릅니다. 제가 대학원 생활을 잘 마무리 할 수 있었던 것은 저의 가족들, 지도 교수님을 비롯한 연구실의 선후배 여러분들과 동호회, 저의 친구들의 많은 배려와 격려로 이 자리까지 왔다고 생각합니다.

제일 먼저 김도현 교수님께 감사하다는 말씀을 드리고 싶습니다. 연구실에 처음 들어와서 적응이 되지 않았을 때 포기하고 싶었지만, 교수님은 많은 격려를 해주시고, 못한 만큼 화내시기 보다는 오히려 자신감을 많이 심어주신 점은 오랫동안 잊지 못할 거 같습니다. 교수님이 충고 해주신 세 가지 조언을 토대로 맘속에 깊이 새기겠습니다.

그리고 학부 때부터 자주 찾아뵙지는 못했지만, 넓은 시각으로 학문과 논문을 연구할 수 있도록 많은 조언을 해주신 김장형 교수님, 안기중 교수님, 곽호영 교수님, 이상준 교수님, 변상용 교수님, 송왕철 교수님, 변영철 교수님 정말 감사드립니다.

우리 모바일컴퓨팅 연구실 방장 김용우, 같은 대학원 동료인 강민성, 남행우, 어엿한 사회인이 된 강경옥, 강문석, 강성철, 박영윤 선배들 그리고, 해외 어학연수 간 우리 연구실의 유일한 동갑내기 문경보, 연구실에 들어오면 항상 반갑게 맞아주는 김소현, 이영수, 이창영, 부준필, 양현규, 문숙희, 고민정 연구실 동생들 같이 지낸 2년 동안 여러분들 덕분으로 즐겁게 연구실 생활을 할 수 있었습니다.

그 밖에도 일일이 호명을 다 할 수 없지만, 대학원 생활을 잘 할 수 있도록 많은 조언을 해주신 김정희, 한경복, 강은철, 권 훈, 노영식 선배님들을 비롯한 많은 대학원 선배님들 감사합니다. 대학원 동기인 연미언니, 성수오빠, 혜선언니 대학원 생활을 잘 마무리 해서 기쁩니다. 그리고 Deer Shafqat ur Rehman!! I got many help to you about the NS2, mobility and speaking english for a year. So I want to say that thank you. 그리고 학부 때나 대학원을 다닐 때 학교생활을 잘 할 수 있도록 많은 도움을 주신 정은경 선생님, 이정하 선생님 고맙습니다.

힘들 때 많은 위로와 격려를 해주신 붉은악마 제주지회 임지현 지회장님을 비롯한 많은 회원여러분 덕분에 힘이 났던 것 같습니다. 그리고 나의 친구들인 솔거엄마 김지영, 수연엄마 강선홍, 한임복, 이나경 여러분들 덕분으로 힘들 때 마음의 쉼터가 되었습니다. 마지막으로 저의 든든한 후원자 이자, 묵묵히 지켜봐 주시는 저의 부모님 힘들 때 마다 충고를 많이 해준 나의 하나뿐인 언니, 형부, 우리 집안의 든든한 기둥인 오빠와 올케언니!! 감사하고, 앞으로 모든 일에 최선을 다하겠습니다.

# 목 차

# 그 림 목 차

**국문초록**

# 센서 네트워크 기반의 Pull/Push 서비스를 위한 개방형 응용 인터페이스 설계 및 구현

컴퓨터공학과 김규리
지 도 교 수 김도현

최근에 센서 네트워크를 이용하여 다양한 유비쿼터스 응용 서비스를 제공하기 위한 연구가 진행되고 있다. 하지만 응용 서비스에서 센서 네트워크에서 수집된 많은 상황 정보에 접근하기 위한 개방형 표준 인터페이스에 대한 연구가 미비하다. 이에 본 논문은 센서 네트워크 기반의 Pull과 Push 서비스를 제공하는 모델을 제시하고, 응용에서 접근이 용이한 개방형 표준 응용 인터페이스를 설계하고 구현한다.

Pull 서비스에서는 개방형 표준 응용 인터페이스로 WSDL(Web Service Description Language), SOAP(Simple Object Access Protocol)를 중심으로 센서 네트워크 기반의 웹 서비스를 제시하고, 닷넷 기반의 원격 서비스(Remote Service)를 이용하여 센서 네트워크에 수집된 온도, 습도 등의 상황 정보를 제공하는 개방형 표준 응용 인터페이스를 설계하고 구현한다. Push 서비스에서는 서버와 클라이언트 간의 TCP/IP 소켓통신을 이용하여 센서 네트워크에 수집된 온도, 습도 등의 상황정보를 제공하는 개방형 표준 응용 인터페이스를 설계 및 구현한다. 이를 통해서 기존의 특정 데이터베이스 중심의 센서 네트워크의 폐쇄적인 응용 인터페이스를 개방적인 표준 인터페이스로 전환하여 사용자가 여러 곳에 널려 있는 많은 센싱 정보를 쉽게 접근하여 확인할 것으로 기대한다.

## ABSTRACT

# Design and Implementation of Open Application Interface for Pull/Push Service based on Sensor Networks

KIM, Gyu-Li
Department of Computer Engineering
Graduate School
Cheju National University

Recently, it is progressing a study for supporting various ubiquitous application services using sensor networks. But it is not enough a study related standard open application interface to access many collected context information of sensor networks for ubiquitous application services. Therefore, this paper presents pull/push service model based on sensor networks, and implements the standard open application interface for application services. And, pull service model of standard open application interface designs open API(Application Program Interface) based proposed web service model using WSDL(Web Service Description Language), SOAP(Simple Object Access Protocol). Additional, we implement open API to support context information of temperature and humidity using the remote service based on .Net framework in sensor networks. Push model use standard open application interface. But it designs and implements a standard open application interface to provide a state of temperature, humidity collecting in sensor networks based on TCP/IP socket communication between server and client. even though it is not standard open application interface. Consequently, user can

develop easily various application services as supporting the web service instead of closed application interface of sensor networks based on existed specific database. But it is not enough a study related standard open application interface to access many collected context information of sensor networks for ubiquitous application services.

# 약어표

| | |
|---|---|
| USN | Ubiquitous Sensor Network |
| API | Application Program Interface |
| KB | Kilo Byte |
| IPv6 | IP Version 6 |
| BcN | Broadband convergence Network |
| SQL | Structured Query Language |
| NCM | Network Coordinator Manager |
| DM | Data Manager |
| UPnP | Universal Plug and Play |
| CM | Context Manager |
| ZM | Zone Manager |
| AiZ | Action in Zone Decision |
| XML | eXtensible Markup Language |
| MIB | Management Information Base |
| SNMP | Simple Network Management Protocol |
| WSDL | Web Services Description Language |
| UDDI | Universal Description, Discovery, and Integration |
| SOAP | Simple Object Access Protocol |
| HTTP | Hypertext transfer Protocol |
| DBMS | DataBase Management System |
| IP | Internet Protocol |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| SCF | Service Capability Feature |
| SCS | Service Capability Server |

# Ⅰ. 서 론

## 1. 연구 배경

현재 유비쿼터스 컴퓨팅 기술은 센서 네트워크를 이용하여 다양한 서비스를 제공하기 위한 연구와 더불어 개방형 인터페이스를 이용하여 다양한 응용 서비스를 제공하기 위한 연구가 진행되고 있다. 하지만 센서 네트워크기반의 미들웨어나 응용들은 폐쇄된 인터페이스를 지향하고 있기 때문에 센서 네트워크에서 수집된 많은 상황 정보에 접근하기 위한 개방형 표준 인터페이스에 대한 연구가 미비하다. 개방형 표준 인터페이스는 정보 기술을 통합하는 가교 역할을 함으로써 새로운 융합 형 통신 서비스의 개발을 가능하게 하는 기술로 대두되고 있다. 개방형 응용 인터페이스의 특징은 세 가지로 구분된다.

첫 번째, 기존의 공개되지 않았던 소스가 공개되면서 개발자들이 자신만의 서비스를 구현할 수 있다. 두 번째, 자신이 만든 개방형 응용 인터페이스를 타인들과 공유할 수 있다. 세 번째, 이용자는 이미 구현된 데이터베이스와 어플리케이션을 활용하여 보다 쉽게 새로운 웹 서비스를 제공할 수 있다.

## 2. 연구 목적 및 방법

본 논문에서는 센서 네트워크 기반의 Pull/Push 서비스 위한 개방형 인터페이스의 모델을 제시한다. Pull서비스에 구성되어 있는 웹 서비스의 구성요소로 WSDL, UDDI을 이용한 센서 네트워크 기반의 웹 서비스 모델을 제시하고, 닷넷 기반의 원격 서비스(Remote Service)를 이용하여 센서 네트워크에 수집된 온도,

습도 등의 상황 정보를 제공하는 웹 서비스와 TCP/IP 소켓 통신을 이용하여 클라이언트 요청 없이 서버가 실시간으로 클라이언트에게 센서 데이터를 전달 할 수 있도록 Push 서비스를 제시한다.

## 3. 논문 구성

서론에 이어 2장 관련 연구에서는 센서 네트워크 기반의 Pull/Push 서비스를 위한 개방형 응용 인터페이스에 대한 정의 및 관련 요구 기술을 고찰하고, 기존 Pull/Push 서비스와 개선된 Pull/Push 서비스를 비교한다. 3장에서는 센서 네트워크 기반의 Pull/Push 서비스의 모델을 제시하고, 4장에서는 센서 네트워크 기반의 Pull/Push 서비스를 이용한 개방형 인터페이스를 구현한다. 그리고 5장에서는 결론으로 맺는다.

# Ⅱ. 관련 연구

본장에서는 센서 네트워크 기반의 Pull/Push 서비스를 위한 개방형 응용 인터페이스의 연구 방향을 제시한다. 개방형 응용 인터페이스 연구방향은 세 가지로 정의한다. 첫 번째, Pull/Push 서비스를 위한 센서 네트워크에 대한 요구기술, Pull 서비스의 요구기술, Push 서비스의 요구기술을 정의한다. 두 번째, 개방형 응용 인터페이스의 응용 기술 사례에 대해서 분석한다. 세 번째, 센서 네트워크 기반의 응용 기술사례에 대해서 분석한다. 본론에 앞서, 개방형 응용 인터페이스 기술에 대한 정의를 한다.

## 1. 개방형 인터페이스의 구현 사례 분석

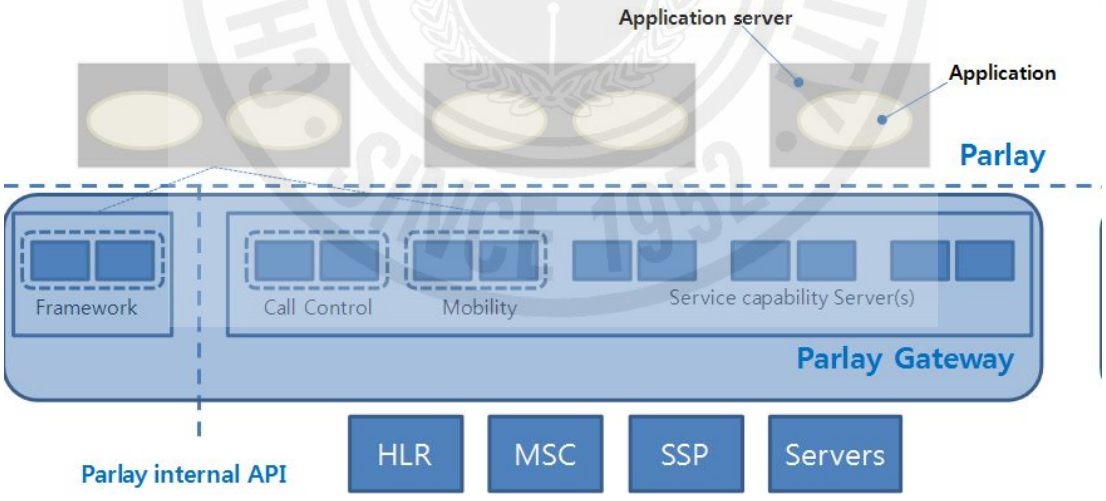


그림 1. Parlay API 구조

대표적인 개방형 응용 인터페이스에는 다양한 통신망에서 일원화된 통신 서비스를 제공하기 위한 OpenAPI가 있다. OpenAPI는 인터넷 기반의 응용 서비스들

의 전화망, 이동통신망, 패킷망 등에서 제공되는 통신 기능들을 쉽고 표준화된 방법으로 이용할 수 있도록 Parlay Group, ETSI, 3GPP 등과 같은 표준화가 추진되고 있다. 현재 Parlay API 4.0, ETSI OSA 3.0, 3GPP OSA 5.0등이 규격으로 표준화되었다. Parlay/OSA API는 그림 1과 같이 Parlay 응용 서버와 게이트웨이로 구성되어 있다. Parlay 응용 서버는 서드 파티 서비스 제공업체에 의해 제공되거나, 망 사업자에 의해 제공될 수 있으며 각 응용 서버 다수의 응용 서비스들을 설치해 구동 할 수 있다. 그리고 Parlay 게이트웨이는 SCS와 프레임워크로 구성이 되어있다. SCS는 네트워크의 자원을 활용할 수 있도록 하는 SCF등을 포함하고 있으며 프레임워크는 SCS와 Parlay 응용 서버와의 인증을 담당한다.

현재 개방형 인터페이스를 이용하여 많은 응용 서비스들이 개발되고 있다. 따라서 본 절에서는 개방형 인터페이스를 이용한 응용 서비스들을 한 사례로 들어서 개방형 표준 API 기반의 액티브 응용 서비스 망 구조의 연구를 한 사례로 나타낸다[15]

개방형 표준 API 기반의 액티브 응용 서비스 망은 DiNS 서비스 플랫폼 기반의 무선 멀티미디어 서비스 제공 시나리오를 나타낸다. 먼저 원격 관리자를 통해 DiNS 서비스 플랫폼 기반의 오버레이 멀티미디어 서비스 망을 구축한다. 멀티미디어 서비스를 제공 받길 원하는 핸드폰 사용자는 DiNS 서비스 플랫폼 기반의 QoS 보장형 응용 서비스 최적화 구간으로 서비스를 요청한다. 그리고 개방형 표준 API 기반의 액티브 응용 서비스 망의 시나리오 순서는 다음과 같다.

첫 번째, DiNS 노드는 캐쉬 매커니즘을 통해 이미 사용자 환경에 최적화된 서비스가 있는지 검색한다. 최적화된 서비스가 존재한다면 그 서비스를 제공하고 그렇지 않다면 Parlay 게이트웨이를 통해 콘텐츠 서버와의 콜을 설정한다.

그리고 두 번째, 콘텐츠 서버에게서 멀티미디어 데이터를 요청한다. 콘텐츠 서버로부터 요청한 멀티미디어 데이터를 받은 DiNS 노드는 멀티미디어 서비스를 핸드폰 사용자에게 맞는 QoS 수준으로 최적화하기 위하여 Parlay 게이트웨이로부터 단말의 상태 및 위치 정보, 단말의 실행환경 정보, 그리고 네트워크 상태정보를 수집하여 서비스를 변환한다. 이렇게 변환된 멀티미디어 서비스를 Parlay 게이트웨이와의 연동을 통해 네트워크 자원을 예약하여 최소한의 지연과 지터 그리고 대역폭을 보장하여 핸드폰으로 전송한다[12].

## 2. 센서 네트워크 기반 기술 및 응용 사례 분석

### 1) 센서 네트워크 기반 기술

센서 네트워크는 센서 노드와 계산, Sensing, 무선 통신을 하는 네트워크로써 특정 지역에서 센서 노드를 설치하여 주변 정보 또는 특정 목적의 정보를 획득하고, 베이스 스테이션이 이 정보를 수집하여 활용하기 위한 서비스 환경이라고 정의할 수 있다.

센서 네트워크의 특징은 노드 당 가격이 저렴하여 대규모 네트워크 구축이 용이하고, 특정 노드에 결함이 생겨도 다른 경로를 이용하여 네트워크가 유지된다는 것이다. 또한 노드의 위치는 필요에 따라서 유동적이고, 네트워크의 수명은 각 노드의 배터리에 의존적이어서 배터리 사용이 최소화되어야 하므로 거의 대부분 sleep 상태였다가 인터럽트 발생 시 활성화 또는 실행 상태로 전환된다.

대표적인 센서 네트워크의 센서 노드 운영체제에는 TinyOS가 있다. TinyOS는 Mote를 위한 이벤트 반응형 운영체제로, C언어의 확장인 NesC로 이루어져 있다. 그리고 저 전력 Ad-Hoc 센서 네트워크이며, 센싱, 검출, 네트워킹/통신, 전력 관리를 하며, 모듈 방식과 재사용이 용이하다. TinyOS의 구조는 그림 2와 같이 크게 스케줄러와 컴포넌트로 구분되며, 컴포넌트는 세부적으로 명령어, 이벤트 핸들러, 프레임 그리고 태스크로 구성된다.


그림 2. TinyOS의 구조

그리고 미국 버클리 대학에서는 TinyDB를 개발하였으며, TinyDB에서는 SQL과 같은 간단한 인터페이스를 제공함으로써 원하는 데이터를 얻을 수 있다. TinyDB는 쿼리가 주어지면 센서 네트워크 환경에서 노드로부터 데이터를 모으고, 걸러내어 싱크노드로 보내는 작업을 수행한다.

### 2) 센서 네트워크 기반 응용기술 사례 분석

본 항에서는 센서 네트워크 기반의 응용기술들을 세 가지 사례를 들어 기술한다. 첫 번째 사례는 유비쿼터스 기반의 건강관리 시스템의 한 예로 유비쿼터스 혈합 측정 시스템이다[19]. 유비쿼터스 혈압 측정 시스템은 인체의 혈압을 측정하여 언제 어디서나 환자의 건강상태를 체크할 수 있는 유비쿼터스 헬스케어 시스템을 구현한 것이다. 구현된 시스템은 혈압 측정 단말기, 자료 수집 베이스 노드, 의료 정보 수집 서버로 구성된다. 구현된 단말기는 ZigBee 프로토콜을 통하여 센서 네트워크를 구성하여 TinyOS가 내장되어 있는 초소형 보드로 설계 되었다.

자료 수집 베이스 노드는 무선 리눅스 단말기로 구성되어 있고, 무선랜을 통하여 센싱된 정보를 서버로 실시간 전송한다. 또한 의료 정보 수집 서버는 단말기에서 얻은 데이터를 저장 관리하며 긴급 상황 발생 시 연계 되는 의료진에게 환자의 상태를 보고하도록 설계되었다.

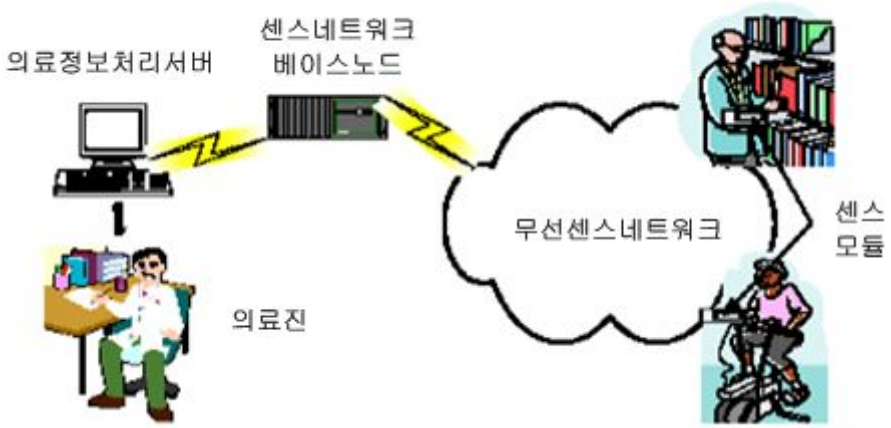


그림 3. 혈압 측정 시스템 구조

그림 3은 혈압 측정 시스템 구조를 나타낸 것이다. 그림 3에서 보듯이 의료서비스 사용자는 신체에 센스 모듈을 장착하고 있어 언제든지 자신의 신체 상태를 모니터링 될 수 있어 긴급 상황이나 정기적인 신체 상태를 무선으로 센서 네트워크로 전송한다. 이 전송된 데이터는 베이스 노드가 취합하여 의료정보처리 서버로 전송된다. 이 서버에서는 다양한 센서로부터 수집된 데이터를 기반으로 의료 서비스를 받을 수 있도록 한다.

이 논문에서는 유비쿼터스 헬스 케어를 위한 프로토타입 수준의 혈압 측정 시스템을 개발하였다. 구현된 시스템은 ZigBee 통신 프로토콜 기반으로 데이터를 전송하며 각 센서 노드에는 TinyOS가 장착되어 이벤트를 처리하며 임베디드 리눅스가 탑재된 게이트웨이는 무선 통신으로 의료 정보를 실시간으로 전송할 수 있다. 언제, 어디서나 의료정보를 의료 전문가에게 전달할 수 있다. 위험 상황이나 주기적 건강을 보고할 수 있는 시스템이다.

두 번째 사례에서는 TinyDB라인 트레이서를 활용한 TinyOS 기반의 센서 데이터 모듈 연구이다[6]. 이동 경로를 주행할 수 있는 라이트레이서를 설계 및 구현하여 쿼리 프로세싱 시스템인 TinyDB를 활용하여 라이트레이서에 센서 노드를 탑재하여 이동 경로 주변의 데이터를 수집하였고, 정지해 있는 특정한 위치에서의 데이터와 비교하였으며, 수집된 데이터를 저장하여 웹 서버에 나타내었다. 또한 사용자의 편의를 위해 임베디드 보드에서 웹 서버로의 접근을 통해서 터치

식으로 쉽게 데이터를 볼 수 있도록 구현하였다. 그림 5는 이동 센서 데이터 처리를 위한 개발 환경을 나타내고 있다. 구성환경은 데이터 감지를 위한 라인트레이서, 센서노드, 게이트웨이 역할을 하는 싱크노드, 데이터를 저장하기 위한 웹 서버, 사용자에게 데이터를 나타내기 위한 임베디드 보드로 구성된다. 이 논문에서는 데이터 통신을 위해서 Micaz 모트는 Atmel ATmeaga128L 마이크로프로세서를 탑재한 매우 작은 센서 노드이고, 데이터를 송수신 할 때 사용된다.

그리고 주위의 빛을 감지해서 Micaz로 전달하기 위해 MTS 310 센서보드를 사용한다. X-Hyper255B 임베디드 보드는 pxa255 마이크로프로세서를 탑재하고 있고, 사용자에게 데이터를 나타내기 위해서 사용된다. 웹 서버에게 Micaz와 MTS310 센서 보드로 구성된 센서 노드에게 데이터를 요청하기 위해서는 싱크노드를 통해서 쿼리를 전송하게 되고, 센서 노드에서는 쿼리에 맞는 조건을 가진 데이터를 센싱하게 되고, 센싱된 데이터는 싱크노드를 통해 웹 서버에 전달되게 된다. 또한 전송된 데이터는 인터넷을 통해서 X-Hyper255B 임베디드 보드에서는 사용자가 데이터 수신을 원하면 확인할 수 있다.

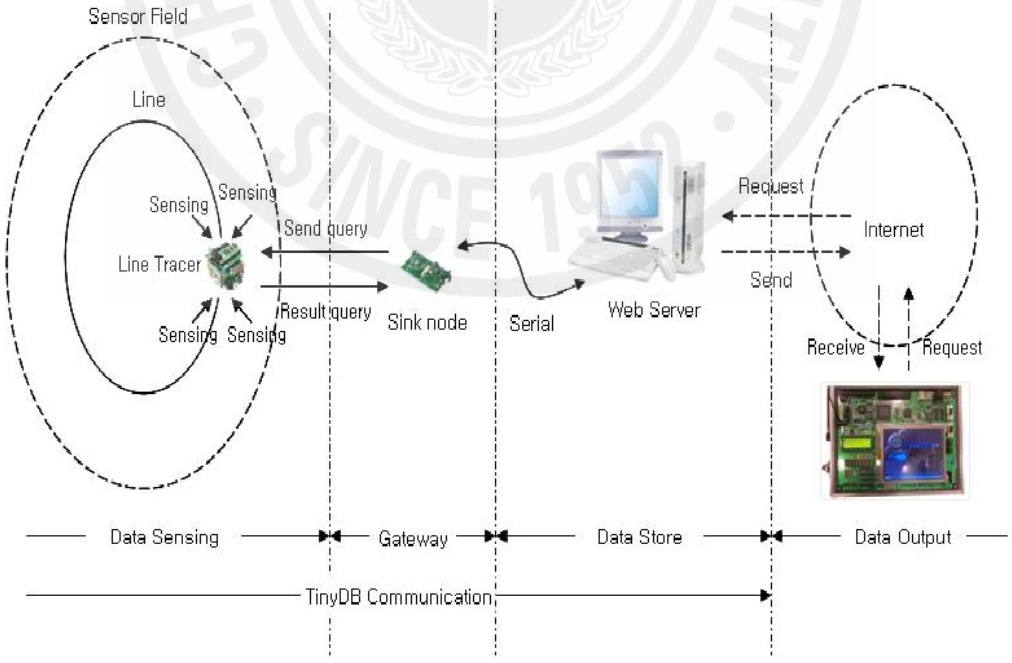


그림 4. 이동 센서 데이터 처리를 위한 개발환경

세 번째 사례는 센서와 미들웨어간의 통신 메커니즘을 지원하는 아키텍처이다 [21]. 여기서 제안한 시스템은 망 관리에서 사용되는 SNMP에 기초하여 센서와 미들웨어간의 통신을 지원하는 것을 목적으로 한다. 이 시스템은 SNMP를 구성 요소인 MIB, 에이전트, 그리고 매니저를 포함하고 있다. 이 시스템은 기본적으로 MIB과 에이전트가 독립적으로 존재한다. 따라서 자연스럽게 데이터를 저장하는 부분과 데이터를 처리하는 부분이 분리가 된다. 즉, 센서에서 감지한 데이터는 에이전트를 거쳐서 MIB로 저장하고, 에이전트는 매니저가 요청한 데이터에 대해서는 MIB에 접근하여 데이터를 전달한다.

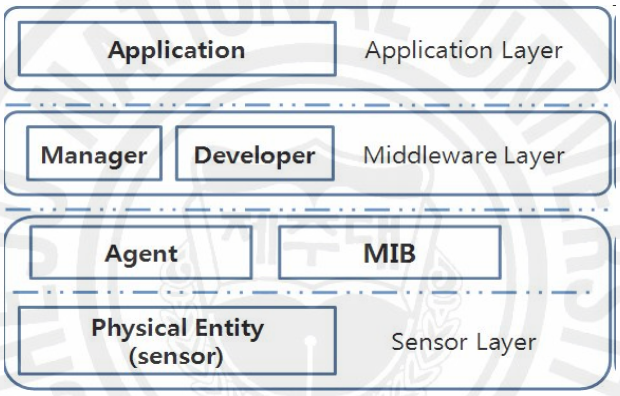


그림 5. 기능 아키텍처

또한 여기서는 센서와 미들웨어간의 통신을 하는데 있어서 데이터가 원활하게 처리 될 수 있도록 SNMP를 도입하였다. SNMP를 구성하는 요소 중에서는 자료를 처리하는 부분은 에이전트가 담당하고, 자료를 저장하는 부분은 MIB이 담당한다. 결과적으로 해당 컴포넌트의 재사용이 가능하게 된다. 또한 MIB과 에이전트의 개발 시간을 단축하기 위해서 SNMP를 이용한 tool kit을 이용한다. 이렇게 함으로써 센서와 미들웨어 사이에 통신하는 컴포넌트를 개발하는 시간이 절약되며 개발자의 수고를 덜게 된다.

그림 5는 본 아키텍처를 도식화 하였다. 본 시스템은 크게 3계층, 6개의 컴포넌트로 구성되어 있다. 6개의 컴포넌트의 역할에 대한 설명은 다음과 같다. Physical Entity는 주위 환경에 존재하는 센서와 컴퓨팅 장치이다. 에이전트는 하

드웨어 센서 혹은 소프트웨어 센서에서 상황 정보를 얻는 컴포넌트이다. 이 에이전트는 두 가지의 연산자를 가지고 있다. 첫째, 센서의 값을 가지고 올 수 있도록 하는 Get 연산자와 둘째, 매니저가 일정한 값을 지정한 후, 센싱 된 값이 그 범위를 벗어날 때 알려주는 Trap 연산자가 있다. 매니저는 Get연산자와 Set연산자를 가진다. Get연산자는 원하는 데이터를 에이전트에 요청할 때, Set연산자는 에이전트에 특정 값을 설정할 때 사용한다. 그리고 매니저의 역할은 크게 두 부분으로 나눌 수 있다. 첫째 자신의 하위에 있는 에이전트를 관리하는 역할을 가진다. 매니저는 에이전트에서 발생된 문제를 알아야 하고 문제가 있는 에이전트가 있다면 적절한 조치를 취해야 한다. 둘째 에이전트에서 받은 데이터를 분석하여 원하는 응용 프로그램에게 전달하는 역할을 한다. Discoverer는 응용 프로그램에게 응용 프로그램이 관심 있는 매니저의 위치를 알려준다. MIB는 Physical Entity에 대한 정보를 객체로 표현하고 이러한 객체들이 구조화된 모임이다. MIB는 객체 자신의 정보나 센서가 감지한 정보, 즉 상황 정보를 저장한다. 애플리케이션은 사용자가 이용하는 프로그램이다.

## 3. Push 서비스 및 사례 분석

### 1) Push 서비스의 정의

Push 서비스는 클라이언트의 특정한 요청 없이 서버 측에서 클라이언트로 컨텐츠를 전송하는 방식으로 정의한다. 그림 6에서는 Push 서비스의 구조를 나타내고 있다.

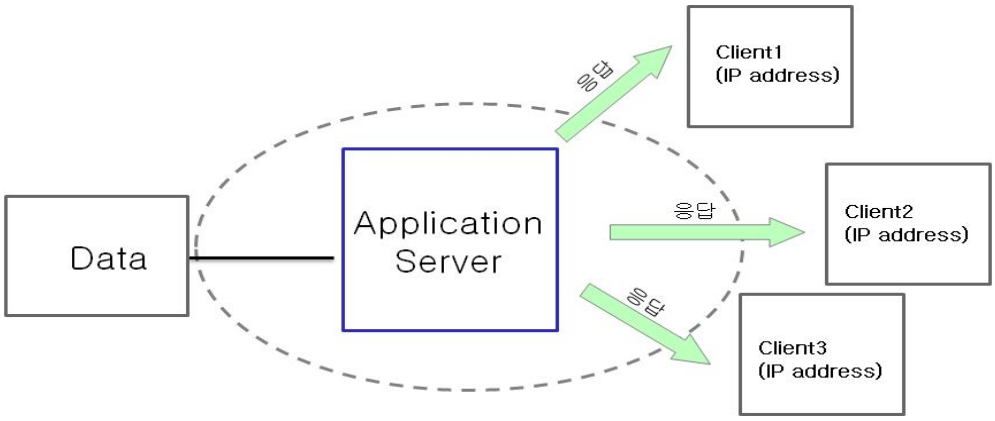


그림 6. Push 서비스의 구조

Push 서비스에서는 데이터를 응용 서버로 보내게 되면 서버에서는 클라이언트에서 접속을 할 경우, 클라이언트 요청 없이 여러 개의 클라이언트 주소로 데이터의 정보를 보내게 된다.

### 2) Push 서비스를 위한 TCP/IP 소켓

Push 서비스를 하기위한 요구기술은 소켓이다. 소켓은 두 개의 네트워크 프로그램이 하부 네트워크에 신경 쓰지 않고 통신하기 위한 방법론을 제공해 주는 도구로써 네트워크의 통신 개념을 전부 내부에 포함하고 있는 간단한 매커니즘을 제공한다. 또한 TCP/IP는 끝점(End Point)라고 알려진 소켓을 사용하여 특정 응용프로그램을 구별한다.

TCP는 패킷을 이용하여 통신을 하는 컴퓨터 네트워크에서 호스트 간의 믿을 수 있는 통신을 하도록 도와주는 프로토콜이며 연결지향형, End-to-End Reliable 프로토콜로서, IP의 상위 계층에서 동작하도록 되어 있다. TCP에서 서버와 클라이언트의 접속은 정해진 규칙에 의해서 진행된다. 연결을 원하는 클라이언트는 먼저 서버에 패킷을 보내는데, SYN 플래그를 세팅하여 서버에게 보내어 서버와 연결을 시도한다. 그러면 서버는 받았다는 의미로 SYN-ACK 플래그를 셋팅한 패킷을 클라이언트에게 보낸다. 이때, 서버의 상태를 절반 열림(Half-Open)이라 한다.

클라이언트는 다시 이 패킷을 서버로부터 받았다는 것을 서버에게 확인시켜주기 위해서 ACK 플래그를 세팅한 패킷을 서버에게 보내고, 서버는 이 패킷을 받음으로써 TCP 연결은 완성된다. 이 다음부터는 정해진 숫자를 순차적으로 증가시켜가면서 특정 데이터를 주고받는다. 이를 'Three way handshaking"이라한다.

TCP의 특징은 네 가지로 정의할 수 있다. 첫 번째 TCP는 연결 지향 프로토콜이다. 각 종단점에는 클라이언트와 서버가 존재한다. 통신 종단점으로 자료를 전송하기 위해 먼저 클라이언트가 연결 요청 메시지를 보내고, 서버는 이를 수락한다. 일단 클라이언트와 서버간의 연결이 맺어지면 각각 표준 read, write 함수를 이용하여 대칭적 양방향 통신을 사용한다. 양단 모두 연결을 닫을 수 있고, 이 경우 다른 쪽은 읽기와 쓰기 명령 시에 이를 감지한다. 따라서 TCP통신을 사용하는 애플리케이션은 언제 작업이 완료되었는지를 원격지로 알릴 수 있다.

두 번째 TCP는 신뢰성이 있다. 일단 연결이 확립되면 TCP는 손실이나 중복이 없이 데이터를 목적지에 확실히 전달한다. 뿐만 아니라 데이터가 안전하게 도착하도록 하기 위해 수신데이터를 점검할 수 있는 체크섬도 같이 전송한다. 주어진 시간에 어떤 확인 응답을 받지 못하면 다시 전송한다.

세 번째 TCP는 슬라이딩 윈도우를 이용한 흐름 제어이다. 슬라이딩 윈도우 매커니즘으로써 채널을 데이터로 가득 채우고 receive Ack를 기다리다가 경험하는 지연시간을 최소한으로 줄인다. 네 번째 TCP 통신은 애플리케이션의 바이트 스트림 기반 보기를 제공한다. 수신 종점에서 TCP 소프트웨어는 세그먼트를 데이터 스트림으로 재결합하고, 애플리케이션 수준의 소프트웨어에 전달한다. 불연속적인 크기로 송수신하는 UDP와 달리 송수신한 데이터의 논리적 경계는 없다.

### 3) 웹 기반 Push 서비스 사례 분석

기존 논문과의 개선된 Push 서비스를 비교하기 위해서 관련 Push 서비스의 사례들을 기술한다. 그러나 기존에 센서 네트워크 기반의 Push 서비스에 대한 연구가 미비하여, 웹 기반의 push서비스들의 사례들을 네 가지로 구분하여 기술한다. Push 서비스의 첫 번째 사례는  Push와 COM을 이용한 클라이언트간의

정보 공유와 이동에 대한 연구이다[22]. 이 연구는 인터넷 상에서 사용자가 그룹을 이루어서 같이 정보를 공유하고, 다른 페이지에서 정보를 얻고자 할 때, 그룹화 된 다른 사용자들도 같은 정보를 서버로부터 제공받을 수 있는 방법을 Push와 Component를 이용하여 사용자들이 그룹을 이루어서 정보를 공유할 수 있는 시스템을 설계하였다.

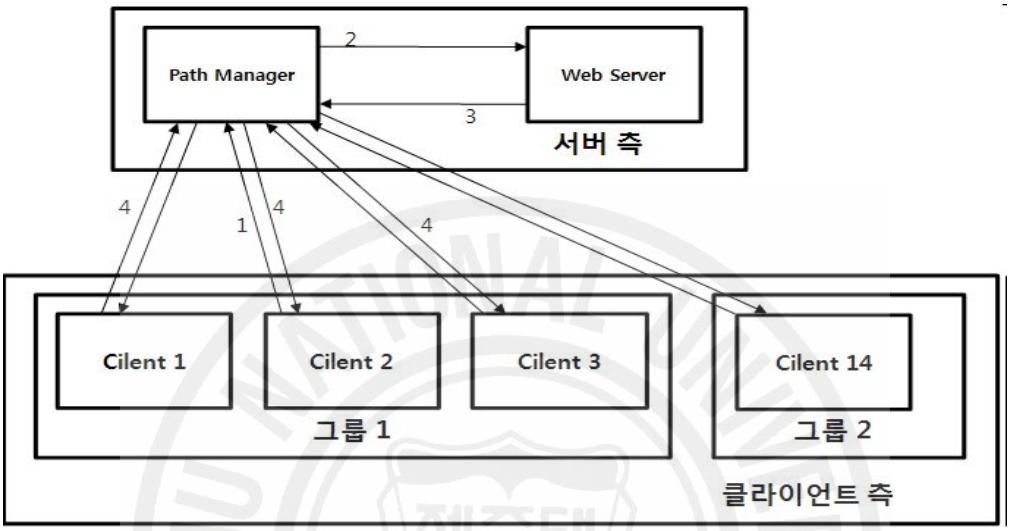


그림 7. 서버 측과 클라이언트 측의 시스템 구성도

두 번째 사례는 Push기반 원격 교육 시스템과 수준별 문항평가 알고리즘에 대한 연구이다[13]. 여기서는 웹 기반 기술 중 능동적 정보전달 방식인 Push기술을 기존 원격교육 시스템에 접목하여 학습자가 인터넷의 학습 데이터베이스에 접속하지 않고도 학습 내용을 제공받을 수 있고, 새로운 학습정보를 실시간 적으로 파악할 수 있는 Push 기반 원격 교육시스템을 제안하였다. Push 기반 원격교육 시스템은 학습 정보의 전송과 관리를 담당하는 Push 서버와 학습 프로필(profile)을 작성하고 컨텐츠의 선택과 정보 전달을 제공받는 Push 클라이언트로 구성된다. 그림 8은 시스템의 전체 구성도를 표현한 것이다.

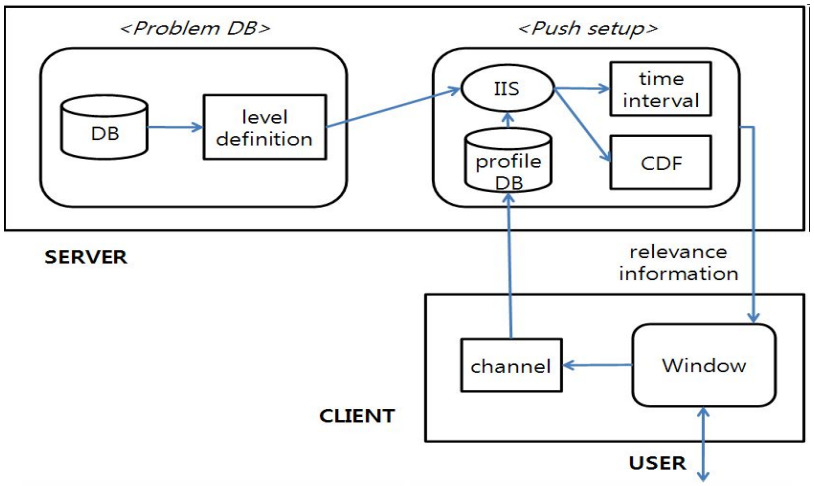


그림 8. DESPC 구성도

Push Server는 Problem 데이터베이스와 Push setup의 두 모듈로 구성되며 기능별로 학습 교과 및 평가를 중심으로 구분한 채널의 등록과 관리를 담당하는 관리영역과 사용자의 정보 및 평가결과 자료를 관리하는 사용자 영역, 학습정보의 제공을 담당하는 정보전송 영역으로 이루어져 있다.

Push 클라이언트는 사용자가 선택하는 프로필을 등록하여 Push 서버로부터 원하는 학습정보를 제공받을 수 있게 한다. 새로운 학습정보나 추가되는 학습내용을 서버로부터 전송받아 실시간적으로 학습자에게 보여준다. Push 서버는 학습자에게 전송할 여러 종류의 학습정보를 구성한다. 즉, 학습정보의 등록과 관리, 각 학습 정보(교과 등)에 포함되는 학습내용 및 평가 문항 등의 관리와 데이터베이스를 관리한다. 그림 9는 Push 서버의 학습 정보 관리 과정을 표현한 것이다.

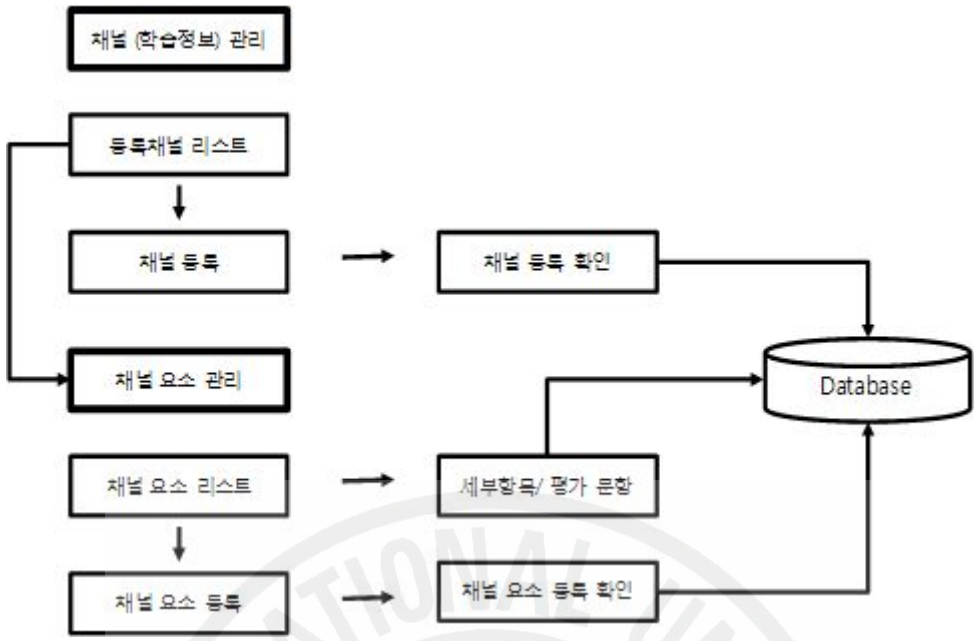


그림 9. Push 서버의 학습정보 관리 과정

세 번째 사례는 Push 기반 이벤트 알림 서비스에 대한 연구이다[15]. Push 기반 이벤트 알림 서비스는 확장성과 효율성을 보장한다. 특히 이벤트 중재자(Broker)를 확장하기 쉽게 구현하였으며, 병렬적 데이터 전송 등을 통하여 데이터 전송 효율성을 제공해 주고 있다. 또한 XML을 사용한 레코드 기반 이벤트 모델을 구현하여 이 기종 호환성 보장과 구현 언어 독립적인 구조를 제공하고 있다. 구현된 이벤트 알림 서비스에 대해 구현 이슈를 살펴보고, 성능을 측정하고 그 결과를 나타냈다.

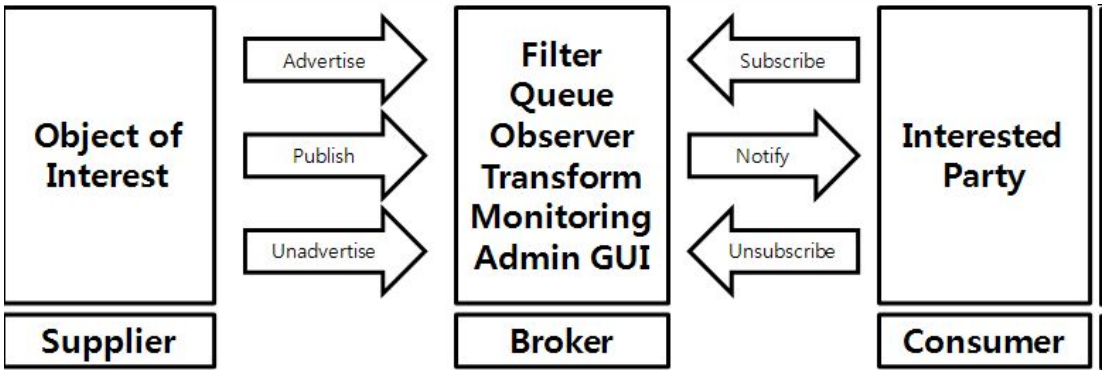


그림 10. Push 기반 이벤트 알림 서비스 개념도

Push 기반 이벤트 알림 서비스는 기본적으로 이벤트 생성자(Producer)가 발생한 이벤트를 이벤트 소비자(Consumer)에게 통지하는 방식이다. 이들을 연결하기 위하여 이벤트 중재자(Broker)들을 두고 있다.

그림 10은 Push 기반이 이벤트 알림 서비스 개념도를 나타낸 것이다. 구성요소로는 이벤트 중재자, 이벤트 생성자가 있다. 중재자는 생성자를 연결시켜 주는 역할을 하며, 부가적인 기능으로 필터링, 큐잉, 이벤트 감시, 모니터링, 관리 GUI, 그리고 형식변환 등을 해준다. 이벤트 소비자는 관심 있는 이벤트의 주제나 이벤트의 패턴 등을 등록(Subscribe)하거나 등록 해지(Unsubscribe), 발생된 이벤트들 중에서 관심 있는 내용을 비 동기적으로 통지 받는다(notify). 이벤트 생성자는 관련 이벤트를 이벤트 중재자에게 광고(Advertise)하거나, 광고 해지(UnAdvertise)를 하며 생성된 이벤트를 통지(Publish)한다.

## 4. Pull 서비스 및 사례 분석

### 1) Pull 서비스 정의

Pull 서비스는 일반적인 서버에서 클라이언트 모델에서는 클라이언트의 요청을 서버가 응답하는 형태로 서비스가 제공되고, 이를 통해 웹에서 URL로 컨텐츠를 요청하고 받아오는 것을 Pull서비스라고 정의한다. 그림 11은 Pull 서비스의 구조를 나타내고 있다.

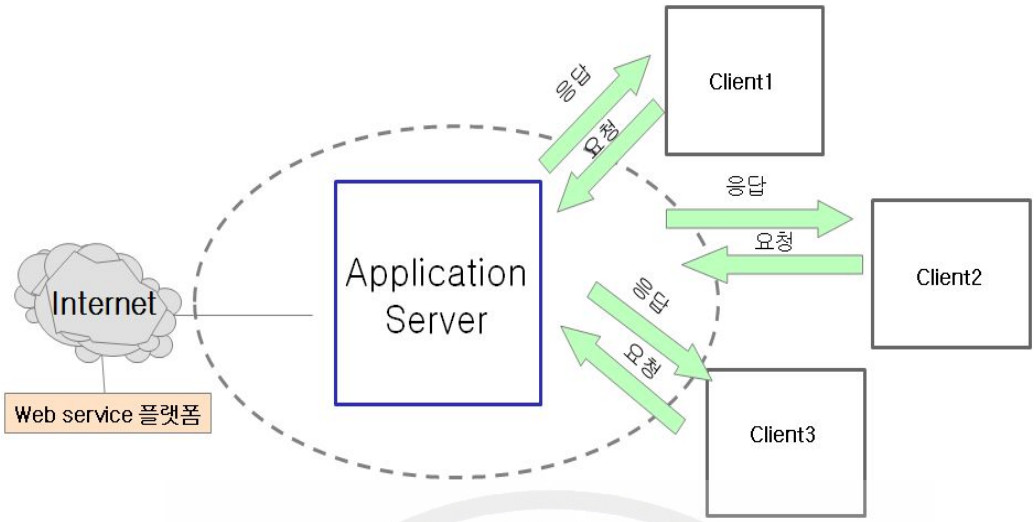


그림 11. Pull 서비스의 구조

### 2) 기존 센서 네트워크 기반의 Pull 서비스 분석

2항에서는 기존 논문과의 개선된 Pull 서비스를 비교 분석하기 위해서 기존 센서 네트워크 기반의 Pull서비스들의 사례들을 기술한다. 그림 12에서는 TinyDB와 SQL을 이용하여 센서 네트워크 기반의 실시간 정보 서비스의 구조를 나타내고 있다. 센서 노드는 센싱 데이터 쿼리를 중심으로 데이터를 수집하여, 센싱 된 데이터들은 싱크노드를 통해 데이터베이스 서버로 보내어지게 된다. 이 응용프로그램에서는 이렇게 실시간으로 받는 데이터 값들을 사용하여 유용한 정보로 변환하여 SQL 데이터베이스에 저장하게 된다.

싱크노드에서 응용프로그램을 거쳐 데이터베이스 서버까지 데이터흐름을 나타내는 데, 이 응용프로그램으로부터 얻은 센싱된 데이터들을 통해서 데이터베이스 서버에 연결하여 저장한다. DBMS로 SQL을 이용하고 실시간으로 센싱된 온도 정보를 저장한다. 여기에는 각각의 센서 노드에서 수집한 데이터를 저장하기 위한 테이블을 갖는다. 클라이언트 응용프로그램은 ODBC를 이용하여 SQL 데이터베이스에 실시간으로 저장되는 센싱 데이터를 읽어 온 다음에 사용자에게 보여준다.

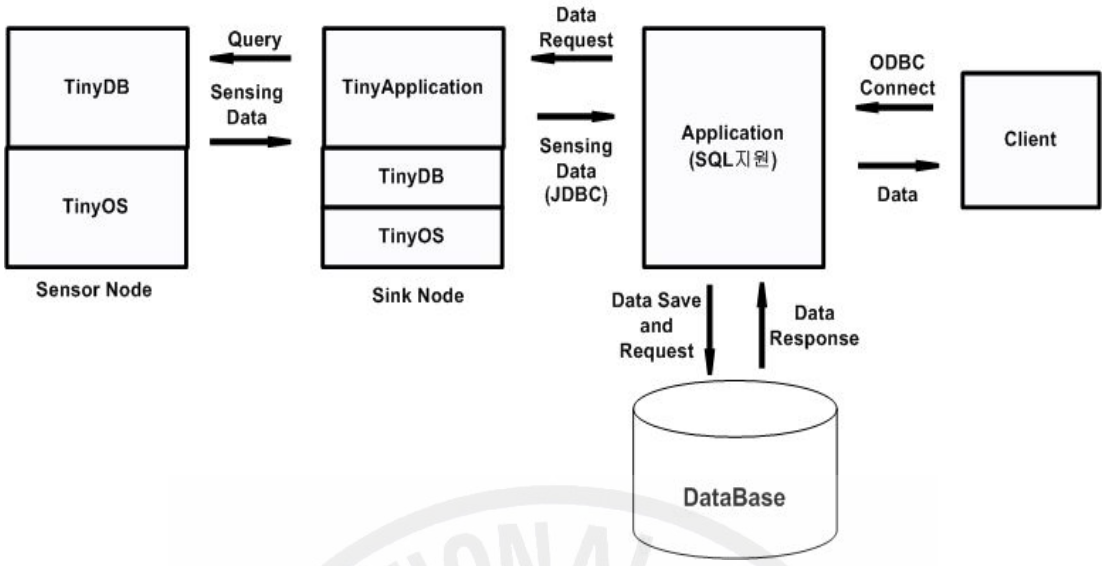


그림 12. 센서 네트워크 응용 서비스 구조

이 프로그램은 센서 노드들에 의해 데이터가 저장된 데이터베이스 서버로부터 현재 온도, 간략한 위치 정보, 온도 상태 값 등을 얻어오게 되며, 가져오는 데이터는 가장 최근의 값을 가져오게 된다. 이때, 온도 상태 값을 가져오는 과정은 멀티스레드로 동작하며 사용자는 어떠한 노드든 현재 상태의 변화가 있다는 것을 실시간으로 확인할 수 있다. 데이터베이스 서버와 클라이언트 응용프로그램의 데이터 흐름을 알 수 있다. 이때 클라이언트 응용프로그램은 데이터베이스 서버에 ODBC로 연결하고 가장 최신의 데이터를 읽어온다. Pull/Push 서비스를 위한 개방형 인터페이스와 기존 연구의 모델과의 차이점을 분석하면, 기존 논문에서는 개방형 응용 인터페이스가 아닌 폐쇄적인 인터페이스를 사용하여 센싱 된 데이터들을 DB 저장소 안에 저장한 뒤, ODBC를 이용하여 클라이언트가 데이터 요청을 하게 되면 서버에서 데이터들을 가져오는 형태였다. 그리고 데이터를 전달받는 클라이언트는 하나에 불과하여 하나의 서버에서 여러 개의 클라이언트 데이터 전송이 안 되기 때문에 동시에 여러 클라이언트들이 센서 정보를 확인하는 것은 어렵다.

### 3) Pull 서비스를 위한 웹 서비스 기술

Pull서비스 하기 위한 개방형 응용 인터페이스를 설계 및 응용을 하기 위해서

요구되는 기술은 웹 서비스이다. 웹 서비스는 웹 상에서 정의된 모듈화 된 소프트웨어 컴포넌트로서, 개방형 표준 데이터 표현 기법인 XML (eXtensible Markup Language)과 인터넷 프로토콜을 결합시킨 새로운 패러다임에 의해서 탄생된 분산 컴퓨팅 기술이다.

그림 13은 웹 서비스 세 가지 기본 오퍼레이션으로 웹 서비스의 사용은 역할에 따라 서비스 제공자, 서비스 등록 저장소로 나눌 수 있는데, 서비스 제공자는 자신의 자원을 개방형 표준에 의해서 서비스화하여 등록 저장소에 공개한다.

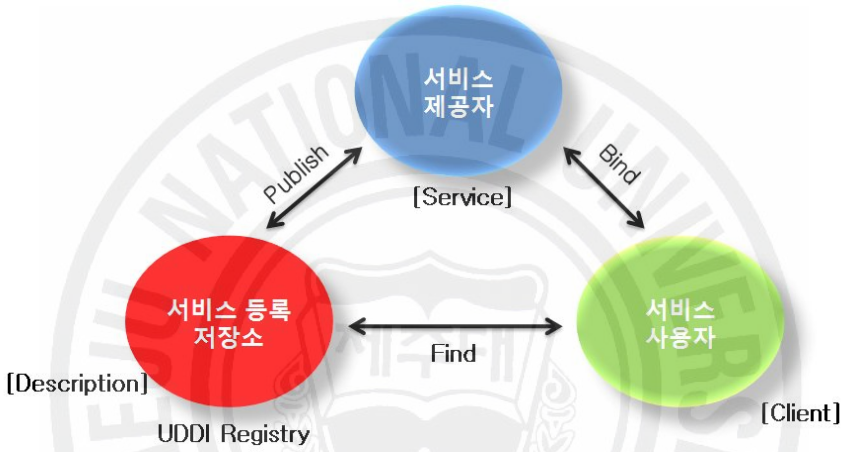


그림 13. 웹 서비스의 구조

서비스 사용자는 서비스 등록저장소로부터 웹 서비스를 탐색하여 찾을 수 있고, 필요한 웹 서비스를 발견하고 서비스 사용자는 서비스를 호출하여 실행할 수 있다. 이를 위하여 웹 서비스에서는 WSDL, UDDI (Universal Description, Discovery, and Integration), SOAP의 세 가지 핵심 기술을 활용한다. SOAP은 XML로 부호화 된 데이터를 전달하기 위한 표준적인 방법을 정의한 메시지 구조에 대한 명세이다. 이것은 또한 메시지를 주고받는 양 끝단 간에 SOAP 메시지를 전달하기 위한 기반 커뮤니케이션 프로토콜로서 HTTP에 바인딩 하는 방법을 정의하고, 응용 프로그램들의 연동이 가능하도록 한다. 웹 서비스를 네트워크 단말 즉, 포트의 집합으로 정의하여 기술한다.

인터넷을 통하여 교환되는 데이터와 포트 타입에 대한 추상적인 기술을 메시지 추상화라 하며, WSDL은 메시지 추상적 정의를 재사용할 수 있도록 한다. 그

리고 포트 타입이란 작업들의 추상적 모임이다. 특정 포트 타입에 대한 구체적인 통신 프로토콜과 데이터 형태의 명세는 바인딩을 구성하며, 네트워크 주소를 바인딩 하도록 정의된다.

그리고 포트의 모임은 하나의 웹 서비스를 정의하며 응용 프로그램이 대상 웹 서비스를 사용할 수 있도록 한다. 서버의 사용자 또는 제공자가 웹 서비스를 발견하거나 등록 할 수 있는 표준적인 방법을 제공하며, 웹 서비스 제공자가 등록하는 웹 서비스의 정보를 저장하기 위하여 Business Entity, Business Service, Binding Template, Technical Model과 같은 핵심 데이터 구조를 정의하며 웹 기반의 글로벌 레지스트리 기능을 제공한다.

# Ⅲ. 센서 네트워크 기반의 Pull/Push 서비스를 위한 개방형 응용 인터페이스

본 장에서는 센서 네트워크 기반의 Pull 서비스 모델을 제시하고, 이를 위한 웹 서비스 기반의 개방형 응용 인터페이스를 설계한다. 그리고 센서 네트워크 기반의 Push 서비스 모델을 제안하고, Push 서비스를 위한 TCP/IP 소켓 기반의 개방형 응용 인터페이스를 설계한다.

## 1. Pull 서비스 및 웹 기반의 개방 응용 인터페이스

본 절에서는 센서 네트워크 기반의 pull 서비스 모델을 제시하고, 이를 위한 웹 서비스 기반의 개방형 응용 인터페이스를 설계한다. 그리고 센서 네트워크 기반의 Push 서비스 모델을 제안하고, Push 서비스를 위한 TCP/IP 소켓 기반의 개방형 응용 인터페이스를 설계한다. 그림 14는 Pull 서비스의 모델을 나타내고 있다. Pull 서비스의 모델은 센서 네트워크, USN 서버, 클라이언트 등의 세 부분으로 구성한다. 센서 네트워크에서는 온도, 습도와 같은 데이터들을 센서노드들끼리 통신을 한다. 싱크노드에서는 데이터 통신을 하는 센서노드들의 데이터들을 센싱한다. 센싱된 데이터들은 USN 서버로 보낸다. USN 서버에서는 센싱 된 데이터들을 가지고 데이터 처리 및 개방형 응용 인터페이스를 이용한 애플리케이션이 이루어진다. 클라이언트에서는 개방형 응용 인터페이스를 이용해서 USN 서버에서 데이터들을 요청하고 서비스를 제공받는다.

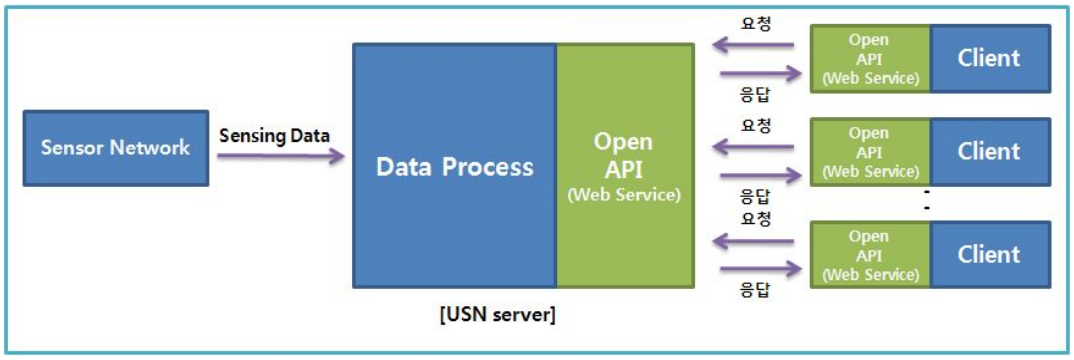


그림 14. 센서 네트워크 기반의 Pull 서비스의 모델 구조

그림 15는 센서 네트워크 기반의 Pull 서비스 구조를 보여주고 있다. 여기서는 Pull 서비스를 제공하기 위해 닷넷 기반의 프레임워크(원격 서비스)를 이용하여 센서 네트워크 기반의 웹 서비스 개방형 응용 인터페이스를 설계한다. Pull 서비스의 구조는 센서 네트워크, 데이터베이스, 서버, 클라이언트 네 부분으로 구분된다. 센서는 온도, 습도 등의 데이터 정보를 여러 개의 센서 별로 통신을 하여, 각각의 데이터들을 싱크 노드로 보내서 데이터들을 센싱하여 수집한다. 데이터베이스에서는 수집된 온도, 습도 등의 센싱된 데이터들을 저장한다. 서버는 닷넷 프레임워크 기반으로 응용을 이용하여 데이터 처리를 도와주는 역할을 한다. 서버의 구성으로는 데이터 매니저, 서비스 제공자, WSDL로 이루어져 있다. 데이터 매니저에서는 센서에서 센싱된 데이터들을 서비스 제공자와 클라이언트가 요청하여 서버와 클라이언트 간의 데이터를 접근할 수 있도록 관리 하는 역할을 담당하고 있다.

그림 15는 그림 14의 Pull 서비스 모델을 지원하기 위한 닷넷 프레임워크를 이용한 센서 네트워크 기반의 Pull 서비스 구조를 보여주고 있다. Pull 서비스의 원격 객체에서도 그림 15와 같은 구조를 갖는다. 그리고 클라이언트에서는 서버에서 WSDL로 부호화 된 XML 데이터를 보고 동적으로 프록시를 생성하여 XML 데이터 정보를 웹으로 배포하게 된다. 클라이언트에서는 원격 객체를 참조할 수 있는 프록시 객체가 있어야 한다. 원격 객체는 원격 서비스를 하기 위해서 사용되는 객체이다. 원격 객체는 클라이언트와 통신하기 위해서 서버 채널을 사용한다. 채널에서는 전송되는 메시지들을 적절한 부호화를 거쳐 통신 상에서 전송되어질 데이터로 변화하게 된다.

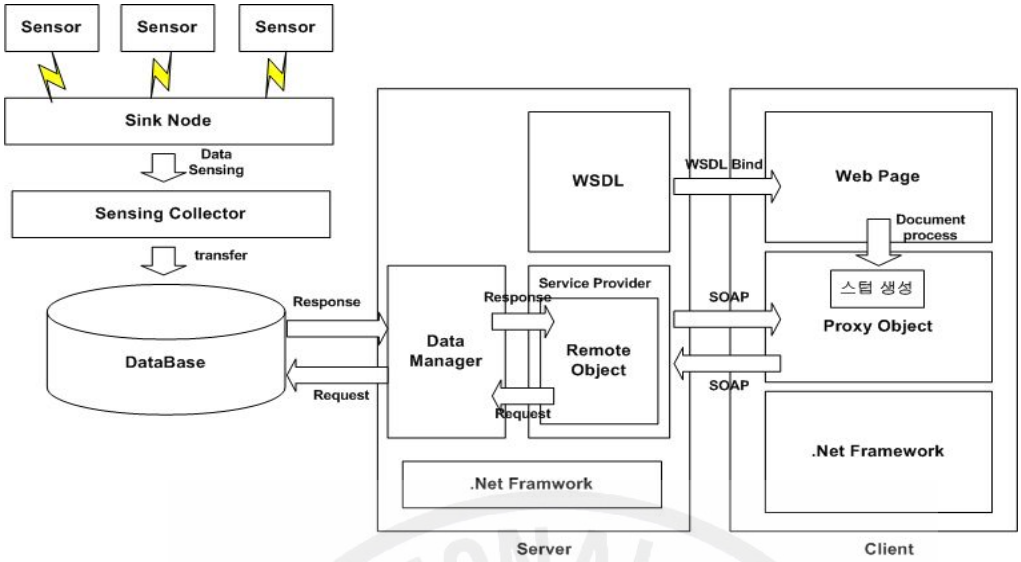


그림 15. 센서 네트워크 기반의 Pull 서비스 구조

이때, 부호화를 담당하는 것이 포멧터인데, 여기서는 SOAP 포멧터를 이용한다. SOAP 포멧터는 데이터를 XML 형식의 SOAP 방식으로 부호화한다. WSDL 문서에서 생성된 코드는 원격 객체에 바인딩하여 SOAP 메시지를 생성하고, 전송시키는 스텁 역할을 맡게 된다. 원격 객체에서는 클라이언트와 서버 사이를 넘나드는 참조 값을 이용하게 되는데, 클라이언트에 존재하는 이 원격 참조 값을 프록시라고 한다. 클라이언트에서도 원격 서비스와 통신하기 위해서 클라이언트 채널을 가지고 있으며 채널로 전송되는 메시지 또한 SOAP 포멧터를 통해서 부호화 하여 웹 서비스화 한다.

## 2. Push 서비스 및 TCP/IP 소켓 기반의 개방형 응용 인터페이스

그림 16은 Push 서비스 모델을 나타내고 있다. Push 서비스 모델의 구성은 센서 네트워크, USN 서버, 수신자 측 데이터 목록과 클라이언트로 이루어진다. Push 서비스는 Pull 서비스와 마찬가지로 센서 네트워크 기반의 개방형 응용 인터페이스를 이용하여 USN 서버가 센싱 정보를 클라이언트에게 쉽게 제공할 수 있도록 Push 서비스를 설계한다. Push 서비스의 모델은 Pull 서비스 센서 네트워크에서 데이터

베이스 처리하는 부분까지의 과정은 같다. Push 서비스에서는 USN 서버에서 클라이언트와 소켓통신을 이용하여 센싱 정보를 전달하여야 하기 때문에, 수신 측 데이터 목록에서는 수신자 측의 데이터, 즉 클라이언트 정보를 입력하여 저장한 후에 USN 서버로 전달한다. 서버에서는 수신자 측의 데이터 목록을 받아서 클라이언트로 센서 데이터 정보를 전달한다. 그림 17은 센서 네트워크 기반의 Push 서비스 구조이다. 여기서는 Push 서비스를 위해 TCP/IP 소켓을 이용하여 미리 접속한 클라이언트에게 센싱 데이터를 수집하도록 제공한다.

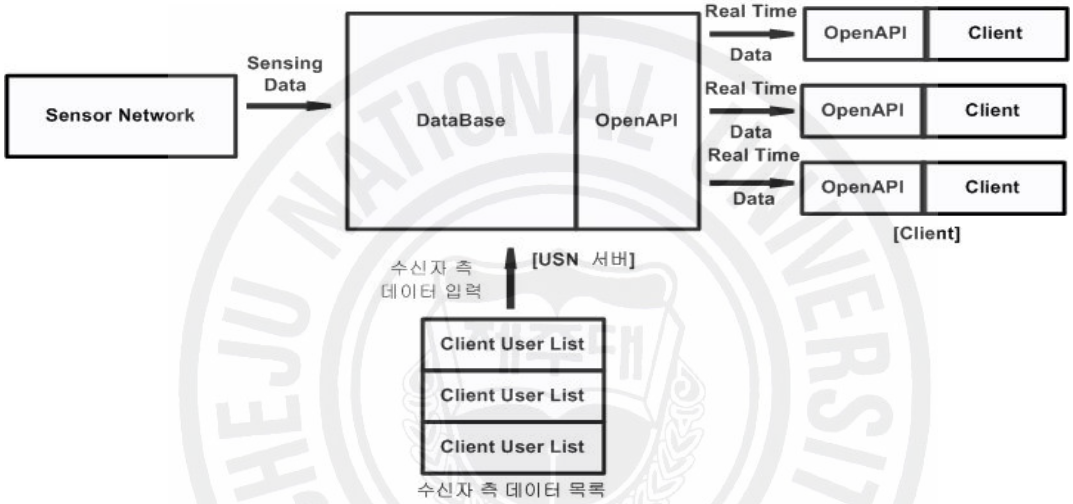


그림 16. 센서 네트워크 기반의 Push 서비스 모델 구조

Push 서비스는 센서 네트워크, USN 서버, Client Configuration Manager, 클라이언트 정보를 저장하는 데이터베이스(클라이언트 리스트), 클라이언트 다섯 부분으로 구성한다. 센서 네트워크는 여러 개의 센서들로 이루어져 있는데, 각각의 센서들은 온도, 습도, 조도등과 같은 정보들을 통신하여 센서 정보들을 센싱한다. Client Configuration Manager에서는 클라이언트의 이름, 포트번호, 아이피 주소, URL 주소를 입력하여 클라이언트 정보가 있는 데이터베이스에 저장한다.

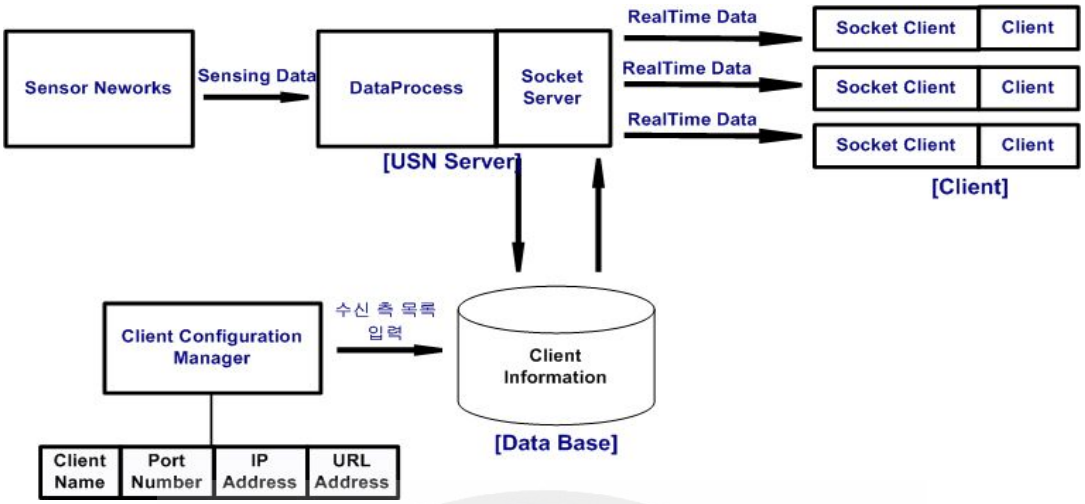


그림 17. 센서 네트워크 기반의 Push 서비스 구조

그리고 데이터베이스에서는 클라이언트 이름, 포트 번호, 아이피 주소, URL 주소의 정보를 포맷한다. 저장된 클라이언트 정보 소켓서버로 보내어지게 되는데, 클라이언트 정보를 받은 소켓 서버에서는 클라이언트가 요청 없이 접속할 때까지 대기하고, 클라이언트가 요청 없이 접속을 하게 된다면 서버에서는 이와 동시에 클라이언트의 아이피 주소, 포트번호를 체크하여 클라이언트 정보와 접속한 클라이언트의 정보가 일치하면 센서 정보들을 전송한다.

# Ⅳ. 센서 네트워크 기반의 개방형 인터페이스 구현

## 1. Pull 서비스

### 1) 서버를 위한 웹 서비스 기반의 개방형 응용 인터페이스

그림 18는 닷넷 프레임워크에서 구현한 웹 서비스는 컴파일 한 후 HTTP POST 프로토콜로 사용하여 메서드를 호출하는 과정을 나타내고 있다. OpenAPI에서 웹 메소드를 정의하여 컴파일 한 뒤 웹 서비스를 이용하여 각각의 메소드들을 호출한다. 이에 웹 서비스는GetSensorData와 GetSensorInformation을 이용하여 두 가지서비스에서 제공하는 속성 값을 요청한다.

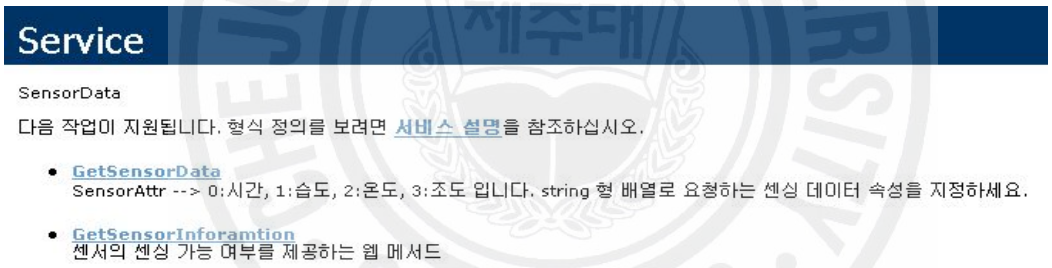


그림 18. HTTP POST 프로토콜을 이용한 메소드 호출

그림 19에서는 시간, 습도, 온도, 조도에 대한 센서 정보들을 확인할 수 있도록 메서드 호출을 하고 있다. 그러나 SensorAttr 배열을 쓰기 때문에 GetSensorData 테스트부분에서는 메소드가 호출이 되지 않고, 결과 값을 확인할 수 없다. 센서 데이터의 결과값을 확인 할 수 있도록 하기위해서는 Pull 서비스의 Client 부분에서 서비스를 제시한다. GetSensorInformation은 HTTP POST 프로토콜을 사용하여 센서의 센싱 가능여부를 확인한다. 센싱 데이터의 속성 그룹, 노드 아이디, 채널의 매개변수를 입력하여 자신이 원하는 센서 데이터의 센

싱 여부를 메서드를 통해 호출한다. 그림 20에서는 GetSensorInformation 메서드를 호출하여 테스트 하는 것을 나타내고 있다. 여기에서는 매개변수 입력 폼에 그룹 1, 노드 아이디 1, 채널 아이디 1이라는 값을 입력하고 호출 버튼을 누르면 각각의 그룹, 노드 아이디, 채널 아이디의 센싱 여부를 확인하는 것을 보여주고 있다.

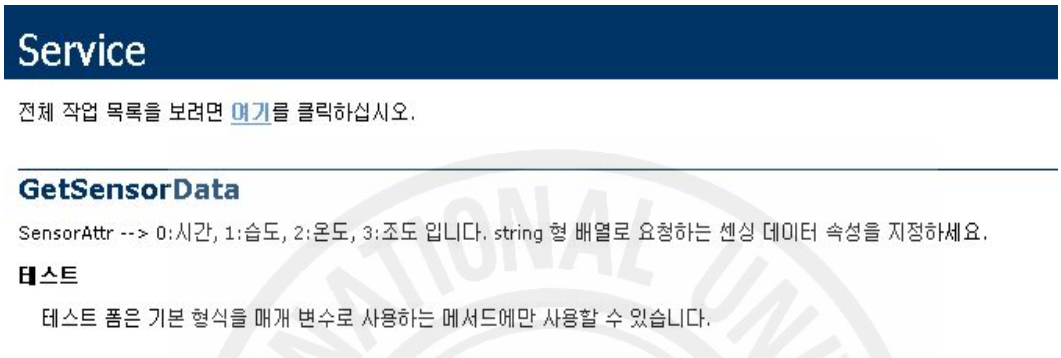


그림 19. GetSensorData의 메소드 호출

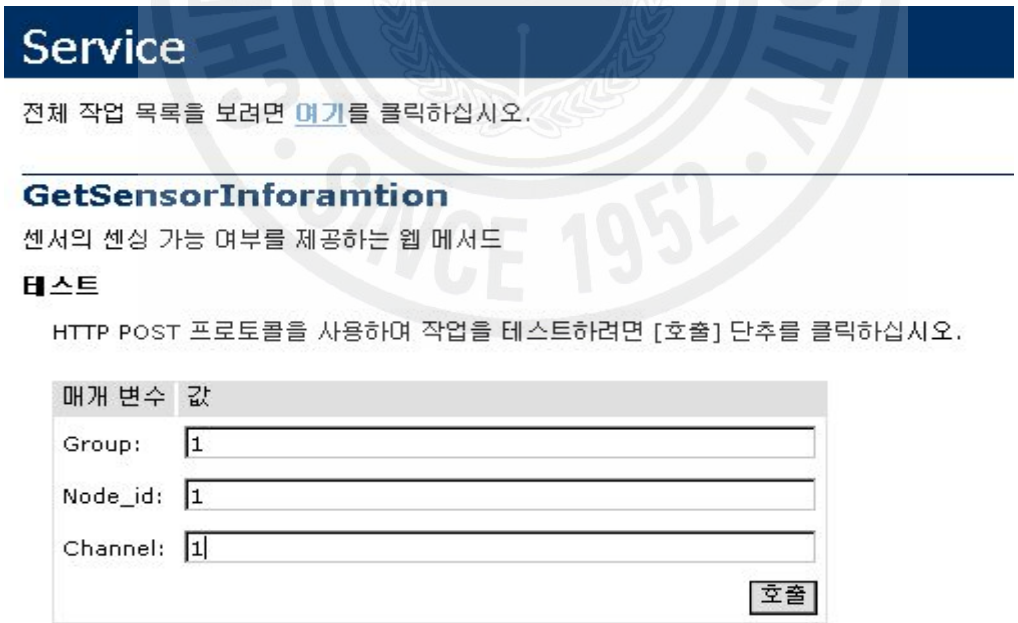


그림 20. GetSensorInformation의 메소드 호출

그림 21에서는 웹 서비스를 통해 XML로 인코드 된 센서 데이터를 전달하기 위한 센싱 가능여부를 표준적인 방법으로 정의한 메시지 구조를 보여준다. 센서 데이터의 그룹, 채널, 노드 아이디의 정보를 SOAP을 이용하여 만든 XML 형식의 테이블에 들어있는 데이터 속성 값을 통해 센싱 여부를 확인할 수 있다.

```
<?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfString xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://203.253.213.222/WebWoo/">
    <string>Y</string>
    <string>Y</string>
    <string>Y</string>
  </ArrayOfString>
```

그림 21. Pull 서비스의 XML 테이블

### 2) 클라이언트를 위한 웹 서비스 기반의 개방형 응용 인터페이스

그림 22는 Pull 서비스를 제공받는 웹 서비스의 클라이언트를 나타내고 있다. 웹 서비스 클라이언트의 구성은 세 가지로 이루어진다. 센서 아이디, 그룹 아이디, 채널번호를 선택할 수 있도록 설정 값을 입력하는 부분과 자신이 원하는 센싱 속성을 선택할 수 있는 체크박스 부분, 체크박스에서 선택한 센싱 데이터의 정보를 확인할 수 있도록 나타내는 부분으로 이루어져있다.

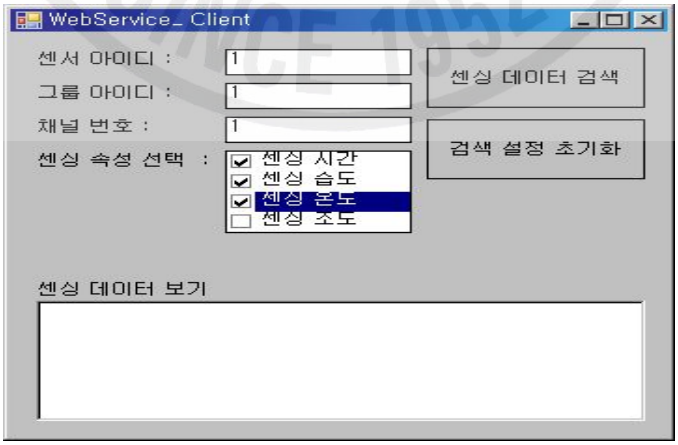


그림 22. 웹 서비스 클라이언트 입력 부분

센싱 속성 선택 체크박스에서는 센싱 된 데이터들의 정보를 쉽게 확인하게 위해서 센싱 시간, 센싱 습도, 센싱 온도, 센싱 조도를 선택할 수 있도록 제공한다. 웹 서비스 클라이언트에서는 센싱 데이터들의 정보를 쉽게 찾을 수 있도록 하기 위해서 센서 아이디, 그룹 아이디, 채널 번호, 센싱 속성선택을 체크박스로 설정하고, 설정된 입력 값은 센싱 데이터 검색을 통해 센싱 데이터를 확인할 수 있다.

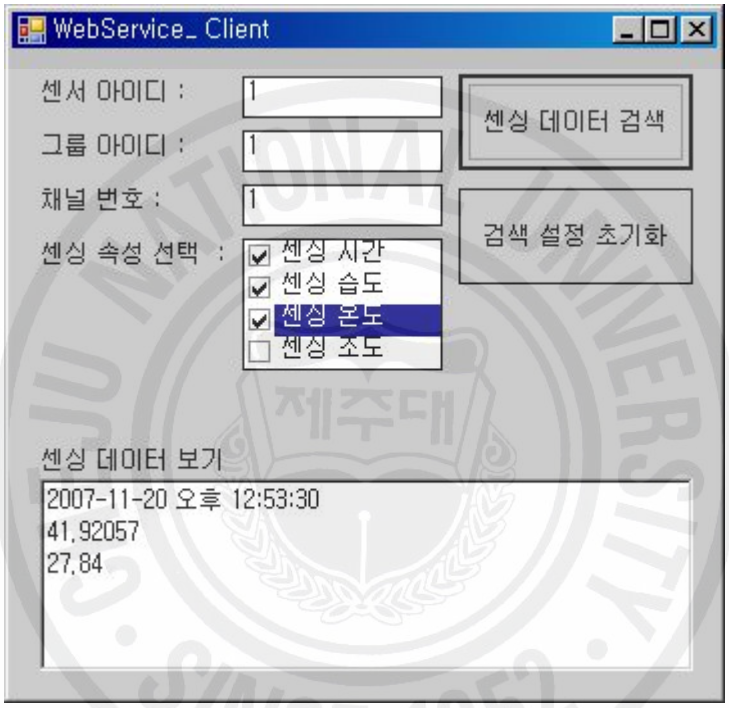


그림 23. 웹 서비스 클라이언트 출력 부분

예를 들면 센서 아이디 번호가 1번이고, 그룹 아이디가 1번, 채널번호를 1번으로 값을 지정하고, 센싱 속성 선택 체크박스에서 센싱 시간, 센싱 속도, 센싱 온도를 선택한다. 그림 23에서처럼 각각의 입력된 값을 통하여 현재 들어온 데이터들을 나타내는 부분에서 확인할 수 있다.

## 2. Push 서비스를 위한 TCP/IP 소켓 기반의 개방형 응용 인터페이스

TCP/IP 소켓을 이용하여 서버와 클라이언트 간에 원활한 데이터 전송을 하기 위해서, Client Configuration Manager, 클라이언트 정보 리스트(데이터베이스) 소켓 서버와 소켓 클라이언트의 네 부분으로 구현하였다. 각각의 역할들은 아래와 같이 기술한다.

그림 24는 Client Configuration Manager의 데이터 입력 폼을 나타내고 있다. 클라이언트 매니저는 클라이언트 정보를 확인하기 위해서 클라이언트 이름, 포트번호, 아이피 주소, URL주소를 입력한다. 입력된 클라이언트 정보들은 클라이언트 정보가 저장되어있는 데이터베이스로 저장된다. Client Configuration Manager를 쓰는 이유는 USN 서버에서 센싱 된 정보를 클라이언트로 보내야 하는데, 데이터 정보를 보내기 위해서는 클라이언트 정보를 알아야 한다. 따라서 Client Configuration Manager에서는 클라이언트 정보들을 포맷하여 클라이언트의 정보가 저장되어 있는 데이터베이스에 클라이언트 이름, 포트번호, 아이피주소, URL 주소를 저장한 후, 클라이언트 아이피 주소와 포트 번호로 센서 데이터를 전달하기 쉽도록 서비스를 제공한다.

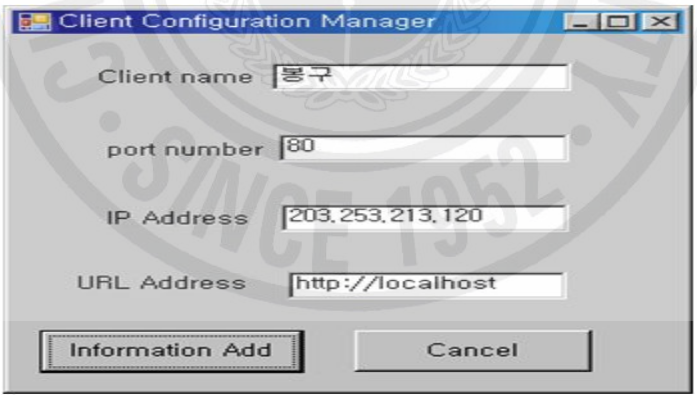


그림 24. Client Configuration Manager 입력 폼

그림 25는 Client Configuration Manager에서 입력한 클라이언트 이름, 포트번호, 아이피주소, URL 주소의 정보들을 저장하기 위해서 만든 클라이언트 정보 리스트를 나타내고 있다. 예를 들어 설명하면, Client Configuration Manager에서 클라이언트이름을 봉구, 포트번호를 80, 아이피주소를 203.253.213.120, URL 주소

를 http://localhost로 설정 창에 값을 입력하게 되면 데이터베이스에 테이블이 추가가 되면서, 클라이언트 정보 리스트에 있는 테이블에 삽입되는 것을 확인 할 수 있다.

| | Clientname | Port_number | IP_address | URL_address |
|---|---|---|---|---|
| | 짱구 | 80 | 203.253.213.133 | http://local |
| | 영구 | 80 | 203.253.213.182 | http://www.chej... |
| | 맹구 | 80 | 203.253.213.133 | http://localhost |
| | 땡구 | 80 | 203.253.213.105 | http://localhost |
| | 병구 | 80 | 203.253.213.133 | http://localhost |
| | 봉구 | 80 | 203.253.213.120 | http://localhost |

그림 25. 클라이언트 정보 리스트 데이터

본 절에서 제안하는 Push 서비스는 소켓으로 통신하는 것이 가장 적합하다. 따라서 Push 서비스는 소켓을 이용하여  서버와 클라이언트 간의 통신할 수 있도록 구현한다. 기본적으로 소켓통신은 TCP통신과 UDP통신으로 이루어진다. Push 서비스에서 제안하는 기술은 클라이언트 정보 리스트를 USN 서버에서 전달하고, 서버에서는 정보를 받고 클라이언트의 IP주소와 포트번호를 이용해서 클라이언트에게 센싱 된 정보를 전달하기 때문에 본 연구는 TCP통신 방법을 이용하였다. TCP의 경우 소켓을 통한 데이터의 송/수신이 이루어지기 전에 컴퓨터간에 소켓끼리 통신할 수 있도록 “연결”이 이루어져야한다.

TCP의 경우 신뢰성 있는 통신을 위해 Three Way Handshaking이라는 방식으로 클라이언트는 서버와 연결된다. 따라서 제안하는 Push 서비스의 동작원리는 다음과 같이 정의한다. Push 서비스의 클라이언트/서버간의 통신이 되기 위해서는 서버와 클라이언트의 위치가 변경되어야 한다. 그 이유는 데이터를 보내는 쪽이 클라이언트이기 때문이다. Push 서비스 측에서 서버와 클라이언트로 구분되어 있지만, 실질적으로 서버는 클라이언트 역할을 하고, 클라이언트는 서버 역할을 한다.

그림 26에서는 TcpClient와 TcpListener의 동작원리를 나타내고 있다. TcpClient에서는 먼저 클라이언트 리스트와 센서 데이터를 데이터베이스에 접근해서 데이터 값들을 가져온다. 그리고 결과 값들이 화면에 잘 보일 수 있도록 ASCII 코드로 데이터의 값의 속성을 변환시킨다. 그래서 TcpClient Server를 연

결하고, 센서 데이터 값의 크기를 반환하고, 센싱 데이터를 Listener로 보낸 후에, 클라이언트를 종료한다. TcpListener 부분에서는 데이터들의 공간을 확보하기 위해서 수신버퍼 크기를 지정한다. 그리고 통신할 포트번호를 지정한다.

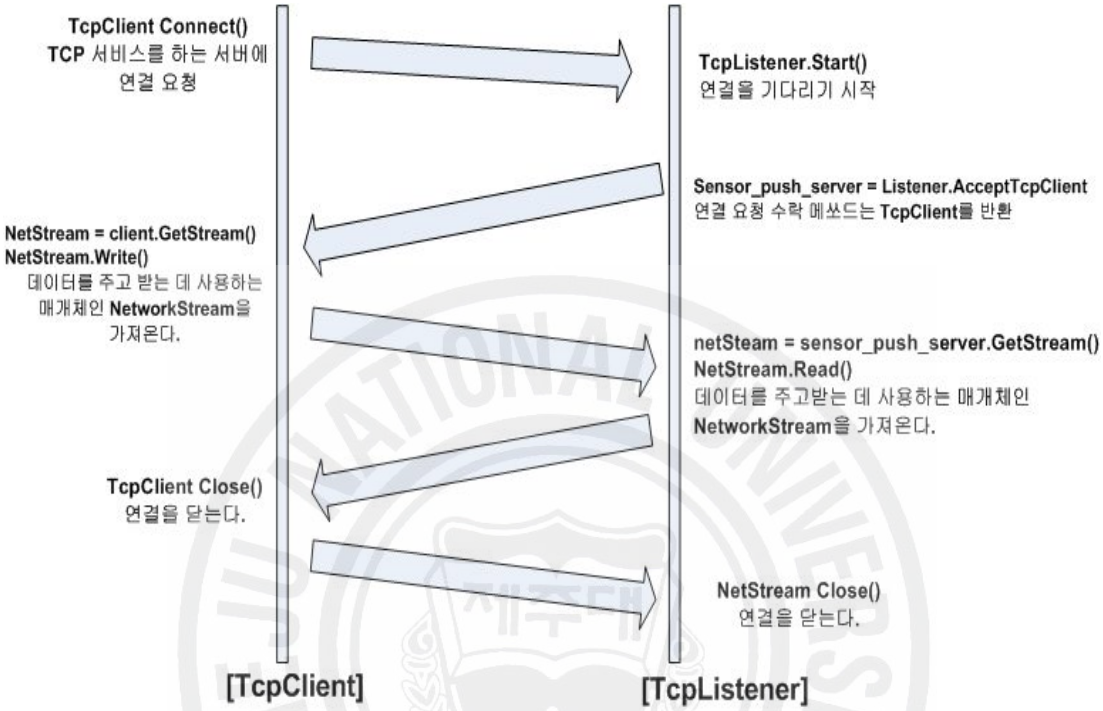


그림 26. TcpClient와 TcpListener의 동작원리

그 이후에 listener를 시작한다. 클라이언트에서 데이터를 받을 공간을 확보하기 위해서 바이트의 크기를 할당한다. 그리고 클라이언트와 연결요청을 수락하고 데이터의 송/수신 방법에 대해서 설정한다. 이때 TcpListener는 클라이언트의 연결 요청을 수락한다지만, 클라이언트에서 서버의 연결 요청을 허락해야 한다. 그 이후에 listener는 클라이언트에서 데이터를 받을 만큼 스트림을 만들어주고, Getstream,read를 한다. 마지막으로 listener는 client에서 받을 데이터 값의 문자 속성을 ASCII값으로 변환하고, netstream을 닫고, Listener를 닫는다.

그림 27은 클라이언트 접속 대기화면을 보여주고 있다. 일반 소켓에서는 클라이언트 측에서 서버 측으로 데이터 전달 요청을 한다. 그러나 여기서는 클라이언트에서 접속을 하고 대기를 한다. 클라이언트 측에서 대기를 하는 이유는 Push 서비스를 하기 위해서는 클라이언트 요청 없이 서버에서 데이터 정보를 보내기

때문이다. 클라이언트에서는 서버에서 센서 데이터를 보낼 때 까지 대기 상태를 유지한다.

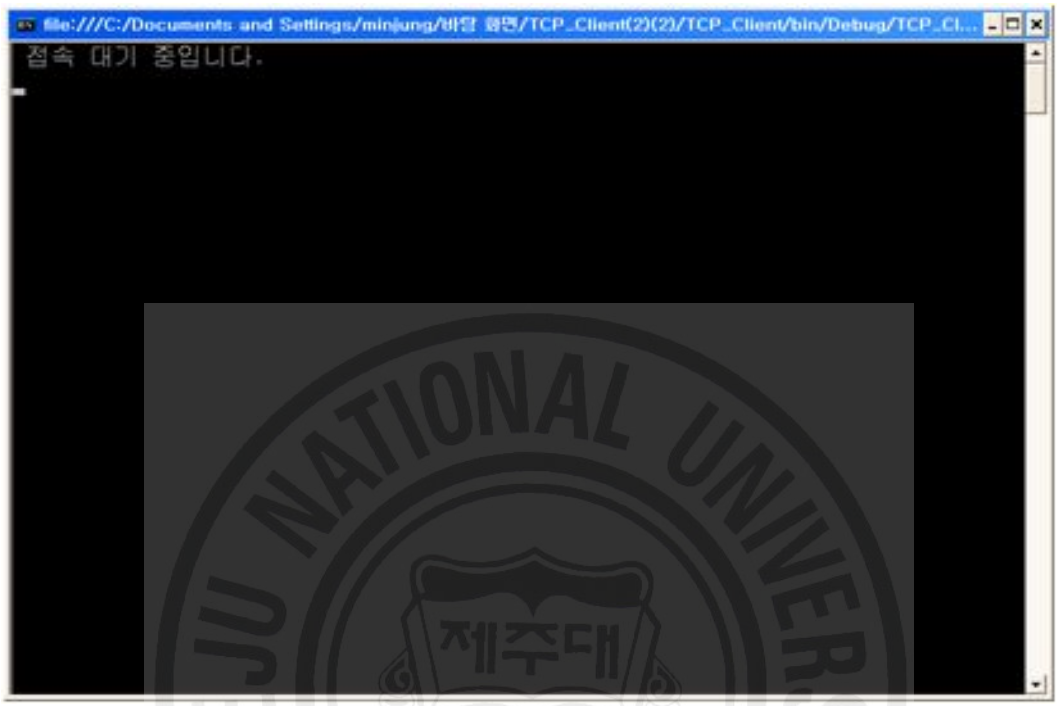


그림 27. 클라이언트 접속 대기 화면

그림 28는 클라이언트에서 서버의 데이터를 전송받는 화면을 보여주고 있다. 여기서는 센서 노드 번호, 데이터를 받은 날짜, 온도, 습도, 조도 등의 다양한 센서 정보를 나타내고 있다. 서버에서는 클라이언트의 이름, 포트 번호, 아이피 주소, URL 주소를 받아온다. 그래서 클라이언트가 서버에 요청을 하지 않아도 서버에서 클라이언트 정보 리스트에서 클라이언트 수신 측 목록을 받아오기 때문에 클라이언트에서는 서버에서 데이터를 보내면 바로 받아서 데이터 정보를 확인할 수 있다.

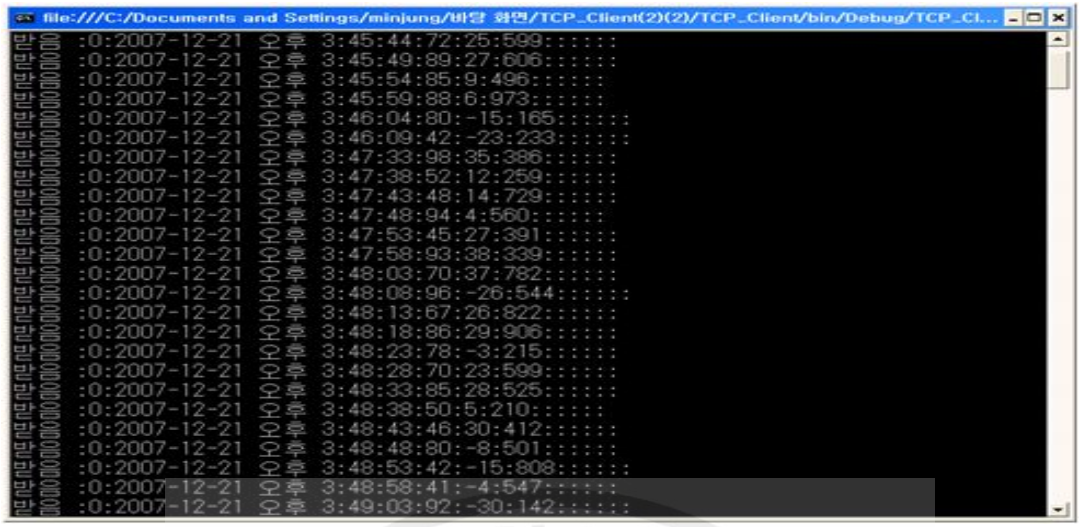


그림 28. 클라이언트에서 서버의 데이터를 전송받는 화면

그림 29는 서버가 클라이언트 정보 리스트에서 전달 받은 수신 측 클라이언트 리스트 목록 화면이다. 여기서는 서버에서 전달 받은 센서 노드 번호, 데이터를 받은 날짜 및 시간 그리고 센서의 온도, 습도, 조도 등의 데이터를 보여주고 있다. 소켓통신에서는 클라이언트에서 서버로 데이터 요청을 하지만, Push 서비스를 하기 위해서 클라이언트는 서버에 요청할 필요 없고, 서버에서 데이터를 보내면 데이터를 전달 받는다.

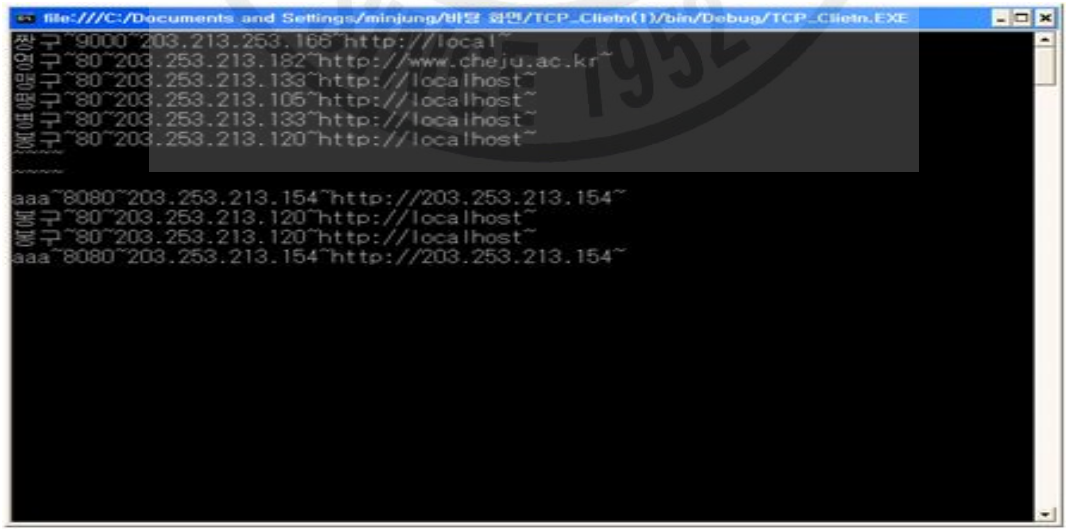


그림 29. 클라이언트 리스트 목록 화면

그림 30은 서버에 저장되어 있는 센서 데이터 정보를 클라이언트로 전송하는 화면이다. 이 화면은 센서 노드 번호와 데이터를 받은 날짜, 시간, 온도, 습도, 조도 등의 센서 데이터를 나타내고 있다. 소켓을 이용한 Push 서비스의 서버가 하는 역활은 클라이언트가 접속하면 클라이언트 정보 리스트 목록을 가져와서 아이피 주소와 클라이언트 측으로 온도, 습도, 조도 등의 센서 데이터 정보를 보낸다. 그래서 서버에서는 센서 노드 번호와 데이터를 받은 날짜, 온도, 습도 등의 정보들을 클라이언트로 보내서, 클라이언트에서는 데이터 요청 없이 서버에서 받은 센서 데이터 정보들을 쉽게 확인할 수 있다..

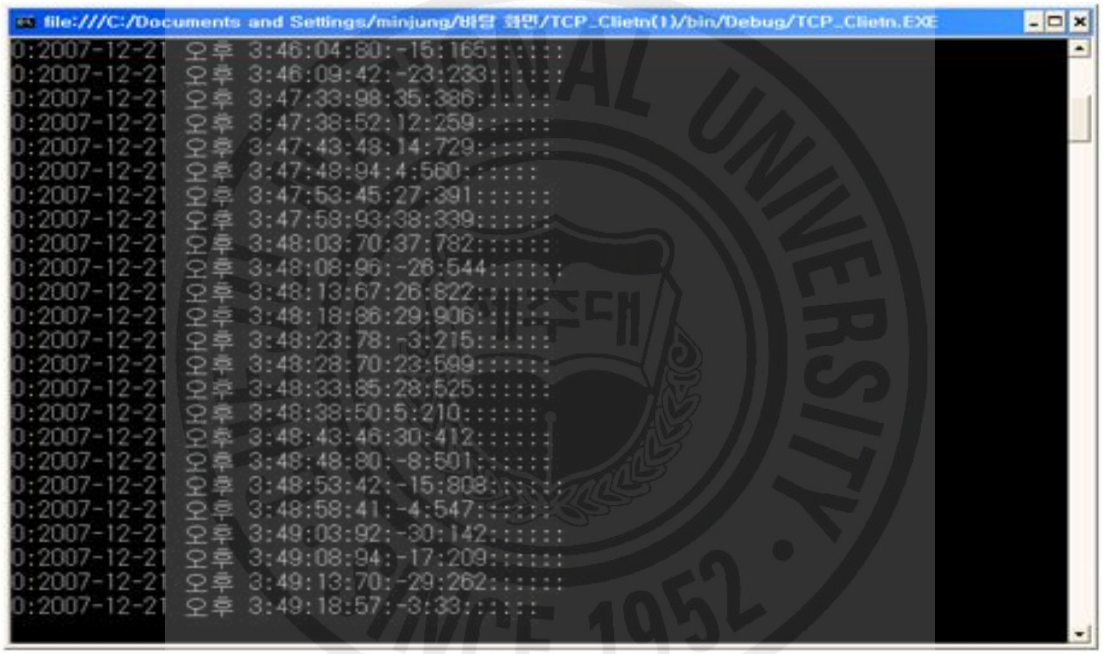


그림 30. 센서 데이터 정보를 클라이언트로 전송하는 서버 화면

# Ⅴ. 결 론

본 논문은 센서 네트워크 기반의 Pull/Push 서비스를 제공하기 위하여, 웹 서비스와 TCP/IP 소켓 기반의 개방형응용 인터페이스를 설계하고 구현한다. 개방형 인터페이스는 Pull서비스, Push서비스에 따른 각각의 장점을 가지고 있다. Pull 서비스는 개방형 표준을 준수하여 이기종 플랫폼과 이기종 언어 사이에서도 데이터 교환이 가능하다.

그리고 서버/클라이언트간의 정보 교환을 보다 안정적이고 쉽게 배포한다. 따라서 Pull 서비스는 닷넷 기반의 원격 서비스를 이용하여 센서 네트워크에 수집된 온도, 습도 등의 상황 정보를 제공한다. Push 서비스는 TCP/IP 소켓 통신을 이용하여 클라이언트가 응답하지 않아도 서버에서 실시간으로 데이터를 처리할 수 있고, 실시간으로 들어오는 데이터를 확인할 수 있기 때문에 서버와 클라이언트간의 통신이 용이하다. 그리고 Pull/Push 서비스는 기존 특정 데이터베이스 중심의 센서 네트워크의 폐쇄적인 응용 인터페이스를 개방적인 응용 인터페이스로 전환하여 사용자가 여러 곳에 널려있는 많은 센싱 정보를 쉽게 접근하여 확인할 것으로 기대한다.

# 참고문헌


[1] 강정훈, 황태호, 송병철, "유비쿼터스 센서네트워크 기술", 한국방송공학회지, 방송공학회지 제10권 제3호, pp. 39-52, 2005.9.

[2] 홍원표, 유비쿼터스 컴퓨팅 개념과 센서네트워크, 한국조명・전기설비학회, 조명・전기설비 제19권 제4호, pp. 4-28,, 2005.8.

[3] ETRI/VDC((2008... Active 시장: 33%예상) 추정 2005.12) VDC, THE RFID OVERVIEW(2005. 08)

[4] RFID/USN 기술 로드맵 ITRM 20012, 정보통신부・IITA 정보통신연구진흥원, 2007.1

[5] 강경옥, 김용우, 권훈, 김부림, 김도현, "Tiny-DB와 MySQL을 이용한 유비쿼터스 센서 네트워크 기반의 실시간 정보 서비스 설계 및 구현", 한국산학기술학회논문지 Vol.7, No.22 pp.175-181, 2006.4.

[6] 이상훈, 문승진, "TinyDB와 라이트레이서를 활용한 TinyOS기반의 센서 데이터 처리 모듈 설계 및 구현", 한국통신학회논문지 Vol.31 No.10B, pp. 883-890, 2006.10

[7] 김부림, 김도현, "센서 네트워크 기반의 웹 서비스 설계 및 구현", 대한임베디드공학회 추계학술대회, pp. 284-288, 2007.11

[8] 임성호, 김주만, "웹 서비스 기반 URC 로봇 원격 모니터링 기술의 설계 및 구현", 한국콘텐츠학회논문지, 제 6권 제 11호, pp. 285-294, 2006.11.

[9] 박유미, 최영일, 이병선, "웹 서비스 기반의 개방형 서비스 기술", 한국통신학회지, 제22권, 제5호, pp. 28-42, 2005.5.

[10] 한창환, 한연희, 길준민, 최장원, 윤준원, "Korea@Home 시스템에서 웹 서비스 Open API활용을 위한 설계", 한국정보기술학회 하계학술대회 논문집, pp. 464-469, 2007.6.

[11] 이원석, "웹 서비스(Web Service)", 디지털 행정, 제 25권, 제 2호, 82-92, 2002

[12] 강보영, 임경식, "개방형 표준 API 기반의 액티브 응용 서비스 망 구조", 한국정보과학회 2004년도 가을 학술발표논문집 제31권 제2호(Ⅲ), pp. 463-465, 2004.10.

[13] 김원영, 김치수, 김진수, "Push 기반 원격교육 시스템과 수준별 문항평가 알고리즘

에 대한 연구", 인터넷정보학회논문지, 제2권 제3호, pp. 19-25, 2001.8.

[14] 이노경, 전시현, "웹 환경에서 Push와 COM을 이용한 클라이언트간의 정보 공유화", 한국정보과학회 2001년도 가을 학술발표논문집, 제28권 제2호(Ⅲ), pp. 880-882, 2001, 10

[15] 한영태, 민덕기, "Push 기반의 이벤트 알림 서비스에 대한 연구", 2003년도 한국정보과학회 가을 학술발표논문집, 제 30권 제2호, pp.628-630, 2003.10.

[16] 송준근, 마평수, 박승민, "센서 네트워크용 초소형 OS", 한국통신학회지(정보와 통신), 제 24권 제 7호, pp. 26-35, 2007.7.

[17] 김영성, 김영환, 석정봉, "TinyOS 메시지 길이에 따른 에너지 절약 연구", 한국정보과학회 2006 한국컴퓨터종합학술대회 논문(D), pp. 343-345, 2006.6.

[181 이지현, 임경식, "Zone-control 기능을 지원하는 지능형 센서네트워크 미들웨어", 한국정보과학회 2004년도 가을 학술발표논문집 제31권 제2호, 628-630, 2004.10.

[19] 김정원, "유비쿼터스 혈압 측정 시스템의 설계 및 구현", 한국 컴퓨터정보학회 논문지 제 11권 제 6호, pp. 143-150, 2006.12.

[20] 김정현, 김대영, 성종우, 송형주, 김수현, "센서 네트워크에서 컨텍스트 인지 서비스 개발을 위한 미들웨어", 한국컴퓨터종합학술대회 2005 논문집 제 32권 제 1호, 2005, 7

[21] 정종태, 정국상, 최덕재, "센서와 미들웨어간의 통신을 위한 아키텍처 설계 및 구현", 한국컴퓨터종합학술대회 2005 논문집, 제 32권 제1호, pp. 535-537, 2005.7.

[22] 이노경, 전시현, "웹 환경에서의 Push와 COM을 이용한 클라이언트간의 정보 공유와 이동", 2001년도 가을 학술발표논문집 제28권 제2호, 880-882, 2001.10.

[23] 최영일, "유비쿼터스를 위한 BcN의 개방형 서비스 인터페이스", 인터넷정보학회지 52 ~ 62 제6권 제3호, pp. 52-62, 2005.9.

[24] 최영관, "소설같은 C# Novel C#", 도서출판 자북, 서울, 2003.

[25] 박상현, "비주얼 C# 2005 익스프렛로 배우는 C# 2.0 프로그래밍", 도서출판 대림, 2007.

[26] http://library.etri.re.kr